Causal Reasoning and Meta Learning using Kernel Mean Embeddings

Jean-François Ton

St Peter's College

University of Oxford

A thesis submitted for the degree of

Doctor of Philosophy

Hilary 2022

Acknowledgements

First and foremost I would like to sincerely thank my amazing supervisors Prof. Dino Sejdinovic and Prof. Yee Whye Teh for their wisdom, patience and guidance throughout my DPhil, without which I would not have been able to complete this thesis. Their continuous encouragement and care have helped me through many rough times and it is thanks to their trust in me that I am where I am today. There were times in the beginning, where I thought of not progressing with the DPhil, but because both my advisors took the time to talk to me, I was able to persist. I am extremely grateful to have had the privilege to work with them and could not have asked for more.

Secondly, I would like to thank the people I have met along my journey while pursuing my DPhil here in Oxford. There have been so many incredible people that I have met that I cannot possibly mention everyone. In no particular order:

Thanks to the Oxford Table Tennis Club, which is where I spend most of my free time while being here. You have been an essential part of my experience and I have met most of my good friends through this club. There are too many people that I have met to enlist them all, so here are some. To Abe, Leon, Dai, Eric, Kritica, Emi, Julia, Jerry, Zhongyi, Coach Philip, Kin, Jason, Seb, Seb2, Klaus, Tianze, Pan, Taylor, Ali, Sam, Aman and the list goes on. You have really made it special for me to come to TT and take my mind off work. All the games we have won and lost together will always stay in my mind. Special thanks to the presidents, Louise, Kritica, Angus, Eric and Seb for making the club so welcoming.

I would also like to thank the people in the statistics department and in particular my office 1.17. To eMilIeN dUpoNT, Jim Aaron Xu, Bobby hUGe, Faaiz Muhammad Stanley, Robert Meme Hu, Charline, Fran, Sheh, Eduard, Carlo, Desi, Andrew. Coming to the office everyday was a delight and something to really look forward to. The coffee breaks, the chats and park walks at 4pm really helped me not go insane. In addition to the office, there have been other amazing people I have met in the department. To Fan, Leon, Michael, Kaspar, Antony, Soufiane, Rob, Alan, Lucian, Edwin, Jake, Tyler, Veit, Shahine, David, Bohao, Yuan, Lisa, Stacey, Adam F, Adam K, Adam G, Tomas, Francesca, Deborah, Jessie, Dom, Ed and many more. Thanks a lot for making the

department feel like home. Special thanks to Prof. Arnaud Doucet, who has also been a great mentor and who I have been lucky enough to collaborate with many times. Also thanks to all my collaborators I have had the privilege to work with throughout the years. I truly believe that I could not have chosen a better department for my DPhil.

Next, I would also like to acknowledge the several group chats: “Fun dis game”, “Incredibles” and the “Simple Lunch group” that have given me motivation, made me laugh and made me think about life. Cheers to all the late night chats, league games and gatherings. If you are reading this, you know which group chats I am talking about and I just want to express my appreciation for you all. Special thanks to my partner Lulu, for her continuous love, trust and encouragements.

Lastly, I want to express my sincerest gratitude to my parents. To mom: You have given me everything I needed so that I was able to grow into the person I am today. You have sacrificed much of your time and patience and I will be forever grateful for your parenting. Over the years I have realised how lucky I am to have a mother like you and am sorry I was oblivious to your love for so long. I hope I can do better in the future. To dad: Thank you very much for all the lessons you taught me and your continuous support throughout my studies. You have always been my number one fan and always will be. You have encouraged me to pursue table tennis, which is now a pillar in my life and I will never forget the car drives we had to the tournaments every Sunday. Hope you are proud of me and will watch over me, wherever you are now.

Statement of Originality

I hereby declare that except where specific reference is made to the work of others, the content of this thesis is my own work and has not been submitted in whole or in parts for any other degree or qualification. This thesis is my own work unless otherwise stated in the authorship form at the end of the chapters.

Jean-Francois Ton
Hillary 2022

I dedicate this thesis to my parents.
To mom, for her unconditional love and to dad, I hope you will always watch over me.

Contents

List of Figures

Abstract

Kernel methods have been an essential instrument in machine learning over the years due to their ability to map data into high dimensional spaces efficiently. Nowadays, especially in machine learning, multimodality in the data is a common phenomenon and hence developing tools that allow us to capture these structures is crucial. Being able to reliably represent these datasets without many distributional assumptions is at the core of the main methodology used in this thesis i.e. Kernel Mean Embeddings. This search for good representations of the datasets become even more challenging when we are only presented with little data. There are two particular areas in machine learning that this thesis aims to contribute to using the kernel mean embeddings, which are 1) Meta learning and 2) Causality.

Meta learning has become a widely popular area of research in machine learning due to its ability to leverage statistical dependencies between similar datasets also known as tasks. By taking advantage of task similarities, meta learning is able to learn and adapt quickly to new unseen problems. This can have many applications, where the data collection or labelling is expensive, such as in computational chemistry or health care. In this thesis, we will tackle limitations of current meta learning methods to multimodal data with the help of kernel mean embeddings and proposed efficient algorithms that do not make explicit assumptions on the distribution of the data. In chapter 2 we consider the problem of conditional density estimation and proposed MetaCDE. MetaCDE is able to accurately determine the density even in multimodal setting, whereas standard methods fail. Furthermore, in chapter 3, we also propose MetaCGNN for causal discovery. MetaCGNN is a meta learning algorithm which leverages the power of kernel mean embeddings in the little data setting and allows us to determine the causal direction in the bivariate case, i.e. determining whether $X \to Y$ or vice versa. Here again, we show that we are able to significantly improve upon existing methods.

This leads us to the problem of causality. Causality is a fundamental problem in machine learning, as it allows us to reason about the cause and effects of our actions. Having dived into the realm of causality through causal discovery in chapter 3, chapter 4 will be tackling a problem on the causal inference with the help of kernel

mean embeddings. Specifically, we introduce BayesIMP, and we demonstrate how to draw causal conclusions, while also accounting for the uncertainty in the data and the model. We achieve this by combining several literature on kernel methods, Gaussian processes and Causal inference.

This thesis proposes several algorithms that aim to tackle some core issues in machine learning with the help of kernel mean embeddings. Given that the problem of small datasets and lack of causal awareness is likely to remain a concern in the area, we believe that this work can be a guide for future research and development.

1 Introduction

This thesis follows the format of an integrated thesis and is composed of 5 chapters. The first and the last chapter consist of an introduction and conclusion respectively. The remaining 3 chapters are formed of separately published conference papers, which each contain their own introduction, background section, and literature review. In this introduction, we would like to give a more general overview/motivation of the work done throughout the DPhil, as well as the relevant background needed.

1.1 Motivation and Contributions

In recent years, machine learning research has gained tremendous popularity due to the abundant data and computational resources available [Harris et al., 2019, Chen and Lin, 2014, Najafabadi et al., 2015]. It is thanks to the combination of increased accessibility to data, advances in hardware (i.e. GPU and CPU) and introductions of software frameworks such as Pytorch and TensorFlow [Paszke et al., 2017, Abadi et al., 2016] that we are able to see a surge of machine learning applications in many fields of study. In particular, healthcare, logistics and finance [Carbonneau et al., 2008, Dixon et al., 2020, Ghoddusi et al., 2019, Rundo et al., 2019, Beam and Kohane, 2018, Shailaja et al., 2018] have started to leverage machine learning to further improve or even automate tasks, that have previously been done by humans.

One of the key vehicles that has enabled these rapid advancements is *Deep Learning* [LeCun et al., 2015]. In essence, the simplest Deep Learning model, Feed Forward Neural Networks (FFNNs) [LeCun et al., 2015, Minsky, 1961], consist of stacking and linking multiple Perceptron units on top of each other. By noting that the parameters of each of these Perceptrons can be efficiently learnt through repeated use of the chain rule, also known as backpropagation, FFNNs are able to capture very complex patterns hidden in the data that can then be leveraged for prediction.

