On Learning in Functional Causal Models

Submitted in partial fulfillment of the requirements for the degree of

Master of Technology

by

Devdatta Kathale
(153079001)

Guide
Professor Girish Kumar
Co guide
Professor S. Nath

Department of Electrical Engineering
Indian Institute of Technology Bombay
2017-2018
Abstract

In this work, we consider the problem of estimating the functions that would be appropriate for computing the values of the child nodes, given the values of the parent nodes, in Functional Causal Models. We use Generative Neural Networks as a tool to infer the causal direction in Functional Causal Models. We describe another method to infer the causal direction, which turned out to be based on an incorrect assumption. We pose the Active Learning problem as a Bandit problem, and draw parallels between the Active Learning problem and the Bandit problem. We leverage the Upper Confidence Bound (UCB) algorithm for Active Learning. Then we compare the results of the experiments performed on simple cases of Functional Causal Models to check if the UCB algorithm gives better results as compared to the baseline methods.
Contents

1 Introduction ........................................ 8
   1.1 FCMs, Function Estimation Problem ............. 8
   1.2 Causal Inference ................................ 9
   1.3 Active Learning ................................ 10

2 Literature Review ................................ 13
   2.1 Functional Causal Models ......................... 13
      2.1.1 Functional Causal Models .................. 13
      2.1.2 Observational and Interventional Distributions of an
             FCM ........................................ 14
      2.1.3 Assumptions ................................ 15
      2.1.4 FCM Paper by [5] ............................ 16
      2.1.5 Additive Noise Assumption .................. 20
2.2 Causal Inference .................................................. 20
   2.2.1 Maximally Separated Subsets ......................... 20
   2.2.2 Problem Addressed ................................. 20
   2.2.3 Methodology Used ............................... 21
2.3 Kernel Embedding ............................................. 21
   2.3.1 Kernels ........................................... 21
   2.3.2 Recursive Kernel Hilbert Space and Kernel Trick ... 22
2.4 Causal Generative Neural Networks ....................... 23
   2.4.1 Causal Generative Neural Networks ............. 23
   2.4.2 No Additive Noise Assumption .................. 24
   2.4.3 Uses Neural Networks to generate data sample and mimic input distribution ... 24
2.5 Active Learning .............................................. 25
   2.5.1 Active Learning and Causality ................. 25
   2.5.2 Bandit problems and UCB .................... 26

3 Modifications we made ........................................ 29
   3.1 To CGNN .................................................. 29
      3.1.1 Noises as parameters ........................... 29
      3.1.2 Change in the objective function ......... 30
3.1.3 Enforcing Gaussianity of noise using MMD .......... 30
3.1.4 Enforcing Independence of input and noise using HSIC 30

3.2 To the method in Indra’s problem ......................... 31
3.2.1 Use of Divisive Clustering, its advantage over Agglomerative Clustering ......................... 31
3.2.2 Using patches of datapoints instead of individual datapoints ........................................ 32
3.2.3 Splitting the dataset into small patches by using trees . 33
3.2.4 The assumption that turned out to be incorrect ...... 33

3.3 To Active Learning Problem ............................. 34
3.3.1 Parallels between UCB and Active Learning ........... 34

4 Results .................................................................. 36
4.1 Modified CGNN ..................................................... 36
4.1.1 Ability of NoiseMMD to identify the Normal Distribution 36
4.1.2 Testing Goodness of Fit ................................. 37
4.1.3 Test for Generalization ................................ 38
4.1.4 Accuracy of Causal Inference ......................... 38

4.2 Active Learning Experiments and Results .................. 39
4.2.1 Algorithms Compared ....................................... 39
4.2.2 FCMs Used ................................................. 40
4.2.3 Experiment 1 .............................................. 43
4.2.4 Experiment 2 .............................................. 44
4.2.5 Experiment 3 .............................................. 45
4.2.6 Modified Sampling Method ................................. 53
4.2.7 Experiment on Modified Sampling Method ............... 54

5 Conclusions ................................................. 60

6 Future Work ................................................. 62

6.1 Connection of Active Learning problem to Bandits ........ 62

6.2 Different statistics that may be used in the CGNN training
objective ......................................................... 63
## List of Figures

1.1 Example of a Causal graph ................................................. 11

4.1 FCM 1: The SCM used in Experiments 1 and 2 ...................... 40

4.2 FCM 2: The SCM used in Experiment 3 ............................... 41

4.3 Experiment 1: Mean Square Error of estimates at each time step. ................................................................. 46

4.4 Experiment 1: True function $f_1$ and its estimates by algorithms. 47

4.5 Experiment 1: True function $f_2$ and its estimates by algorithms. 48

4.6 Experiment 1: True function $f_3$ and its estimates by algorithms. 49

4.7 Experiment 1: True function $f_4$ and its estimates by algorithms. 50

4.8 Experiment 2: Mean Square Errors for multiple instances of algorithms. ................................................................. 51

4.9 Experiment 3: Mean Square Errors for algorithms on FCM 2. 52

4.10 Experiment 1: Mean Square Error of estimates at each time step. ................................................................. 55
4.11 Experiment 1: True function $f_1$ and its estimates by algorithms. 56
4.12 Experiment 1: True function $f_2$ and its estimates by algorithms. 57
4.13 Experiment 1: True function $f_3$ and its estimates by algorithms. 58
4.14 Experiment 1: True function $f_4$ and its estimates by algorithms. 59
List of Tables

4.1 Ability of NoiseMMD to identify the Normal distribution . . . 37
4.2 Values of Errors for training with different Objectives . . . . . 37
4.3 Generalization Results on Seen and Unseen Data . . . . . . . 38
Chapter 1

Introduction

1.1 FCMs, Function Estimation Problem

We can model the events as variables, and the cause-effect relationship can be represented as a directed graph. In these graphs, the arrow would start at the cause and go towards the effect.

Most of the literature on causality is focused on using samples in order to obtain the causal graph. This is known as causal discovery or causal inference. It is motivated by realistic problems in science. For example, a biologist may wish to discover which genes regulate which other genes, or, a public health researcher may wish to determine whether or not certain habits in a population influence specific health outcomes [5].

Our work considers causal discovery and the step after that. We have worked on obtaining the causal direction from observational data, and on obtaining the precise functional relationships assuming that the causal direction is known.
As a trivial example of causal discovery, consider the altitude and temperature readings taken in a region. It is clear that the two will be correlated. But, it is not obvious from the data whether temperature is the cause of altitude or is it the other way round.

As an example of finding the functional relationship, consider a cell in a living organism. Suppose that gene A down-regulates gene B, which in turn up-regulates gene C. This is the causal relationship that we know. But we do not know the precise functional form of the relationship. So, we cannot predict quantitatively what will happen if the expression of gene A is suppressed by, say, 10% or 20% or by 30%.

As another example, we cannot answer questions like, what steps will reduce the overall occurrence of lung cancer? Would it be better to convince 50% of the smokers to quit smoking completely, or would it be better to convince each smoker to cut down the consumption by 50%? [5]

In the above examples, we know something about the relationships. But, this information is not sufficient to make predictions. Nor is it helpful to take specific decisions.

1.2 Causal Inference

Causal inference, or causal discovery is the process of finding causal relationships between variables. Causal inference in the context of FCMs is, given a set of nodes representing variables, obtain the causal directions. The causal directions are represented as directed edges in the FCM graph.

Causal inference can be used to find relations between variables. Suppose we have observational data about multiple variables. We know that some of
the variables are correlated. The question we want to answer is, can we infer from the observational data, which variable is the cause of which other variables?

In the context of FCMs, each of the variables is a node in the causal graph. The task is that of finding the directed edges of the causal graph.

1.3 Active Learning

An agent is an autonomous entity, that senses the environment around it using sensors, and acts on the environment using actuators. The activity of an agent is directed towards achieving some goals [6].

In the case of an agent that does not influence the generation of the data, the problem of obtaining the functional relationships is equivalent to solving a set of independent regression problems, after the causal graph is known. But many problems of getting the functions can be looked at as a sequential decision making process. A decision made based on observed data at one point affects the data that is received at the next point. We analyze the data that we have obtained, and on the basis of that we decide on the next experiment to be performed.

In an Functional Causal Model (FCM), there are two types of variables, causal parent variables, and caused child variables. The FCM consists of functions that connects these.

An intervention is an action by which one or more variables are forced to take a specific value. So, an intervention is a mathematical representation of performing an experiment. The intervened variables are not calculated from their parents. Non-intervened variables are calculated from their parents and
We estimate the functional relationships between parents and children using a Bayesian approach.

As an example, consider three variables whose causal relationship is given by the graph shown in Figure 1.1.

We aim to learn function $f_1$ and function $f_2$. In each step, we have to take the decision of

1. performing one of the possible interventions, or
2. passively observe the system.

Every action that is chosen will have a cost.

With the choice of observing passively, we will learn partial information about both $f_1$ and $f_2$, but this learning will only be in areas where there is probability mass under the distributions of $X_1$ and $X_2$. When the number of samples is low, then in areas where the probability of the parents is low, there will be no learning.

By doing an intervention, we can choose exactly where we want to learn a particular function. If we perform an intervention on $X_2$, we can choose
where in the domain we want to learn \( f_3 \). But, we do not learn anything about the function \( f_2 \).

At each point in time, we must decide on which intervention out of the given set of interventions we would perform. Or, we may choose not to intervene, and make a passive observation about the system. In case of the passive observation, we would learn about the system in the areas where the parents have high probability mass. However, when we have a small number of samples, we are unlikely to learn anything about the function where the parents have low probability mass. In case of intervention, we can choose specifically where in the domain we want to learn the function. However, we learn nothing about the function of the parents of the intervened variable. The problem is to pick which intervention to perform, given all the data that we have about the system.

Also, we associate each intervention with a positive cost. This helps to generalize the model to scenarios where performing certain experiments could be expensive or potentially dangerous.

We formalize these ideas by using the framework of Functional Causal Models, which have been mentioned briefly earlier in this chapter, and will be elaborated on in the next chapter.
Chapter 2

Literature Review

2.1 Functional Causal Models

2.1.1 Functional Causal Models

Here we define Functional Causal Models (FCMs), which will enable us to formalize the problem of obtaining functional relationships between variables.

**Definition 1:** Suppose that $X_1, \ldots, X_N$ and $E_1, \ldots, E_N$ are variables with each $X_n$ and $E_n$ taking values in the domains $X_n = \mathbb{R}$ and $E_n = \mathbb{R}$ respectively [5]. Denote by $X$ the vector of variables $(X_n)_{n=1,\ldots,N}$, and by $\mathcal{X}$ the domain of the variables in the vector $X$, $\prod_{n=1,\ldots,N} X_n$, similar for $E$ and $\mathcal{E}$. A Functional Causal Model $\mathcal{M}$ is a tuple $(\mathcal{S}, \mathcal{I}, \mathcal{P}_E)$ consisting of the following quantities:

- $\mathcal{S}$ is the set of structural equations $X_n = f_n(X_{pa(n)}, E_n)$ where $pa(n) \subset \{1, \ldots, N\}$ and $X_{pa(n)}$ is the vector of variables $(X_m)_{m \in pa(n)}$. We refer
to $X_{pa(n)}$ as the causal parents of $X_n$.

- $\mathcal{I}$ is the set of interventions. An intervention is an operation that replaces a subset of equations in $\mathcal{S}$ by equations setting the variables to specific constants. We write the intervention $i \in \mathcal{I}$ that intervenes on a subset of variables $X_{\text{var}(i)}$, $\text{var}(i) \subseteq \{1, \ldots, N\}$, setting them to values $x_{\text{val}(i)}$, as $\text{do}(X_{\text{var}(i)} = x_{\text{val}(i)})$. We will also use the notation $\text{do}(i)$ for brevity. We write the resulting equations as $\mathcal{S}^{\text{do}(i)}$.

- $\mathbb{P}_E$ is the distribution over the exogenous/noise/unexplained variables $E$ taking values in $\mathcal{E} = \mathbb{R}^N$. This distribution is fixed and does not change due to interventions.

The ability to perform interventions makes it possible for the learning algorithm to query. The data obtained under the chosen intervention is the label. These labels of queried instances can help the algorithm learn the unknown functions with a small number of examples.

### 2.1.2 Observational and Interventional Distributions of an FCM

Directed acyclic graphs are finite graphs in which, starting from a particular node $v$, there is no path of consistently directed edges that comes back to $v$.

We consider only the FCMs whose causal graphs are directed acyclic graphs. Note that the acyclic SCM implies that given the error values, there is a unique $x \in X$ that satisfies the structural equations. This implication can be seen by calculating the value of the root nodes using the corresponding exogenous values, and then finding the values at the child nodes by substituting in the downstream equations and adding the appropriate exogenous values at the child nodes.
By a similar reasoning, it follows that any intervention will also yield an acyclic set of equations. Thus, given the value $e$ for the exogenous variable, there exists only one solution $x$ to the set of structural equations.