However, there are two key problems we draw the reader's attention to when it comes to machine learning: 1) the need for big datasets and 2) correlation-based inference instead of causal based inference.

In this thesis, we will demonstrate why it is paramount to be able to represent the distribution of a dataset in a statistically rigorous as well as sample efficient manner. Assume we are tasked to represent a mixture of Gaussians where we have two modes, one at -1 and one at 1. In this case capturing only the mean and variance as is done in many algorithms that use a parametric model and assume e.g. plain gaussianity, would be misleading. This is where kernel mean embeddings come into play. Kernel mean embeddings are a powerful framework that allows us to represent the whole probability distributions in a well-defined function space known as a Reproducing Kernel Hilbert Space (RKHS) without making any assumptions about the data. In the following, we will describe how informative representations of the data can help in both small data settings and in the setting of causal inference.

Small data problem: Firstly, an important aspect when using Deep Learning models is the quantity of data itself. In many applications, we are provided with an abundance of data such as in logistics or in finance where thousands if not millions of data points are recorded every hour [Heaton et al., 2017]. In these cases, neural networks excel as they are able to extract the underlying key features of the data, which in turn allows them to have high precision when used for downstream tasks. Nevertheless, having access to copious amounts of data is not always the case. As an example, in health care we usually do not have many data points for a given patient and hence training a Neural Network on only few data points is problematic [Johnson and Khoshgoftaar, 2019, Miotto et al., 2018]. Furthermore, in computational chemistry, there are cases where only little data is available as synthesizing drugs can be very expensive and time-consuming. Hence, using neural networks in many of these cases can be challenging. In spite of that, all is not lost. There are many cases we have a

little dataset for a given drug, but we have a lot of these small datasets corresponding to different drugs. The goal is to leverage the statistical dependencies between datasets, in order to be able to generalize well on a new given drug. This is where meta learning can help [Vilalta and Drissi, 2002, Vanschoren, 2018].

In chapter 2, we tackle the problem of meta learning for conditional density estimation with applications to computational chemistry. In this application, various symmetrical properties around the molecules can cause multimodality in the data. Specifically, in density estimation, it is important to be able to capture the full density rather than just the first two moments as mentioned previously. This becomes even more challenging when we are presented with little data. Hence, we proposed an algorithm MetaCDE, which allows us to first represent the given small dataset using kernel mean embeddings, which is subsequently used to perform conditional density estimation. Being able to represent the small dataset well is the key to many meta learning problems, and we show how kernel mean embeddings can be incorporated to do so in a simple and efficient manner.

In chapter 3, we then proposed a similar algorithm, but this time for causal discovery. Causal discovery is the task of determining the causal graph based on pre-collected data also known as observational data. One specific challenge is that of determining the causal direction of each edge, i.e. determining whether A caused B or vice versa [Glymour et al., 2019, Spirtes and Zhang, 2016, Peters et al., 2017]. Hence, in this chapter, we focus particularly on determining the causal direction in the bivariate case. This seemingly simple bivariate case has been widely studied [Mitrovic et al., 2018, Hoyer et al., 2008, Zhang and Hyvarinen, 2012, Ton et al., 2021c, Shimizu et al., 2006], and has also seen success in determining the causal direction given. However, in many cases, we are given large datasets which might not always be the case due to the expensive data collection processes and hence meta learning can be applied. We propose MetaCGNN [Ton et al., 2021c], which is an algorithm that, similarly to MetaCDE, represents the small dataset using kernel mean embeddings, which can then be used to construct a generative model that allows us to determine the correct causal direction.

Correlation based inference: The second shortcoming of standard deep learning methods is that of correlation-based inference instead of causality based inference [Peters et al., 2017]. Let us take as an example the following Causal graph in Figure1.1, where X is the plane price ticket, Y is the profit and Z is the demand. If we were to simply use a standard model without accounting for Z we might come to the conclusion

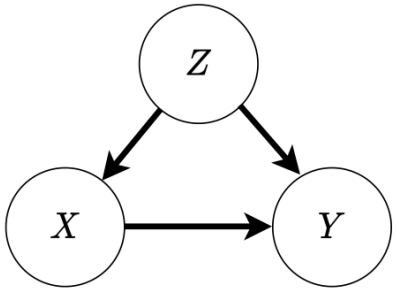

Figure 1.1: Causal Graph with a confounding variable Z.

that by increasing the price X we also increase the profit Y, as there is a positive correlation between price and profit. Hence the conclusion would be to increase the price of tickets as much as possible if the goal is to maximize profit. This is however a misleading conclusion as this model did not take into account the confounding variable: demand Z. Causal inference allows us to account for these variables efficiently by solving various density estimation problems (assuming all variables are known). We will elaborate more on this in the background section. Here again we note that kernel mean embeddings can play an important role by instead representing the distribution in a RKHS and performing the necessary operations there instead of having to perform density estimation in the original data space.

In chapter 4, we develop BayesIMP which allows us to perform causal inference using kernel mean embeddings. This allows us to circumvent the need for density estimation, which can oftentimes be a difficult task, as we will demonstrate. Furthermore, we also develop an efficient way to quantify the uncertainty due to insufficient data or an increase in noise using Gaussian Processes [Rasmussen and Williams, 2005b].

In summary, the contributions of this thesis can be summarized as follows:

- We proposed a novel way to perform meta learning for conditional density estimation using kernel mean embeddings. The main idea is to capture the conditional density using kernel mean embeddings and then use noise contrastive estimation to output the final conditional density. We use a meta learning training scheme to apply our proposed method, MetaCDE, to computational chemistry problems where data is scarce.

- Next we propose a meta learning algorithm, which again leverages kernel mean embeddings to capture the conditional density. In this case, instead of using

the kernel mean embeddings for density estimation, we utilize the information stored for causal direction detection. We experimentally verify that our proposed methods MetaCGNN is able to determine the causal direction successfully and is in addition allowing significant speedups of the previously proposed algorithm Causal Generative Neural Network (CGNN) [Goudet et al., 2018] significantly.

- Lastly, inspired by the previous project that focused on causal discovery, we dive deeper into the realm of causality, this time investigating causal reasoning more broadly. By assuming we have access to the causal graph, we are able to draw causal conclusions from observational data. In particular, chapter 4 focuses on Causal Bayesian Optimization which is a special type of sequential optimization that leverages causal graphs at each iteration. More specifically, we introduce the use of kernel mean embeddings for causal reasoning while also accounting for uncertainty when estimating the causal effect.

1.2 Background

Throughout this thesis, the use of kernel mean embeddings will be ubiquitous and is hence introduced formally in this background section. In particular, we start with the basics of kernel methods and move on to kernel mean embeddings once the foundation has been set.

We will then give an overview of causality, in particular causal discovery and causal inference, which are treated in Chapters 3 and 4 respectively.

Lastly, we describe the current developments and algorithms in meta learning and in particular how kernel methods can play an important role in it.

1.2.1 Kernel Methods

Kernel methods are a classical machine learning technique [Shawe-Taylor et al., 2004] and have played a crucial role in setting the foundation for many machine learning models to date. We will start off by defining the kernel function which is going to subsequently define a Reproducing Kernel Hilbert Space (RKHS). The below paragraphs were inspired by and follow the exposition of the lectures notes of Gretton [2019] and Sejdinovic [2019] as well as [Berlinet and Thomas-Agnan, 2011]

Definition 1. *Let $\mathcal{X}$ be a non-empty set. A function $k : \mathcal{X} \times \mathcal{X} \to \mathbb{R}$ is called a kernel if there exists a Hilbert space and a map $\phi : \mathcal{X} \to \mathbb{R}$ such that $x, x' \in \mathcal{X}$,*

$$k(x, x') := \langle \phi(x), \phi(x') \rangle_{\mathcal{H}}$$

Now that we have defined a kernel we can define the corresponding RKHS associated to that kernel.

Definition 2. *(Reproducing kernel Hilbert space) Let $\mathcal{H}$ be a Hilbert space of $\mathbb{R}$-valued functions defined on a non-empty set $\mathcal{X}$. A function $k : \mathcal{X} \times \mathcal{X} \to \mathbb{R}$ is called a reproducing kernel of $\mathcal{H}$ and $\mathcal{H}$ is a reproducing kernel Hilbert space, if k satisfies:*

- $\forall x \in \mathcal{X}$, $k(\cdot, x) \in \mathcal{H}$
- $\forall x \in \mathcal{X}, \forall f \in \mathcal{H}, \langle f, k(\cdot, x) \rangle_{\mathcal{H}} = f(x)$ *(reproducing property)*

In particular, it follows that $\forall x, y \in \mathcal{X}$,

$$k(x, y) = \langle k(\cdot, x), k(\cdot, y) \rangle_{\mathcal{H}}$$

The use of kernel methods became particularly popular as many algorithms such as Linear regression, Principal Component Analysis, K-means clustering etc only rely on the inner product $x^T x$ of the data points. This is a crucial observation, because for a given kernel, we implicitly defined a corresponding feature map. Hence intuitively, if an algorithm solely relies on the inner product, we are indirectly first applying a mapping to the datapoint using $\phi \in \mathcal{H}$ before using the above mentioned algorithm, without ever having to evaluate ϕ

As an example, one of the most popular kernels used in kernel methods is the Radial Basis Function kernel (RBF),

$$k(x, y) = \exp\left(\frac{-||x - y||^2}{2\sigma^2}\right) \tag{1.1}$$

whose corresponding feature map ϕ is infinite-dimensional and can be derived using a series expansion. Hence, by noting that many algorithms only require the inner product of the data points, it follows that methods such as linear regression, PCA etc can easily be "kernelized". Implicitly, this corresponds to mapping the original data x to $\phi(x)$ first and then applying the algorithm on $\phi(x)$ without ever having to evaluate $\phi(x)$.

In this thesis, contrary to standard kernel methods, we will make use of neural networks as feature maps. We argue that even though, theoretically kernels such as RBFs operate in infinite dimensions, by learning feature maps through neural networks we are able to provide a flexible way to capture important patterns in the data.

Now that we have established the foundation and motivation for kernel methods we can move on to kernel mean embeddings which is the leading theme in this thesis.

1.2.1.1 Kernel Mean Embeddings

Kernel mean embeddings (KME) have been widely used in many applications such as two-sample testing, reinforcement learning, meta learning, explainable machine learning etc. [Lever et al., 2016, Gretton et al., 2012, Muandet et al., 2017, Chau et al., 2021b]. For a thorough review on KME we refer the reader to Muandet et al. [2017] whose notation and exposition we follow here. Intuitively, we map probability distributions into an RKHS which we can then use to perform various operations that were not possible or very hard in $\mathcal{X}$. Let us define the KME $\mu_{\mathbb{P}}$ for distribution $\mathbb{P}$ as follows:

Definition 3. *[Berlinet and Thomas-Agnan [2011], Smola et al. [2007], Muandet et al. [2017]] The kernel mean embedding of probability measures in $M_+^1(\mathcal{X})$ into an RKHS $\mathcal{H}$ endowed with a reproducing kernel $k : \mathcal{X} \times \mathcal{X} \to \mathbb{R}$ is defined by mapping:*

$$\mu : M_+^1(\mathcal{X}) \to \mathcal{H}, \mathbb{P} \to \int k(x, \cdot) d\mathbb{P}(x)$$

where we use $M_+^1(\mathcal{X})$ to denote the space of probability measures over a measurable space $\mathcal{X}$

Empirically for a given sample $\{x_i\}_i^N \sim \mathbb{P}$, the associated KME can be estimated as:

$$\widehat{\mu_{\mathbb{P}}} = \frac{1}{N} \sum_{i=1}^{N} k(x_i, \cdot) \tag{1.2}$$

A pictorial illustration of KME can be seen in Figure1.2. As mentioned previously, one of the key applications of KME is that of two-sample testing, where given a sample $\{x_i\}_i^N \sim \mathbb{P}$ and $\{y_i\}_i^N \sim \mathbb{Q}$, [Gretton et al., 2012] proposed a way to determine whether $\mathbb{P} = \mathbb{Q}$. Other applications connected to neural networks that have made use of KME are MMD GANs [Bińkowski et al., 2018] and DeepSets [Zaheer et al., 2017]. MMD GANs in essence use the two-sample test proposed by [Gretton et al., 2012] as a drop-in replacement for the discriminator in standard GANs [Goodfellow et al.,

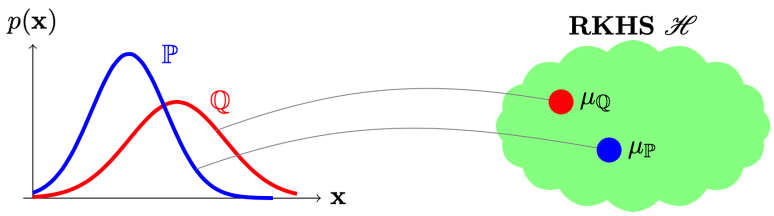

Figure 1.2: Illustration of kernel mean embeddings were taken from Muandet et al. [2017]. The figure shows how we embed marginal distributions into an RKHS using Def. 3

2014]. DeepSets also borrows ideas from KME, by essentially using neural networks as feature maps and then defining the representation of a set of elements $\{z_i\}_{i=1}^N$ as $r = \sum_i \phi^{NN}(z_i)$. These representations r can then for example be used in downstream tasks and have extensively been used in meta learning [Garnelo et al., 2018a,b, Kim et al., 2019]. However, KMEs only capture marginal distributions into the RKHS. In the next section, we will define the Conditional Mean Embeddings, which, as the name suggests, will allow us to represent conditional distributions in pre-defined RKHSs.

1.2.1.2 Conditional Mean Embedding

Similarly to kernel mean embeddings which aim to map marginal distributions into an RKHS, conditional mean embeddings (CMEs) aim to map conditional distributions into an RKHS. Contrary to the standard kernel mean embeddings, the conditional mean embedding does not constitute a single element in the RKHS but rather a collection of elements each indexed by the individual conditioning variable X. By fixing the conditioning variable $X = x$, we obtain the conditional mean embeddings $\mu_{Y|x}$ which we will define shortly. In turn, we will also define the conditional mean embedding operator $\mathcal{U}_{Y|X}$ which maps elements from the RKHS of X to the RKHS defined for Y, which satisfy the following two properties Song et al. [2009]:

1. $\mu_{Y|x} := \mathbb{E}_{Y|x}[\phi(Y)|x] = \mathcal{U}_{Y|X} k(x, \cdot)$
2. $\mathbb{E}_{Y|x}[g(Y)|x] = \langle g, \mu_{Y|x} \rangle_{\mathcal{H}_Y} \qquad \forall g \in \mathcal{H}_Y$

Song et al. [2009] then define the conditional mean embeddings as follows:

Definition 4. *The operator $\mathcal{U}_{Y|X}$ is defined as: $\mathcal{U}_{Y|X} := \mathcal{C}_{YX}\mathcal{C}_{XX}^{-1}$*

where $\mathcal{C}_{YX}$ is defined as $\mathcal{C}_{YX} = \mathbb{E}_{XY}[\phi_x(X) \otimes \phi_y(Y)] - \mu_X \otimes \mu_Y$. They further show in Song et al. [2009, Theorem 4] that this definition of the conditional mean embedding operator indeed satisfies the above two properties.