However, $E_1, \ldots, E_n$ are random variables with known distributions. This, along with the fact that a given $e$ admits only one solution $x$, implies that there is a distribution induced on $\mathcal{X}$ due to $E$.

This is called the Observational Distribution of $X$. Let us denote this distribution by $P_{\mathcal{X}}^{do(\emptyset)}$, where $\emptyset$ signifies the empty intervention, i.e. no intervention.

Similarly, we can think of each intervention inducing a distribution. This distribution is called the interventional distribution for that particular intervention. Similar to the observational distribution, let us denote the interventional distribution under the intervention $i$ by $P_{\mathcal{X}}^{do(i)}$.

Here we should observe that given the structure of the graph, the only parameters in the model $\mathcal{M}$ are the functions $f_n$ and the distribution $P_E$ over the variables $E$.

### 2.1.3 Assumptions

We make the non-trivial assumption that each function $f_n$ is a non-linear function of $X_{pa(n)}$, and has additive Gaussian Noise with zero mean. That is, each node equation is of the form $X_n = f_n(X_{pa(n)}) + E_n$, where $E_n$ is a zero-mean Gaussian random variable. We also assume that the $E_n$ at different nodes are independent. That is, $P_E \sim \mathcal{N}(0, \Lambda)$, where $\Lambda$ is diagonal. We assume that the entries in $\Lambda$ are known.
2.1.4 FCM Paper by [5]

The problem of estimating functions in an FCM has been addressed by [5].

Gaussian Process Regression

The problem that we are trying to solve is the problem of estimating functions or learning functions, given samples of data. One method to do this is that of Gaussian Process Regression. We describe this method in brief.

For estimating a real function $f_n$, we consider a random process on the domain of $f_n$. We assume that this random process is a Gaussian Process.

Having assumed this, we also assume a prior belief distribution on the functions. Also, we fix a mean function and covariance function for the Gaussian Process. The covariance function is also called the kernel of the Gaussian Process.

Here, we assume a zero-mean prior distribution for the functions $f_n$. That is,

$$f_n \sim \mathcal{GP}(0, k_n)$$

Given the prior mean function and the kernel of the Gaussian Process, it is completely defined.

Suppose that there is a function that is unknown, and all we have is its values at some points in its domain. That is, we have data $\mathcal{D}$ of the form $(i, x)$, where $i$ is the intervention performed, and $x \sim \mathbb{P}_{X}^{do(i)}$ is the vector of the known function values. Given this data, we can compute the posterior
distribution of the Gaussian Process. The posterior distribution is also a
Gaussian Process, but with different parameters.

\[ f_n|D_n \sim \mathcal{GP}(\mu_{f_n|D_n}, k_{f_n|D_n}) \]

where \( \mu_{f_n|D_n} \) and \( k_{f_n|D_n} \) can be written explicitly in terms of \( k_n \) and the
data \( D \).

The same kind of regression as above can be done for each \( f_n \). This will
give the posterior distribution over the \( n \)-tuple of functions \( f = (f_1, \ldots, f_n) \).

**Loss Functional and Risk**

We need a metric to assess how close our estimate of the functions is to the
actual functions. We define an error function that we wish to minimize. We
also assume that we are given a measure \( \Pi_n \) for each \( n \in \{1, \ldots, N\} \), which
specifies the importance of estimating or learning the function \( f_n \) at each
point in its domain. \( \Pi_n \) is such that it puts a large weight in the areas where
we need to learn \( f_n \) more accurately and a small weight in areas where only
an approximation would suffice. A zero weight in a specific area implies that
we do not care to learn \( f_n \) in that area of the domain.

We define the Mean Squared Error as the loss function.

For a single estimated function \( \hat{f}_n \), the Mean Integrated Squared Error is
defined as

\[ L_n(\hat{f}_n \parallel f_n) = \int_{X_{\Pi_n(n)}} (f_n(x) - \hat{f}_n(x))^2 \, d\Pi_n \]
To define the loss function for the entire set of functions \((f_n)_{n\in\{1,\ldots,N\}}\), we simply consider a weighted sum of the individual Mean Integrated Square Errors,

\[
L(f \parallel \hat{f}) = \sum_{n=1}^{N} \alpha_n L_n(f_n \parallel \hat{f}_n), \quad \alpha_n > 0 \quad \forall \ n \in \{1, \ldots, N\}
\]

The above is the total loss function. \(\hat{f}\) is the tuple of the estimated functions \((\hat{f}_n)_{n\in\{1,\ldots,N\}}\) and \(f\) is the tuple \((f_n)_{n\in\{1,\ldots,N\}}\).

It has been shown in [5] that the posterior mean function of the Gaussian Process minimizes the Expected Mean Integrated Square Error function. So \(\hat{f}_n = \mu_{f_n|D_n}\) is the best estimate of the function \(f_n\) in the Mean Integrated Square Error sense.

For a given prior, \(\hat{f}\), is a function of \(D\) alone, and the minimum of the loss function is also a function of \(D\) alone. Using Bayes rule, given the data \(D\) and the prior distribution, the posterior distribution for \(f\) can be found. Hence the expected total risk defined by \(E_{f|D}[L(\mu_{f|D} \parallel f)]\) is a function of \(D\) alone. We denote the expected total risk by \(R(D)\).

**Sampling Approach used in [5]**

An algorithm for sampling from the believed interventional distribution has been presented in [5].

The algorithm uses the fact that the believed interventional distribution can be factorized into Gaussian distributions.

Let us denote the believed interventional distribution by \(\mathbb{P}_{X}^{do(i)}\).
Consider the variables that are parentless, but not intervened upon. Say $X_n$ is such a variable. This variable is the sum of a constant function (which is unknown), and an exogenous variable (whose variance is known). Since samples of the variable are available, we can estimate the constant function. The exogenous variable is independent of the constant function. The parentless variable can thus be estimated by drawing it from the distribution $\mathcal{N}(\mu_n|D_n, k_n|D_n + \sigma_n^2)$. For every variable which is parentless and not intervened upon, we can draw it from its corresponding distribution.

The values of the intervened variables are forced to specific constants by definition.