Theorem 1 (Song et al. [2009]). *Assuming that $\mathbb{E}_{Y|X}[g(Y)|X] \in \mathcal{H}$, the embedding of the conditional distributions in definition 4 satisfies properties 1 and 2.*

One thing to note is that contrary to the marginal mean embedding the operator $\mathcal{C}_{YX}\mathcal{C}_{XX}^{-1}$ for the conditional mean embeddings might not always exist in the continuous domain [Fukumizu et al., 2004, Song et al., 2009]. This ill-defined issue has been addressed in Fukumizu et al. [2013] by adding a regularization to the above definition, thereby defining the conditional mean embedding $\mu_{Y|x} = \mathcal{C}_{YX}(\mathcal{C}_{XX}^{-1} + \lambda\mathbf{I})(k(x, \cdot))$ where $\lambda > 0$ is the regularization parameter. Now that we have defined the conditional mean embedding, we move on to its empirical estimation as follows. Given a dataset $\mathcal{D} = \{(x_i, y_i)\}_{i=1}^m$ the empirical conditional mean embedding can be estimated:

Theorem 2 (Song et al. [2009]). *Let $\Phi_x = (\phi_x(x_1), ..., \phi_x(x_m))$, $\Phi_y = (\phi_y(y_1), ..., \phi_y(y_m))$, $k_x = \Phi_x^T \phi_x(x)$ and $H = I - \frac{1}{m}\mathbf{1}\mathbf{1}^T$. Then $\hat{\mu}_{Y|x}$ can be estimated as:*

$$\hat{\mu}_{Y|x} = \Phi_y(HK + \lambda mI)^{-1} H k_x$$

Note, that from the theorem above, the conditional mean embedding can in fact be seen as the solution to a vector-valued regression as first noted by Zhang et al. [2011] and then further elaborated by Grünewälder et al. [2012]. More concretely the conditional mean embedding is the solution to the following minimization problem given a sample $(x_1, z_1), ..., (x_m, z_m) \in \mathcal{X} \times \mathcal{H}_Y$:

$$\hat{\mathcal{E}}_\lambda(f) = \sum_{i=1}^{n} ||z_i - f(x_i)||_{\mathcal{H}_Y}^2 + \lambda ||f||_{\mathcal{H}_X}^2$$

This vector-valued regression interpretation for conditional mean embeddings will play a particularly important role in Chapter 4 where instead of using vector-valued kernel ridge regression for conditional mean embeddings we will be using a vector-valued Gaussian process. This will allow us to, in addition to capturing the full conditional distribution of the data, also capture the uncertainty due to lack of data and/or noise.

Applications of kernel mean embeddings: Over the years kernel mean embeddings have been used in various fields in machine learning including representation learning [Tsai et al., 2022], reinforcement learning [Lever et al., 2016, Kim and Park,

2018], distribution regression [Szabó et al., 2016], causal inference [Sgouritsa et al., 2013, Lopez-Paz et al., 2015b, Chen et al., 2014, Schölkopf et al., 2015, Mitrovic et al., 2018, Singh et al., 2019a, 2021, Chau et al., 2021c], sampling methods [Lacoste-Julien et al., 2015, Schuster et al., 2020, Chen et al., 2012, Sejdinovic et al., 2014], meta learning [Chen et al., 2020, Ton et al., 2021b,c] etc. These are just a few applications and many more can be found in the comprehensive review paper on kernel mean embeddings Muandet et al. [2017].

1.2.1.3 Gaussian Processes

Another key technique, in connection to kernel methods, that will become important in Chapter 4 is that of the Gaussian Process. We refer the reader to Rasmussen and Williams [2005b] for a comprehensive and full overview of the Gaussian Process. We included this section for completeness.

Definition 5 (Rasmussen and Williams [2005b])**.** *A Gaussian process is a collection of random variables, any finite number of which have a joint Gaussian distribution.*

A Gaussian process is a Bayesian non-parametric algorithm which allows us to put distributions over functions instead of parameters. In particular for Gaussian processes, they are fully defined through their mean function $m(x)$ and covariance function $k(x, x')$, such that:

$$m(x) = \mathbb{E}[f(x)] \tag{1.3}$$

$$k(x, x') = \mathbb{E}[(f(x) - m(x))(f(x') - m(x'))] \tag{1.4}$$

and can be written as:

$$f(x) \sim \mathcal{GP}(m(x), k(x, x'))$$

Hence for a given set of points X_* and assuming for simplicity that we have a simple constant mean function $m(x) = 0$ with some covariance function k such as in Equation 1.1, we obtain a sample from the GP as follows:

$$\mathbf{f}_* \sim \mathcal{N}(0, K(X_*, X_*))$$

However, we are usually not interested in sampling a random function from a multivariate Gaussian, but rather we would like to sample function that are able to

take into account a set of training data $\mathbf{x}, \mathbf{y} = \{(x_i, y_i)\}_{i=1}^m$ where $y = f(x) + \epsilon$ such that $\epsilon \sim \mathcal{N}(0, \sigma^2)$.

In this case we can compute the posterior distribution following Rasmussen and Williams [2005b] of the $\mathcal{GP}$ i.e. $f_*|\mathbf{x}, \mathbf{y}, X_* \sim \mathcal{N}(\tilde{\mathbf{f}}_*, cov(\mathbf{f}_*))$ where

$$\tilde{\mathbf{f}}_* = \mathbb{E}[\mathbf{f}_*|\mathbf{x}, \mathbf{y}, X_*] = K(X_*, \mathbf{x})[K(\mathbf{x}, \mathbf{x} + \sigma^2 I)]^{-1}\mathbf{y} \tag{1.5}$$

$$cov(\mathbf{f}_*) = K(X_*, X_*) - K(X_*, \mathbf{x})[K(\mathbf{x}, \mathbf{x}) + \sigma^2 I]^{-1}K(\mathbf{x}, X_*) \tag{1.6}$$

1.2.2 Causality

As already motivated in the introduction, many problems require causal reasoning instead of correlation-based reasoning. The infamous saying that "*correlation does not imply causation*" has been many times repeated and for good reason. The example that is often given to illustrate the logic flaw is that an increase in ice cream sales seems to be correlated with an increase in the crime rate. This is not because selling more ice cream induces more violent crime, but rather, both of these events seem to be more prevalent during hot weather. Hence simply looking at correlations can be very misleading if they are not analysed properly.

There are two main frameworks that have been proposed to put causality into a rigorous mathematical framework: the potential outcome framework proposed by Rubin [2005] and the structural causal model (SCM) framework proposed by Pearl [1995]. In this thesis, we will be mainly taking a look at Pearl's framework and we leave the interested reader to consult Rubin [2005] for the alternative modelling framework.

Before we start with the definitions of Pearl's framework of SCM's we would like the reader to note that causality is broadly divided into causal discovery and causal inference.

Causal discovery is the field of inferring the causal graph from a set of observational data points, whereas causal inference aims to draw causal conclusions based on the observational data and identifiability assumptions. Each of these tasks in itself is challenging and hence we attempt to fix certain shortcomings of causal discovery as well as causal inference in this thesis (Chapter 3 and 4 respectively).

Let us first define the structural causal model.

Definition 6. *(Structural Causal Model)[Pearl et al. [2000]] A structural causal model (SCM) M is defined as follows:*

1. *A set* $U = \{u_1, ..., u_n\}$ *of noise or exogenous variables, representing factors outside the model which nevertheless affect the relationship within the model*
2. *A set* $V = \{V_1, ..., V_n\}$ *of observed endogenous variables where each* V_i *is functionally dependent on a subset* PA_i *of* $U \cup V \setminus \{V_i\}$
3. *A set* F *of functions* $\{f_1, ..., f_n\}$ *such that each* f_i*, determines the value of* $V_i \in V, v_i = f_i(pa_i, u_i)$
4. *A joint probability distribution* $P(u)$ *over* U *which is factorizable.*

Note, that in addition, we will only be dealing with Directed Acyclic Graphs (DAGs) which are causal graphs that do not contain any cycles i.e. circular causal relationships.

1.2.2.1 Rules of *do*-calculus

Now that we have established the foundational framework, we will introduce the necessary tools for causal inference which is *do*-calculus of Pearl [1995, 2012]. Interventions in *do*-calculus are defined through the $do(x)$ operation, which emulates the physical interventions of deleting functions from the SCM and setting them to constants $X = x$. The rest of the model remains unchanged. Let X, Y, Z and W be arbitrary disjoint sets of nodes in a causal graph G. For notational convenience, we will denote $G_{\overline{X}}$ the graph obtained by removing all arrows going into X. Equivalently, we will denote $G_{\underline{X}}$ the graph obtained when removing all arrows coming out of X. We will subscript the conditional independence statements according to which graph we are considering.