Now, consider the child variables. Let $X_n$ be a child variable. However, $f_n$ is uncertain. We have assumed $f \sim \mathcal{GP}(\mu_{f_n}|D_n, k_{f_n}|D_n)$. $X_n$ is the sum of the $f_n(X_{pa(n)})$ and $E_n$. Given the values of the parents $X_{pa(n)}$, the conditional distribution is

$$X_n|X_{pa(n)} \sim \mathcal{N}(\mu_{f_n}|D_n(X_{pa(n)}), k_{f_n}|D_n(X_{pa(n)}, X_{pa(n)}) + \sigma_n^2)$$

Given the parent values, we can draw a sample of $X_n$ from this distribution.

Thus, we can draw samples of the vector of variables $X$ from a believed interventional distribution.

We can estimate the usefulness of an intervention using this sampling method. This approach has been used in [5]. For every possible intervention $i$, they draw several samples $x^i$. For every sample, they find the risk $R(D \cup \{(i, x^i)\})$. The average over these risks serves as an estimate of the risk that would result if intervention $i$ is chosen. Finally, the intervention with the
least estimated risk is chosen.

2.1.5 Additive Noise Assumption

The work done by Rubenstein et. al. [5] is based on the assumption that the noise at each node is additive. The output can be any function of the parent nodes, but the noise term may only be added to it. This assumption is quite restrictive, and may not hold in practice. Thus, methods that do not rely on this assumption are desirable.

2.2 Causal Inference

2.2.1 Maximally Separated Subsets

Having explored Functional Causal Models, we were interested in following the problem of causal inference using Maximally Separated subsets, addressed by Indradyumna in his thesis [4].

The sections ahead explain the problem addressed and the methodology used by them.

2.2.2 Problem Addressed

The problem here is to determine the causal direction. That is, whether $X$ is the cause of $Y$, or vice versa. The standard method to determine causality is by using interventions. But there are scenarios where interventions can be excessively costly, harmful or unethical. In such scenarios, we must rely on
observational data to make inference about the causal direction.

Causation is different from correlation. Correlation between two variables $X$ and $Y$ indicates that there is some relation between them. This could be because $X$ causes $Y$, or because $Y$ causes $X$, or $X$ and $Y$ have a common cause.

2.2.3 Methodology Used

The basic idea is to split the dataset into two subsets.

The causal mechanism is going to be the same for the correct causal direction, no matter what subset is being considered. Thus, $f_{Y|X}$ will be the same for any subset of data, if $Y$ is the effect as $X$ is the cause.

$$f_{Y|X} = \frac{f_{X|Y}f_{i}}{f_{X}} = \frac{f_{X|Y}f_{j}}{f_{X}}$$

The subsets are to be chosen such that they are maximally in the following sense:

$$\left\| f_{X}f_{i}^{j} - f_{X}f_{j}^{i} \right\|$$

2.3 Kernel Embedding

2.3.1 Kernels

Extraction of features is an essential step in any Machine Learning algorithm. The purpose of extracting features is to map the data to a feature space where
it can be used for learning. There are several features that are commonly used. Higher order statistical features are captured using high-dimensional non-linear functions. Eg. Polynomial features of higher order. Many of these features are complicated functions of the data.

The idea behind kernel embedding is to map each of the data points to an infinite-dimensional feature space. The feature maps are non-parametric in the sense that the All further operations are then carried out in the infinite-dimensional feature space.

The kernel features have been successfully applied to problems in the domains of Image and Video Processing.

2.3.2 Recursive Kernel Hilbert Space and Kernel Trick

A Recursive Kernel Hilbert Space (RKHS) is a vector space with an inner product defined on its elements.

A reproducible kernel Hilbert Space $\mathcal{F}$ on $\Omega$ with kernel $k$ is a Hilbert space of functions $f : \Omega \rightarrow \mathbb{R},$

$$< f(\cdot), k(x, \cdot) >_{\mathcal{F}} = f(x)$$

This property allows us to do function evaluations as inner products.

Let $k : \Omega \times \Omega \rightarrow \mathbb{R}$ be a kernel

Since $k(x', \cdot)$ is also a function $\Omega \rightarrow \mathbb{R},$ $< k(x', \cdot), k(x, \cdot) >_{\mathcal{F}} = k(x', x)$
The inner products for finite samples can be expressed in the form of kernel evaluations.
\[ \phi(x) = k(x, .) \] can be considered a feature map.

For the random variable \( X \), the feature embedding is

\[ \mu_X = \mathbb{E}[\phi(X)] = \int_{\Omega} \phi(X) dP(X) \]

## 2.4 Causal Generative Neural Networks

### 2.4.1 Causal Generative Neural Networks

Causal Generative Neural Networks (CGNNs) learn Functional Causal Models as generative neural networks. They are trained by backpropagation to minimize the Maximum Mean Discrepancy (MMD) between the observational and generated data [3].

The representational power of CGNN is leveraged to generate the effect as a function of the cause and noise. We try two models. In the first, \( X \) is taken as the cause. We train the neural network to generate \( Y \) from \( X \) and noise samples. In the second model, \( Y \) is taken as the cause, and the network is trained to generate \( X \) from \( Y \) and noise inputs.

The premise here is that, whichever causal direction is correct, can be better fit by the neural network. Thus, the values of the objective achieved by each of the models can be compared to infer the causal direction. If the network where \( X \) is taken as the cause does better than the other directions, \( X \) is declared as the cause.

This approach can be extended to the multivariate case. Suppose there are \( n \) nodes in the FCM, and the causal structure is not known. We look at
the correlation between each pair of nodes. If two variables are uncorrelated, we do not draw and edge between them. If they are correlated, we draw an undirected edge between the variables. Finally, we get an undirected graph.

Now, the neural network fitting method described above is used to determine the orientation of each of the edges. The FCMs are assumed are to acyclic, so any directed cycles existing in the graph must be broken.

2.4.2 No Additive Noise Assumption

The work done on causal inference so far assumes the additivity of noise at each node.

This is an approach that does not make this restrictive assumption, and allows every node to be a function of the parent nodes and noise, where noise is not necessarily additive. This approach gives the flexibility that the output may be any function of the input and the noise. This is an advantage because, in practical situations, we can give no guarantees about the additivity of noise.

2.4.3 Uses Neural Networks to generate data sample and mimic input distribution

The objective used in CGNNs is the MMD between the observational joint distribution and the generated joint distribution. MMD is a measure of how different the distributions are. The finite sample MMD is being used here.

An advantage of MMD is that it can be expressed entirely in terms of the inner products of elements of the RKHS. This allows us to use the kernel
trick, and obtain an expression in terms of the kernel values between the points in the dataset.

Another advantage is that the differentiability of MMD with respect to the parameters is guaranteed when the Gaussian Kernel is used. The differentiability is important for training the neural network using backpropagation.