The three rules proposed in Pearl [1995] that are valid for every *do*-operation with respect to G are:

1. **Rule 1:** (Insertion/deletion of observation):

$$P(y|do(x), z, w) = P(y|do(x), w) \text{ if } (Y \perp\!\!\!\perp Z|X, W)_{G_{\overline{X}}}$$

2. **Rule 2:** (Action/observation exchange):

$$P(y|do(x), do(z), w) = P(y|do(x), z, w) \text{ if } (Y \perp\!\!\!\perp Z|X, W)_{G_{\overline{X},\underline{Z}}}$$

3. **Rule 3:** (Insertion/deletion of action):

$$P(y|do(x), do(z), w) = P(y|do(x), w) \text{ if } (Y \perp\!\!\!\perp Z|X, W)_{G_{\overline{X},\overline{Z(W)}}}$$

where $Z(W)$ is the set of Z-nodes that are not ancestors of any W-node in $G_{\overline{X}}$. Notation was taken from Pearl [2012].

For any interventional query $P(Y|do(x))$ on any causal graph G, we then repeatedly apply the rules above in order to remove the *do*-operation. If this is possible using the observational data available, we call the causal graph identifiable. Furthermore, it has been proven to be complete Huang and Valtorta [2012], Shpitser and Pearl [2006], i.e. that if we cannot remove the *do*-operations with *do*-calculus, causal query or intervention is not identifiable.

As an example, let us take a closer look at Figure 1.3 where we would like to reason about $P(Y|do(x))$. This corresponds to removing all the arrows that go into the intervened variable X i.e. going from the left image to the right in Figure 1.3. This specific case is also known as the backdoor adjustment, which we will derive in the following using the three rules above.

Firstly, let us marginalize over the variable Z:

$$P(y|do(x)) = \sum_x P(y|do(x), z)P(z|do(x))$$

Now we can apply Rule 2 which states that $P(y|do(x), z) = P(y|x, z)$ because $(Y \perp\!\!\!\perp X|Z)_{G_{\underline{X}}}$ and hence we get:

$$P(y|do(x)) = \sum_x P(y|x, z)P(z|do(x))$$

Next, we apply rule 3 which states that $P(z|do(x)) = P(z)$ because $(X \perp\!\!\!\perp Z)_{G_{\overline{X}}}$ which lastly leads us to the backdoor adjustment formula, where all the *do*-operators have been replaced.

$$P(y|do(x)) = \sum_x P(y|x, z)P(z)$$

Note that in the case where Z is unobserved, the above quantity with the given causal graph G cannot be estimated and is hence unidentifiable.

1.2.2.2 Causal Discovery

The second pillar of causality is that of causal discovery. In most causal inference problems we have to explicitly assume to know the causal graph structure, which in itself can be a big assumption. Hence, causal discovery plays an equally important role in any causality problem as causal inference itself.

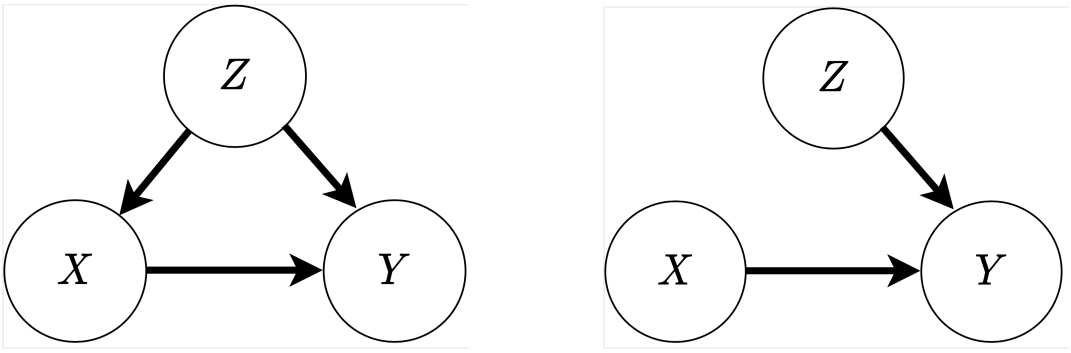

Figure 1.3: (left) Causal graph from which we collected data. (right) Causal Graph from which we would like to reason.

There are many methods that have been proposed to recover the causal graph from observational data [Glymour et al., 2019, Spirtes et al., 2000, Peters et al., 2017]. However many of them (without explicit assumptions on the edges) only recover the causal graph only up to markov equivalence. i.e. some of the edges might not be oriented.

This problem of orienting the edged is very challenging as we do not have access to randomized control trials (RCTs). RCTs are the gold standard experiments designed to specifically isolate the effect of a certain node in the causal graph. However, in many cases, RCTs are not feasible due to financial or ethical constraints. For example, it would not be ethical to force someone to smoke in order to determine the isolated effects of smoking. Hence statistical techniques with certain structural assumptions have been invented to still draw causal conclusions in these scenarios. We restrict ourselves in this thesis to the bivariate case of causal discovery and refer to [Glymour et al., 2019, Spirtes et al., 2000] for further details on the more general case.

One stream of research in the bivariate case is to exploit the cause-effect asymmetry to identify causal direction [Mooij et al., 2016]. By assuming that the causal mechanisms in the SCM are restricted to a certain class of functional causal models, we say that X is causing Y if $p(Y|X)$ is identifiable i.e. can be identified by the functional causal models but $p(X|Y)$ cannot be identified. This use of FCM allows us to determine the inherent asymmetry in the problem by imposing different function relationships between nodes. Here below, we cite some of the more popular methods. Models that put restrictions on the class of causal mechanisms, as well as noise, are Linear non-Gaussian acyclic models (LiNGAM) Shimizu et al. [2006], additive noise models

(ANM) Hoyer et al. [2008], post-nonlinear models Zhang and Hyvarinen [2012], etc. A model that views the problem from a information-theoretical perspective can be found in Daniusis et al. [2012]. Methods that use the idea of minimum description length (MDL) are Mian et al. [2021], Marx and Vreeken [2017], Mitrovic et al. [2018]. Lastly, Goudet et al. [2018], Ton et al. [2021c] have also proposed generative methods with neural networks to determine the causal relationship.

1.2.3 Meta Learning

1.2.3.1 Meta learning motivation and setup

Many machine learning models have achieved superhuman performances [Silver et al., 2017, 2018, Schrittwieser et al., 2020] but are trained to specialize at a specific task. Contrary to humans, who are remarkably able to adapt quickly to new unseen tasks i.e. if someone plays tennis well, it is likely that the same person would be decent at table tennis as well, given the similarity in the sport. This, however, cannot be said about most machine learning models, which are trained for one purpose and one purpose only. Hence, the field of meta learning aims to bridge this gap, by training models that are able to generalize well to unseen tasks when only provided with limited data.

Let us first start with defining the meta learning setup here. In particular, we will be taking a closer look at meta learning for regression. The goal of meta learning is to train a model that is able to perform well over the distribution of datasets $P(\mathcal{D})$. A simple example of a distribution over tasks would be datasets $\mathcal{D}_i = \{(x_j^i, y_j^i)\}$ where $X \sim \mathcal{N}(0,1)$ and $Y \sim a * sin(bX - c)$ such that $a, b, c \sim U(0,1)$. In this case, each dataset $\mathcal{D}_i$ corresponds to a specific sinusoidal curve and we can sample various sinusoidal datasets which we call tasks. The objective that we would like to optimize ideally is:

$$\theta^* = \underset{\theta}{\operatorname{argmin}}\, \mathbb{E}_{\mathcal{D}\sim P(\mathcal{D})}[\mathcal{L}_\theta(\mathcal{D})]$$

where $\mathcal{L}_\theta(\mathcal{D})$ denotes the loss given a model $\mathcal{M}_\theta$ such as for example MSE loss. Note that contrary to standard supervised learning, we treat each dataset as a datapoint.