Finally, the MMD with Gaussian Kernel guarantees that the distance between samples from the same distribution will tend to zero as the sample size is increased.

2.5 Active Learning

2.5.1 Active Learning and Causality

Active Learning

Active Learning is a semi-supervised form of Machine Learning where the learning algorithm can actively query the information source to obtain desired output at new data points [7].

In active learning, the learner is “curious”. It chooses the examples to learn from. The algorithm asks queries in the form of instances [7]. The information source, or the oracle provides a label for the queried instance. The hypothesis of active learning is that the learner would learn better with less training.

Active learning is applied to scenarios where labeled instances are expensive to obtain, and we can supply the label for an instance that the algorithm queries. The labels for the queries have costs associated with them.
In such a setting, the algorithm has to intelligently choose the queries. It should choose the queries that would enable it to obtain the maximum amount of information.

### 2.5.2 Bandit problems and UCB

**Background**

The bandit problem is a problem of selecting an action, based on the information about the outcomes of the actions until that point in time.

The word "bandit" here refers to a slot machine. Pulling the arm of the slot machine gives a reward. Suppose now that we are facing a bandit with $K$ arms. Each arm has a reward distribution associated with it. We do not know the reward distribution. We have to take decisions about which arm to pull, with an objective of maximizing the reward that we will obtain.

Had the reward distributions of the $K$ arms been known, we could have simply selected the arm which has the highest expected reward. But, we do not know the reward distributions, so we must make some exploratory moves to get an estimate of the expected reward of each arm. After we are fairly confident about our estimates of the average rewards from each of the bandit arms, we can then take the arm that has the highest expected reward according to our estimate. This is called exploiting the best arm. [8]

But because the rewards are stochastic, we can never be sure about which arm is the best. This is what makes the bandit problem non-trivial. Several algorithms have been proposed in the literature to solve the bandit problem. The Upper Confidence Bound algorithm is one such algorithm.
UCB Algorithm for Bandit Problems

The Upper Confidence Bound (UCB) algorithm is a popular algorithm that is used for solving bandit problems.

The idea behind the algorithm is "optimism in the face of uncertainty" [2]. Specifically, in addition to calculating the expected reward of each arm, we also obtain the confidence interval around the expected reward. And, the algorithm chooses the arm for which the upper confidence bound is higher.

As we take more samples from any arm, our estimate of the mean gets closer to the true mean with greater confidence. The confidence interval that we consider around the estimated expected reward shrinks.

Let $k^*$ be the arm with the highest mean reward.

For a given algorithm, the regret after $T$ rounds is defined as the expected difference between the total reward obtained by the algorithm in $T$ rounds and the total reward that would have been obtained if the arm $k^*$ would have been pulled in each of the $T$ rounds.

In the case of the UCB algorithm being applied to bandits with stationary reward distributions, it is guaranteed that the regret after $T$ rounds is upper bounded by $O(\sqrt{KT \log(T)})$.

The UCB algorithm is as follows:

- Take one sample from each arm. For each arm if the empirical expected reward is calculated, it turns out to be the reward obtained from the sample of that arm.

- Calculate the value of the following expression for all arms $k$,
\[ \bar{x}_k + \sqrt{\frac{2\ln(t)}{n_k}} \]

where,

- \( n_k \) is the number of times arm \( k \) has previously been pulled
- \( t \) is the total number of pulls (of all arms combined) till this time step
- \( \bar{x}_k \) is the empirical estimate of the expected reward from arm \( k \). The empirical estimate is calculated as the total reward obtained from arm \( k \) divided by the number of pulls of arm \( k \).

- Identify the \( k \) that gives the maximum value of the UCB.
- Pick the \( k^{th} \) arm. Sample it.
- After obtaining the reward \( r \) at time step \( t \), update the empirical expected reward of the arm that was pulled. So, replace \( \bar{x}_k \) by \( \bar{x}_k \left( \frac{n_k}{n_k+1} \right) + r \left( \frac{1}{n_k+1} \right) \). Also, replace \( n_k \) by \( n_k + 1 \).

Observe that because the first step in the algorithm enforces taking each action at least once, the empirical rewards are well defined before the Upper Confidence Bound is evaluated.

Similarly, for any arm, \( n_k \) is an integer that is greater than or equal to one, so there is no chance of division by zero in the Upper Confidence Bound expression.

Hence, the Upper Confidence Bound expression is well defined.
Chapter 3

Modifications we made

3.1 To CGNN

3.1.1 Noises as parameters

We have assumed that the output is $y = f(x, N)$ where $N$ is the noise. We assume that $N$ is normally distributed with mean 0 and standard deviation of 1. The noise variables are independent of $x$. We set noises as parameters for the network. In other words, the network will optimize for the noise variables that will be used to generate the output $y$. We cannot arbitrarily assign values to the noise variables. There are conditions that the noise variables must satisfy. 1. The noise is normally distributed with mean 0 and variance 1. 2. The noise is independent of the $x$. We modified the objective in order to enforce these conditions during the optimization.
3.1.2 Change in the objective function

Goudet et. al. use the MMD between the given joint distribution and the generated joint distribution as their objective to train their neural network. We modified the objective so that it captures the characteristics of noise described above. We added two terms to the objective.

\[
\text{Objective} = w_1 \cdot \text{MMD}(\text{observed Joint}, \text{generated Joint}) + w_2 \cdot \text{MMD}(\text{noise Params}, \text{normal Samples}) + w_3 \cdot \text{HSIC}(\text{cause}, \text{noise Params})
\]

3.1.3 Enforcing Gaussianity of noise using MMD

This term was added to ensure that the estimated noise parameters are normally distributed. This term is calculated as,

\[
\text{NoiseMMD} = \langle \mu_Z, \mu_Z \rangle + \langle \mu_V, \mu_V \rangle - 2 \cdot \langle \mu_Z, \mu_V \rangle
\]

Where \( V \) represents the noise parameters, \( Z \) represents samples drawn from an independent standard normal distribution.

We compute the MMD between our estimated noise parameters, and an independent random sample drawn from the standard normal distribution. We take a larger sample from the reference normal distribution to reduce the sampling error.

3.1.4 Enforcing Independence of input and noise using HSIC

The second property that the noise parameters must satisfy is that, the noise parameters should be independent of the input \( x \). We enforce this using
Hilbert Space Independence Criteria (HSIC).

HSIC between $X$ and $Y$ is defined as,

$$HSIC = \| \mu_X \mu_Y - \mu_{XY} \|^2$$

If $X$ and $Y$ are independent, the HSIC between them is zero, because $\mu_X \mu_Y = \mu_{XY}$. In other cases, the HSIC returns a positive value.