Another important difference between meta learning and standard supervised learning is the way we train and test our models. For a given $P(\mathcal{D})$ we firstly split tasks into training tasks $\mathcal{D}^{train}$ and test tasks $\mathcal{D}^{test}$, where each tasks consists of covariates x and their respective labels y (not for the test set).

Meta Training: During training for each task $\mathcal{D}_i^{train} = \{(x_j^i, y_j^i)\}_{j=1}^m$ in $\mathcal{D}^{train}$ we additionally split the dataset into support set $S = \{(x_j^i, y_j^i)\}_{j=1}^q$ and target set $T = \{(x_j^i, y_j^i)\}_{j=q+1}^m$. We then use S as the input to our model $\mathcal{M}_\theta(x, S)$ and use T to compute how well $\mathcal{M}_\theta$ generalizes with respect to $\mathcal{L}_\theta$ and compute the respective backprogation to train θ. Note that $\mathcal{M}_\theta$ and $\mathcal{L}_\theta$ can be very general and the main differences of meta learning algorithms is how these are defined.

Meta Testing: At test time, given a support set S we can predict on any given target set T where we are now not given the labels y anymore. In this case we can simply use $\mathcal{M}_\theta$ for prediction on T and compute the performance using $\mathcal{L}_\theta$.

1.2.3.2 Meta learning types

Meta learning is mainly divided into 4 algorithm types and we refer to [Li et al., 2021, Weng, 2018] for a comprehensive overview whose exposition we closely follow below.

Learning an initialization: Initialization based algorithms aim to learn initializations of a given model $\mathcal{M}_\theta$ in such a way that for a given new unseen task, we only require a few gradient updates on θ in order to adapt well. Examples of these initialization-based meta learning algorithms can be found abundantly [Finn et al., 2018, Jamal and Qi, 2019, Chen et al., 2019, Tokmakov et al., 2019, Finn et al., 2017] and the seminal work has been conducted by [Finn et al., 2017].

Generation of parameters meta learning: Next type of meta learning algorithm is the parameter generation ones. Here again, having trained on various tasks previously, these methods aim to construct a second model which is able to generate parameters for the primary model that is then used for the prediction task. There have been many works that have focused on this area of meta learning and here we list only a few: [Bertinetto et al., 2016, Munkhdalai and Yu, 2017, Li et al., 2019a, Gidaris and Komodakis, 2018, 2019, Qi et al., 2018]

Memory-based meta learning: Memory-based meta learners, generally speaking, follow the same principle, which is to define a read, write and use memory efficiently for the meta learning tasks. In particular, these methods aim to leverage different types of neural networks such as FeedForward Networks, LSTMs [Hochreiter and Schmidhuber, 1997], CNN [LeCun et al., 1995] etc. in order to efficiently save and access the stored information which is then accessed at meta testing time. Several examples that make use of this strategy are [Munkhdalai et al., 2018, Mishra et al., 2017, Santoro et al., 2016].

Embedding-based meta learning: Lastly and most importantly, we overview embedding-based meta learning which is also the route that this thesis has opted for. In essence, given a support set S, the goal is to embed the datapoints from S in an efficient manner such that given data points from a target set T can utilize the information contained in the embedding to then perform inference. The steps can be described more concretely below: During training

1. We are given support set $S = \{(x_i, y_i)\}_{i=1}^m$ and a target set $T = \{(x_i^*, y_i^*)\}_{i=1}^j$
2. We construct an embedding of the support set S i.e. $\mu = f_\theta(S)$.
3. During training we then train an additional function $g(\mu, x^*)$ to predict y^*

At test time we are then only given support set $S = \{(x_i, y_i)\}_{i=1}^m$ but only a target set $T = \{x_i^*\}_{i=1}^j$ without the labels. We can then construct the embedding μ for each task separately, which subsequently allows us to adapt the model g to predict well on the new tasks. We refer to embedding-based meta learning as methods that follow the same structure as described in [Garnelo et al., 2018a]. Other methods that follow a similar structure are listed here [Kim and Park, 2018, Gordon et al., 2020, Ton et al., 2021c,b, Garnelo et al., 2018b], each with their own variation which fixes certain shortcomings of the vanilla version proposed in [Garnelo et al., 2018a].

1.3 Thesis outline

This thesis contains 3 published papers followed up with a conclusion and a future work chapter. Even though not included, I have been fortunate enough to have been part of several other works which I list below.

- Towards a unified analysis of random Fourier features
  Zhu Li, **Jean-Francois Ton**, Dino Oglic, Dino Sejdinovic
  Accepted at ICML 2019
- Metafun: Meta-learning with iterative functional updates
  Jin Xu, **Jean-Francois Ton**, Hyunjik Kim, Adam Kosiorek, Yee Whye Teh
  Accepted at ICML 2020
- Automated Model Selection with Bayesian Quadrature
  Henry Chai, **Jean-François Ton**, Roman Garnett, Michael A Osborne
  Accepted at ICML 2019

- Robust pruning at initialization
  Soufiane Hayou, **Jean-Francois Ton**, Arnaud Doucet, Yee Whye Teh
  Accepted at ICLR 2021

- Grassmann Stein Variational Gradient Descent
  Xing Liu, Harrison Zhu, **Jean-François Ton**, George Wynne, Andrew Duncan
  Accepted at AISTATS 2022

- Regularized Training of Nearest Neighbor Language Models
  Jean-Francois Ton, Walter Talbott, Shuangfei Zhai, Josh Susskind
  Under Review at NAACL

For the sake of a cohesive thesis, I have thus decided to present only the following three chapters.

In chapter 2 we will present our work on noise contrastive estimation for conditional density in the meta learning setting. Motivated by the popular meta learning algorithm Neural process [Garnelo et al., 2018a,b], we set out to tackle a problem that was previously not considered, which is that of multimodality in the output space. Many algorithms consider a Gaussian likelihood for simplicity when training a model, however, in cases that we will show later a uni-modal assumption on the outputs can be misleading. When presented with data where a single covariate can have multiple outputs such as for example in computational chemistry it is essential to not just model the mean and variance of the distribution but rather the whole distribution. This is where we will be utilizing the power of conditional mean embeddings that we have introduced in our background section. In addition to that, we will show how we can perform this conditional density estimation in the meta learning setting allowing us to capture densities that were previously not possible with the standard Neural Process [Garnelo et al., 2018a,b].

- Noise Contrastive Meta-Learning for Conditional Density Estimation using Kernel Mean Embeddings
  Jean-Francois Ton, Lucian Chan, Yee Whye Teh, Dino Sejdinovic
  https://arxiv.org/abs/1906.02236
  Accepted at AISTATS 2021

In chapter 3, we then build upon our previous paper where we now extend the idea of meta learning using conditional mean embeddings to causal discovery. In particular, we consider the algorithm "*Causal Generative Neural Networks*" by Goudet et al. which essentially trains a two generative models, one where we aim to generate Y based on X and one where we want to generate X based on Y. Depending on which direction is "easier" to generate Goudet et al. are then able to determine the causal direction of the dataset. Their methods, however, have two major shortcomings: 1) The need for a lot of data to train the generative model on 2) Slow during inference as they have to train a generative model. We address these problems using a meta learning framework where we meta train a generative network that at inference time does not need to be retrained. Here again, we use kernel mean embeddings to capture the task at hand before using it as a feature for the generative model.

- Meta Learning for Causal Direction
  Jean-Francois Ton, Dino Sejdinovic, Kenji Fukumizu
  https://arxiv.org/abs/2007.02809
  Accepted at AAAI 2021

Lastly, inspired by the previous project on causal discovery, we then present in chapter 4 an application of kernel mean embeddings to a different task in causal reasoning. In particular, we proposed a methodology to rewrite the causal adjustment formulas we presented in the background section using kernel mean embeddings while also accounting for the underlying uncertainty due to e.g. data quality or data quantity issues. This is achieved by noting that independent and concurrent work of Singh et al. [2021] allows us to rewrite the causal adjustment formulas using kernel mean embeddings. We, in addition to that, take this idea a step further by realizing that kernel mean embeddings can be combined with Gaussian process. In particular, when considering the problem of combining two causal graphs, i.e. causal data fusion, we are able to effectively utilize both the Gaussian Process as well as kernel mean embeddings for enhanced causal reasoning.