Here we consider the HSIC between the noise parameters and the cause variable we are testing for.

### 3.2 To the method in Indra’s problem

#### 3.2.1 Use of Divisive Clustering, its advantage over Agglomerative Clustering

Indra had used Agglomerative Clustering to form maximally separated subsets of data. Agglomerative clustering initializes a subset with a single point, and adds a point in each iteration. The point to be added is chosen greedily so as to maximise the distance between the subset and the dataset. The iterations are continued till the subset has half the total number of points in the dataset. By removing the generated subset from the complete dataset, we obtain the other subset.

However, this procedure is prone to small-sample errors. In the beginning stages of the iteration, the subset is very small in size. The distance between the distributions of dataset and the small subset is not reliable, and may result in wrong points added to the subset. To tackle this issue, we changed the clustering algorithm to Divisive Clustering.
In Divisive Clustering, the subset is initialized to the entire dataset. In every iteration, one point is removed. The point to be removed is chosen greedily so as to maximise the distance between the generated subset and the dataset. The iterations are continued till the generated subset is half the size of the dataset. By removing the generated subset from the complete dataset, we obtain the other subset.

In the Divisive Clustering procedure, the sets being compared are not small. The size of the generated subset is at least half the size of the dataset.

### 3.2.2 Using patches of datapoints instead of individual datapoints

The Divisive clustering algorithm alleviates the small-sample error, but is computationally expensive. This is because at each iteration, it checks the distance for all the points that can be removed from the subset.

So, we divided the data into several smaller patches. At each step in the Divisive Clustering, instead of removing only one point, we remove an entire patch. The success of the clustering depends heavily on properly assigning patches.

While we do the splitting procedure, we have to be careful to perform exactly the same operation on X and Y. If we do different operations on X and Y, we will be introducing asymmetry in the data, and bias the result. We first tried to make patches by splitting the grid of the data equally into 4 parts, picking the non empty patches, and splitting them into another 4, and so on. By doing the splitting in this manner, although all patches had the same area, some of the patches had a large number of data points, and some had very few data points.
3.2.3 Splitting the dataset into small patches by using trees

We realized that for some datasets, the data is extremely dense in some regions, and somewhat sparse in the others. Thus, evenly splitting the dataset by area was giving patches with very different number of points. So, we used a tree structure to split the dataset into patches, and applied Divisive Clustering on the patches.

We split the dataset in a way such that the number of samples in both the subsets does not differ by more than 10% of the total size of the dataset. This is to ensure that the finite sample errors don’t influence the decision. The splitting is done adaptively, so that at least 2000 subset pairs with comparable size and generated.

Then we iterate through all these subset pairs, and find the maximally distant subsets.

3.2.4 The assumption that turned out to be incorrect

Our work was based on the assumption that if we select arbitrary subsets from the dataset, the conditional distribution will remain the same. This assumption had been made because the causal mechanism generating the samples is the same for both the subsets. However, we missed the point that if we are going to be choosing subsets which are maximally separated, there is no guarantee that the conditional distribution will be the same. The correct assumption is,

"If $X \to Y$, the distribution of $X$ and the function $f$ mapping $X$ to $Y$ are independent since they correspond to different mechanisms of nature."
However, during the selection of maximally separated subsets, we are choosing the marginal of $X$ in a subset, after observing the values of $Y$. Thus, there is no guarantee that the distribution of $X$ and the function $f$ are independent.

### 3.3 To Active Learning Problem

In our research, we have leveraged the UCB algorithm to tackle the Active Learning problem by posing the Active Learning problem as a bandit problem.

#### 3.3.1 Parallels between UCB and Active Learning

For applying the UCB algorithm to the Active Learning problem, we consider the following:

- **Arm**: We consider the intervention to perform as an arm of the bandit.
- **Reward**: We consider the following as the reward for the intervention $i$ chosen by the agent:

$$\frac{\mathcal{R}(\mathcal{D}) - \mathcal{R}(\mathcal{D} \cup \{(i, x)\})}{c(i)}$$

where $\mathcal{R}$ is the risk, $x$ is the sample obtained under the intervention $i$, and $c(i)$ is the cost associated with the intervention $i$. The assumption that $c(i) > 0 \forall i \in \mathcal{I}$ is crucial here.
Discretizing the space of Interventions

The multi-armed bandit problem considers only a finite number of arms or actions that are available to the agent. In the active learning problem, however, the space of interventions could possibly be infinite.

For applying the UCB algorithm to the active learning problem, the space of interventions in the active learning problem has to be discretized.

The choice of the reward function

In the active learning problem, the objective is to reduce the risk function, without incurring much cost. In the bandit problem, the reward is to be maximized.

The reward function that we have chosen compares the risk before the intervention with the risk after we the intervention. This is one way of determining the usefulness of a given intervention.

If multiple interventions have the same cost, the expression for the reward evaluates to a higher number in the case of the intervention that provides greater reduction in the risk.

Maximizing the cumulative reward over time steps is equivalent to reducing the total risk.
Chapter 4

Results

4.1 Modified CGNN

4.1.1 Ability of NoiseMMD to identify the Normal Distribution

The purpose of the NoiseMMD term in our objective is to enforce normally distributed noise parameters.

This experiment was performed to see if the NoiseMMD term is able to distinguish between distributions.

We calculated the NoiseMMD of samples drawn from Standard Normal, Uniform(0,1), Exponential(1) and Cauchy distribution. The sample size taken as 109. This is the smallest dataset in the Tuebingen Cause-Effect Paris.

The results are summarized in table 4.1.1.
<table>
<thead>
<tr>
<th>Distribution</th>
<th>NoiseMMD</th>
</tr>
</thead>
<tbody>
<tr>
<td>Normal(0,1)</td>
<td>0.0001</td>
</tr>
<tr>
<td>Uniform(0,1)</td>
<td>0.33</td>
</tr>
<tr>
<td>Exponential(1)</td>
<td>0.35</td>
</tr>
<tr>
<td>Cauchy</td>
<td>0.12</td>
</tr>
</tbody>
</table>

Table 4.1: Ability of NoiseMMD to identify the Normal distribution

<table>
<thead>
<tr>
<th>Training Objective</th>
<th>Scoring</th>
<th>MMD Only</th>
<th>Noise MMD</th>
<th>Our Objective</th>
<th>Abs Square Error</th>
</tr>
</thead>
<tbody>
<tr>
<td>Our Objective</td>
<td>0.03</td>
<td>0.05</td>
<td>0.075</td>
<td>0.07</td>
<td></td>
</tr>
<tr>
<td>MMD Objective</td>
<td>0.05</td>
<td>0.02</td>
<td>0.07</td>
<td>0.03</td>
<td></td>
</tr>
</tbody>
</table>

Table 4.2: Values of Errors for training with different Objectives

We see that the samples drawn from the normal distribution have a very small value of NoiseMMD. From the table we can see that for the Uniform and Exponential distributions, the NoiseMMD value is fairly high as compared to that of the standard normal distribution.