- BayesIMP: Uncertainty Quantification in Causal Data Fusion
  Siu Lun Chau*, **Jean-Francois Ton***, Javier Gonzalez, Yee Whye Teh, Dino Sejdinovic
  https://arxiv.org/abs/2106.03477
  Accepted at Neurips 2021

2

Noise Contrastive Meta-Learning for Conditional Density Estimation using Kernel Mean Embeddings

Abstract

Current meta-learning approaches focus on learning functional representations of relationships between variables, *i.e.* estimating conditional expectations in regression. In many applications, however, the conditional distributions cannot be meaningfully summarized solely by expectation (due to *e.g.* multimodality). We introduce a novel technique for meta-learning conditional densities, which combines neural representation and noise contrastive estimation together with well-established literature in conditional mean embeddings into reproducing kernel Hilbert spaces. The method shows significant improvements over standard density estimation methods on synthetic and real-world data, by leveraging shared representations across multiple conditional density estimation tasks.

2.1 Introduction

The estimation of conditional densities $p(y|x)$ based on paired samples $\{(x_i, y_i)\}_{i=1}^n$ is a general and ubiquitous task when modelling relationships between random objects x and y. While the standard problem of regression focuses on estimating the conditional expectations $\mathbb{E}[y|x]$ of responses y given the features x, many scenarios require a more expressive representation of the relationship between x and y. In particular, the distribution of y given x may exhibit multimodality or heteroscedasticity. A simple example of such a relation between x and y can be seen in the equation of a circle, as for any given $x \in [-1, 1]$ there are two potential values $\pm\sqrt{1 - x^2}$ the model could regress on. In this case, any standard regression model would fail to capture the true dependence between y and x, because clearly $\mathbb{E}[y|x] = 0$.

Thus conditional density estimation requires a flexible nonparametric model of the conditional density and not just the expectation. Estimating conditional densities becomes even more challenging when the sample size is small. Hence, we approach this problem from a meta-learning perspective, where we are faced with a number of conditional density estimation tasks, allowing us to transfer information between them via a shared learned representation of both the responses y and the features x.

Our contribution can be viewed as a development which parallels that of Neural Processes (NP) [Garnelo et al., 2018b] and conditional Neural Processes (CNP) [Garnelo et al., 2018a] in the context of regression and functional relationships. Our proposed method is applicable to a much broader set of relationships between random objects, i.e. cases where for a given x there are multiple potential responses y. To that end, we will make use of the framework of conditional mean embeddings (CME) of distributions into reproducing kernel Hilbert spaces (RKHSs) [Muandet et al., 2017, Song et al., 2013], which we discuss on in section 2.2.

In the RKHS literature, the feature maps that yield kernel mean embeddings that fully characterize probability distributions correspond to the notion of characteristic kernels [Sriperumbudur et al., 2011b] and are infinite-dimensional. However, such kernels can often be too rigid for specific tasks (e.g. a Gaussian RBF kernel) when we do not have many data points [Wilson et al., 2016, Wenliang et al., 2018]. Moreover, even though they give a unique representation of a probability distribution and can

be a useful tool to represent conditional distributions[1], they do not yield (conditional) density estimates, as they are merely points in the RKHS, and it is not clear how to adopt them for such tasks.

To address this challenge, we propose using a technique based on noise contrastive estimation (NCE) [Gutmann and Hyvärinen, 2012]. We treat CMEs as dataset features in the binary classifier discriminating between the true and artificially generated samples of (x_i, y_i) pairs. Hence, even though CME estimation for fixed feature maps is well understood [Song et al., 2013, Muandet et al., 2017], we are concerned with the challenge of linking our CME estimates back to the conditional density estimation (CDE) task. This requires *learning* the feature maps, ϕ_x, ϕ_y, as they define the CME, which in turn determines the binary classifier for NCE.

In particular, we propose to use neural networks to learn appropriate feature maps ϕ_x and ϕ_y by adopting the meta-learning framework, i.e. by considering a number of (similar) conditional density estimation tasks simultaneously. Our meta-learning setting is concerned with CDE using small amounts of data (e.g. in the synthetic data setting each dataset only has 50 points). In this small data regime, standard CDE methods cannot work well even on simple 1D problems as they usually require many data points to train on. Hence, in this paper, we mainly focus on low dimensional problems and discuss the higher dimensional problem in the conclusion.

Following the meta-learning principle that test and train conditions must match [Vinyals et al., 2016], we consider the case where multiple small datasets are available to train the system, such that it is able to estimate the conditional density well on a new unseen (also small) dataset. This is effectively a system for sharing statistical strength across multiple CDE problems. In contrast to prior state-of-the-art work on meta-learning for regression, i.e. (attentive) Neural Process [Garnelo et al., 2018b, Kim et al., 2019], our proposed method is the first that can capture multimodal and complex conditional densities.

The proposed method is validated on synthetic and real-world data exhibiting multimodal properties, namely on the NYC taxi data used in [Trippe and Turner, 2018] to model the conditional densities of dropoff locations given the taxi tips, as well as on Ramachandran plots from computational chemistry [Gražulis et al., 2011], which represent relationships between dihedral angles in molecular structures. We

[1] In particular, CME $\mathbb{E}[\phi_y(y)|x]$ can be used to estimate conditional expectations $\mathbb{E}[h(y)|x]$ for a broad class of functions h, namely functions in the RKHS determined by the feature map ϕ_y.

demonstrate significant improvement in terms of held-out loglikelihood over standard conditional density estimation methods, including those based on meta-learning.

Algorithm 1 MetaCDE Training (one training step)

Input: $\mathcal{D}_{context} = \{(x_1^{cq}, y_1^{cq}) \dots (x_{m_{cq}}^{cq}, y_{m_{cq}}^{cq})\}_{q=1}^{l}$ and $\mathcal{D}_{target} = \{(x_1^{tq}, y_1^{tq}) \dots (x_{m_{tq}}^{tq}, y_{m_{tq}}^{tq})\}_{q=1}^{l}$ $\mathcal{D}^{test} = \{\mathcal{D}^{test}\}$
Output: Learned ϕ_x^θ, ϕ_y^θ, b_θ.

1: **for** $q = 1, \dots, l$ **do**
2: Compute $\widehat{\mathcal{C}}_{Y|X}$ using $\mathcal{D}_{context}^q$ Eq.(2.3)
3: Generate κ fake samples $\{y_{ij}^f\}_{i=1}^{\kappa}$ from the fake distribution $p_f(y)$ for each y_j^{tq}
4: Compute $\widehat{\mu}_{Y|X=x_j^{tq}}$, $j = 1 \dots m_{tq}$ Eq.(2.7)
5: Now construct the respective *scoring functions* $s_\theta(x_j^{tq}, y_j^{tq})$ and $s_\theta(x_j^{tq}, y_{ij}^f)$ using ϕ_y Eq. (2.10)
6: **end for**
7: Optimize the parameters θ of the NNs ϕ_x^θ, ϕ_y^θ, b_θ using *Adam* jointly over all tasks Eq.(2.11)
8: **return** ϕ_x^θ, ϕ_y^θ, b_θ

Algorithm 2 MetaCDE Testing

Input: $\mathcal{D}_{context}^* = \{(x_1^{cq*}, y_1^{cq*}) \dots (x_{m_{cq*}}^{cq*}, y_{m_{cq*}}^{cq*})\}_{q=1}^{l}$, $\mathcal{D}_{target}^* = \{(x_1^{tq*}) \dots (x_{m_{tq*}}^{tq*})\}_{q=1}^{l}$
Input: ϕ_x^θ, ϕ_y^θ, b_θ, y Evaluating the conditional density at y
Output: $p_\theta^q(y|x^{tq*})$, for $q = 1, ..., l$ (Conditional density for each task.)