The Cauchy distribution is also bell-shaped. Its NoiseMMD value is low as compared to Uniform and Exponential distributions. But it is still much higher than that of the standard normal distribution.

### 4.1.2 Testing Goodness of Fit

We wanted to check if our model is actually fitting the functions well. We trained the network using our objective, and evaluated it with various metrics. We then trained the network using MMD Objective used in [3], and evaluated using the same metrics. The results are summarized in Table 4.1.2.

We can see from the Table that our model is able to fit the functions well.
Table 4.3: Generalization Results on Seen and Unseen Data

<table>
<thead>
<tr>
<th></th>
<th>MMD Unseen</th>
<th>Sq. Error Unseen</th>
<th>MMD Seen</th>
</tr>
</thead>
<tbody>
<tr>
<td>MMD Objective</td>
<td>0.2</td>
<td>0.4</td>
<td></td>
</tr>
<tr>
<td>Our Objective</td>
<td>0.8</td>
<td>0.4 to 0.8</td>
<td>0.1</td>
</tr>
</tbody>
</table>

The errors are close to those achieved by training with the MMD Objective.

4.1.3 Test for Generalization

From the last experiment we know that our model fits the function reasonably well. We performed this experiment to see if our model can generalize well.

We trained a neural network using our objective. We then evaluated the network using the seen data, but with different, randomly chosen noise parameters. The network was then evaluated on unseen data. Since the noise parameters were not present for unseen data, we chose samples out of a standard normal distribution.

We repeated these procedures for MMD Objective.

The results are summarized in 4.1.3.

4.1.4 Accuracy of Causal Inference

We tried the model with various values of hyperparameters like the number of nodes in the hidden layer, and the weights in our objective function.

The best weighted accuracy of causal detection that we achieved was 59%. This was achieved by using weights of 9,1,9 for MMD,NoiseMMD and HSIC term, respectively.
4.2 Active Learning Experiments and Results

Here we describe the experiments we performed in order to compare our algorithms with baseline algorithms.

4.2.1 Algorithms Compared

We have performed experiments with the following algorithms:

1. The Observe-only Algorithm:
   This algorithm always chooses to do the empty intervention.
   All the samples that this algorithm obtains are drawn from the observational distribution.

2. The Random Intervention Algorithm:
   From the set of all possible interventions (including the empty intervention), this algorithm randomly picks one intervention at each time step.
   The random intervention algorithm gets the advantage of drawing samples from the observational distribution as well as intervened distributions. Because of the interventions it performs, it gets information about the functions in parts of the domain that the Observe-only algorithm possibly does not get.

3. The UCB Algorithm:
   This algorithm selects the intervention to be performed using the UCB algorithm described in the subsequent sections.
   This algorithm chooses the action on the basis of the data observed so far, with an aim to maximize the reward obtained.
4.2.2 FCMs Used

We tested each of the above algorithms on two FCMs. We describe these SCMs below.

FCM 1

- Graph: The causal graph of this FCM is shown in Figure 4.1.
- Equations: The set of equations in this FCM is:

\[
\begin{align*}
X_0 &= 0 + E_0 \\
X_1 &= \sin(2X_0) + E_1 \\
X_2 &= \cos(X_1^2) + E_2 \\
X_3 &= \cos(X_2) + E_3 \\
X_4 &= \cos(X_3^2) + E_4
\end{align*}
\]

- Distribution over Exogenous Variables: The exogenous random variables \(E_1, \ldots, E_4\) are independent. Each of them is normally distributed, with a mean of zero, a variance of 0.1.
- Interventions Allowed: We allow interventions that act on only one variable at a time.
The intervened variable can be forced to any value in the following set,

\[ \{-3 + \frac{2k}{3} : k \in \mathbb{Z}, 0 \leq k \leq 9\} \]

The empty intervention is also a part of the intervention set.

The costs \( c(i) \) are all assumed to be equal.

**FCM 2**

- **Graph:** The causal graph of this FCM is shown in Figure 4.2.

- **Equations:** The set of equations in this FCM is:

\[
X_0 = 0 + E_0 \\
X_1 = -1 + E_1 \\
X_2 = \cos(X_0 + X_1) + E_2 \\
X_3 = \cos(X_2) + \sin(X_1) + E_3 \\
X_4 = \sin(X_2X_3) + E_4
\]
• Distribution over Exogenous Variables: The exogenous random variables $E_1, \ldots, E_4$ are independent. Each of them is normally distributed, with a mean of zero, with a variance of 0.1.

• Interventions Allowed: We allow interventions that act on only one variable at a time. The intervened variable can be forced to any value in the following set,

$$\left\{-3 + \frac{2k}{3} : k \in \mathbb{Z}, 0 \leq k \leq 9\right\}$$

The empty intervention is also a part of the intervention set. The costs $c(i)$ are all assumed to be equal.
4.2.3 Experiment 1

This experiment was performed on FCM 1. Each of the three algorithms was allowed to act on the SCM for 50 time steps.

- Radial Basis Functions with bandwidth parameter 1 were used as kernels for the Gaussian Process Regression.
- At each time step, the algorithm’s prediction of each of the functions is compared to the true function.
- The region of interest is taken as [-3,3]. It is discretized into 50 points.
- Equal weight is given to each of these points.
- The parameters $\alpha_n$ which are the weights of the errors of each function, are taken to be 1.
- For each of the nodes, the Mean Square Error between the algorithm’s prediction of the function and the true function is calculated in this region.

The graph of the error is shown in the Figure 4.3.

The true functions at each node are plotted. The estimates of these functions by each of the algorithms is also plotted. These plots are shown in Figures 4.4, 4.5, 4.6 and 4.7.
4.2.4 Experiment 2

This experiment was also performed on FCM 1. The kernel, the region of interest, the and the number of time steps used, were the same as in Experiment 1.

In Experiment 1 we saw the Mean Squared Errors for each of the functions for each of the algorithms.

In this Experiment, we perform the Mean Squared Error test multiple times for each agent. For example, we allow the Random Intervention algorithm to run on the FCM for 50 time steps, noting the true Mean Square Error at each time step. Then, we run the algorithm fresh on the SCM, for 50 time steps, again noting the true Mean Square Errors at each time step. We repeat this process 10 times.

Each of these errors is plotted in pale blue in the figure. Then, we average the Mean Square Error over these iterations at each time step. This average is plotted as the thick blue line in the figure.

The same process is repeated for the Observing algorithm and the UCB algorithm.

These plots are shown in Figure 4.8.

This experiment gives results about the average performances of the algorithms. From the graph, we can see both the consistency and accuracy of each algorithm.
4.2.5 Experiment 3

This experiment was performed on FCM 2. Each of the three algorithms acted on the SCM for 50 time steps. Unlike SCM 1, which has only 1-D functions, SCM 2 has 2-D functions.

- Radial Basis Functions with bandwidth parameter 1 were used as kernels for Gaussian Process Regression.

- At each time step, the algorithm’s estimate of the functions is compared with the true functions.

- For 2-D functions, the region of interest is the square with vertices (-3,-3),(-3,3),(3,-3), and (3,3). Both the axes were discretized into 50 points each, resulting in 2500 points in the region.

- Equal weight is given to each of these points.

- The parameters $\alpha_n$ which are the weights of the errors of each function, are taken to be 1.

- For each of the nodes, the Mean Square Error between the algorithm’s prediction of the function and the true function is calculated in this region.

The graph of the error is shown in the Figure 4.3.
Figure 4.3: Experiment 1: Mean Square Error of estimates at each time step.
Figure 4.4: Experiment 1: True function $f_1$ and its estimates by algorithms.
Figure 4.5: Experiment 1: True function $f_2$ and its estimates by algorithms.
Figure 4.6: Experiment 1: True function $f_3$ and its estimates by algorithms.
Figure 4.7: Experiment 1: True function $f_4$ and its estimates by algorithms.
Figure 4.8: Experiment 2: Mean Square Errors for multiple instances of algorithms.
Pale lines show errors of individual instances. Bold lines show the average error for the algorithms.
Figure 4.9: Experiment 3: Mean Square Errors for algorithms on FCM 2.
4.2.6 Modified Sampling Method

In the sampling algorithm used by Rubenstein et. al. [5], they have used only the estimated risk to decide on the intervention.

However, the estimated risk itself is based on the estimate of the samples drawn.

It is possible that a useful intervention gets a high estimated risk, and other interventions get lower risks.

To select the most useful intervention, we take the following approach. We calculate the estimated risk $R_{est}$, as well as the estimated standard deviation of the risk $R_{SD}$. We pick the intervention which yields the minimum value of $R_{est} - R_{SD}$.

This is a more optimistic choice. The chosen intervention is the one that plausibly has the potential to provide the lowest risk.
4.2.7 Experiment on Modified Sampling Method

This experiment was performed on FCM 1. Each of the four algorithms was allowed to act on the SCM for 50 time steps.

- Radial Basis Functions with bandwidth parameter 1 were used as kernels for the Gaussian Process Regression.

- At each time step, the algorithm’s prediction of each of the functions is compared to the true function.

- The region of interest is taken as [-3,3]. It is discretized into 50 points.

- Equal weight is given to each of these points.

- The parameters $\alpha_n$ which are the weights of the errors of each function, are taken to be 1.

- For each of the nodes, the Mean Square Error between the algorithm’s prediction of the function and the true function is calculated in this region.

The graph of the error is shown in the Figure 4.10.

The true functions at each node are plotted. The estimates of these functions by each of the algorithms is also plotted. These plots are shown in Figures 4.11,4.12,4.13 and 4.14.
Figure 4.10: Experiment 1: Mean Square Error of estimates at each time step.
Figure 4.11: Experiment 1: True function $f_1$ and its estimates by algorithms.
Figure 4.12: Experiment 1: True function $f_2$ and its estimates by algorithms.
Figure 4.13: Experiment 1: True function $f_3$ and its estimates by algorithms.
Figure 4.14: Experiment 1: True function $f_4$ and its estimates by algorithms.
Chapter 5

Conclusions

We have studied and addressed two problems in the context of FCMs: Causal Inference and Function Estimation.

We explored two approaches for Causal Inference. One was based on Generative Neural Networks, where we built upon the framework used in [3]. We used noise values as parameters, and generated output samples using them. The objective function was modified suitably to properly estimate the noise values, and to make sure that they follow certain properties that noise is expected to follow.

The other approach was based on finding maximally distant subsets of the dataset, building upon the work done by [4]. However, this approach was based on an assumption that was later found to be incorrect.

For function estimation, we applied the UCB algorithm to the active learning problem. We also compared the performance of this algorithm with the Observe-only algorithm and with the Random intervention algorithm.
We learnt that observing the system without interventions is not very helpful for learning the functions. We also learnt that Randomly selected interventions do better. However, they are not very consistent. The UCB algorithm consistently performs better than the Observe-only algorithm and the Random intervention algorithm.

We learnt that Gaussian Regression can be used to learn fairly non-trivial functions. Our SCMs had functions which involved sines and cosines, and also 2-D functions in SCM 2 had which were trigonometric functions of the products of the inputs. These functions would not be learned well using a linear regressor.
Chapter 6

Future Work

6.1 Connection of Active Learning problem to Bandits

For future work, we will look into the application of sampling from the intervened distributions, as done in [5]. We will, however, take a different number of samples from the interventional distribution of each of the possible interventions. The number of samples taken from the interventional distribution will depend on the lower confidence bound.

This approach will be more robust than the approach that we have used. This is because the estimate of the lower confidence bound will be more accurate. Also, this approach takes into account the changing of data at each step. Thus, this approach will be better at tracking changes in the utility of each intervention.

We intend to further study the Modified Sampling Method. We will test
our hypothesis that the Modified Sampling Method will perform better on FCMs with a large set of interventions, but only a small number of samples being allowed.

We hope to find better results by making these changes as well as other adjustments to the algorithm.

6.2 Different statistics that may be used in the CGNN training objective

There are multiple ways in which we can compare the distribution of noise parameters against the standard normal distribution. Using the MMD in the objective is one of them. There are other functions that may be used to test the Gaussianity of the noise parameters. The test in [1] is a one such criterion. It is framed as a test for the hypothesis that the noise parameters are drawn from a standard normal distribution. This approach also uses the MMD, but the level of the test can be adjusted by varying the threshold for the rejection region.
Bibliography