1: **for** $q = 1, \dots, l$ **do**
2: Compute the CMEO $\widehat{\mathcal{C}}_{Y|X}$ using Eq.(3) with $\mathcal{D}_{context}^{q*}$ and trained ϕ_x, ϕ_y
3: Compute $s_\theta^q(x^{tq*}, y)$ using Eq.(10) with $\widehat{\mathcal{C}}_{Y|X}$ and trained ϕ_x, ϕ_y
4: Compute the conditional density $p_\theta^q(y|x^{tq*})$ using Eq.(4) i.e. with $s_\theta^q(x^{tq*}, y)$ and b_θ
5: **end for**

2.2 Background

We first introduce the notation used throughout this chapter. Let $\mathcal{D} = \{(x_j, y_j)\}_{j=1}^n$ be the observed dataset, with $x_j \in \mathcal{X}$ being the input and $y_j \in \mathcal{Y}$ being the output. We denote the learned RKHS of inputs X and responses Y by $\mathcal{H}_X$ and $\mathcal{H}_Y$ respectively. Kernels of $\mathcal{H}_X$ and $\mathcal{H}_Y$ are denoted $k_x(\cdot, \cdot)$ and $k_y(\cdot, \cdot)$, and the corresponding feature maps are $\phi_x(\cdot)$ and $\phi_y(\cdot)$, i.e. $k_x(x_1, x_2) = \langle \phi_x(x_1), \phi_x(x_2) \rangle$ and similarly for k_y.

2.2.1 Conditional Mean Embeddings (CME)

Kernel mean embeddings of distributions provide a powerful framework for representing and manipulating probability distributions [Song et al., 2013, Muandet et al., 2017]. Formally, given sets $\mathcal{X}$ and $\mathcal{Y}$, with a distribution P over the random variables (X, Y) taking values in $\mathcal{X} \times \mathcal{Y}$, the conditional mean embedding (CME) of the conditional distribution of $Y|X = x$ is defined as:

$$\mu_{Y|X=x} := \mathbb{E}_{Y|X=x}[\phi_y(Y)] = \int_{\mathcal{Y}} \phi_y(y)p(y|x)dy. \tag{2.1}$$

Intuitively, Eq.2.1 allows us to represent a probability distribution $p(y|x)$ in a function space (RKHS), by taking the expectation under $p(y|x)$ of features $\phi_y(y) \in \mathcal{H}_Y$. Hence, for each value of the conditioning variable x, we obtain an element $\mu_{Y|X=x}$ of $\mathcal{H}_Y$.

Following [Song et al., 2013], the CME can be associated with the operator $\mathcal{C}_{Y|X} : \mathcal{H}_X \to \mathcal{H}_Y$, also known as the conditional mean embedding operator (CMEO) s.t.

$$\mu_{Y|X=x} = \mathcal{C}_{Y|X}\phi_x(x) \tag{2.2}$$

where $\mathcal{C}_{Y|X} := \mathcal{C}_{YX}\mathcal{C}_{XX}^{-1}$, $\mathcal{C}_{YX} := \mathbb{E}_{Y,X}[\phi_y(Y) \otimes \phi_x(X)]$ and $\mathcal{C}_{XX} := \mathbb{E}_{X,X}[\phi_x(X) \otimes \phi_x(X)]$.

As a result, [Song et al., 2013] have shown that if finite-dimensional feature maps ϕ_x and ϕ_y are used, the finite sample estimator of $\mathcal{C}_{Y|X}$ based on the dataset $\{(x_j, y_j)\}_{j=1}^n$ can be written as

$$\widehat{\mathcal{C}}_{Y|X} = \Phi_y(K + \lambda I)^{-1}\Phi_x^T \tag{2.3}$$

where $\Phi_y := (\phi_y(y_1), \ldots, \phi_y(y_n))$ and $\Phi_x := (\phi_x(x_1), \ldots, \phi_x(x_n))$ are the feature matrices, $K := \Phi_x^T\Phi_x$ is the kernel matrix with entries $K_{i,j} = k_x(x_i, x_j) := \langle\phi_x(x_i), \phi_x(x_j)\rangle$, and $\lambda > 0$ is a regularization parameter.

In fact, when using finite-dimensional feature maps, the conditional mean embedding operator is simply a solution to a vector-valued ridge regression problem (regressing $\phi_y(y)$ to $\phi_x(x)$) [Grünewälder et al., 2012], which allows computation scaling linearly in the number of observations n. The Woodbury matrix identity allows us to have computations of either order $\mathcal{O}(n^3)$ or $\mathcal{O}(d^3) + \mathcal{O}(d^2n)$, where d is the dimension of the feature map ϕ_x. In our case, given that we are in the meta-learning setting, the dataset size n is usually rather small and hence the CME can be efficiently computed.

Now that we can compute the CME for a given new x, i.e. embed the conditional distribution $p(y|x)$ into a RKHS, the problem is how we are going to model the conditional density using the CME. In order to do that, we use ideas from noise contrastive estimation.

2.2.2 Noise Contrastive Estimation (NCE)

The seminal work on noise contrastive estimation [Gutmann and Hyvärinen, 2012] allows converting density estimation into binary classification, via learning to discriminate between noisy artificial data and real data.

In order to understand the methodology more concretely, let us first assume that the true underlying density of the data is $p(y|x)$ and that the distribution of the fake/artificial data is $p_f(y)$ (taken to be independent of for simplicity x). Following [Gutmann and Hyvärinen, 2012] we set up the classification problem, such that we see κ times more fake examples than the real ones, which are all fed together with their labels (True/Fake) into the classifier. In Van den Oord et al. [2018] they related this type of classification problem to mutual information estimation. Hence, the data arises from $\frac{1}{\kappa+1}p(y|x) + \frac{\kappa}{\kappa+1}p_f(y)$ and the probability that, conditioned on a x, any given y comes from the true distribution is

$$P(\text{True}|y,x) = \frac{p(y|x)}{p(y|x) + \kappa p_f(y)}.$$

Since our goal is to learn the true density $p(y|x)$, we construct the probabilistic classifier, by imposing a parameterized form of $p(y|x)$ as $p_\theta(y|x)$. In particular, we consider a generic density model given by

$$p_\theta(y|x) = \frac{\exp(s_\theta(x,y))}{\int \exp(s_\theta(x,y'))dy'} = \exp(s_\theta(x,y) + b_\theta(x)). \tag{2.4}$$

for some function $b_\theta : \mathcal{X} \to \mathbb{R}$ and $s_\theta : \mathcal{X} \times \mathcal{Y} \to \mathbb{R}$, the latter is referred to as the scoring function, following terminology in Mnih and Teh [2012]. Hence the model for our classifier will be

$$P_\theta(\text{True}|y,x) := \frac{p_\theta(y|x)}{p_\theta(y|x) + \kappa p_f(y)}. \tag{2.5}$$

Note that the parameters of the classifier are the parameters of s_θ and b_θ. Assuming for the moment that the learned probabilistic classifier $P_\theta(\text{True}|y,x)$ attains Bayes

optimality, we can deduce the point-wise evaluations of the true conditional density $p_\theta(y|x)$ directly from expression Eq.2.5 by simply rearranging for the density $p_\theta(y|x)$. We would also like to highlight some recent theoretical results with regards to NCE that have been developed in [Arora et al., 2019, Gutmann and Hyvärinen, 2012]. [Gutmann and Hyvärinen, 2010, Theorem1] state that one only requires the support of the noise distribution to cover the true density support to recover the underlying density.

In section 2.3, we will merge the idea of NCE together with CME to perform conditional density in the meta-learning setting. In particular, we will link our problem of computing the conditional density from the CME to NCE, by relating CME to $s_\theta(x, y)$ as defined in Eq.2.4.

2.2.3 Meta-Learning

Meta-learning is a growing area of which allows machine learning models to extract information from similar problems, and use this extracted (prior) information to solve new unseen problems quicker. Meta-learning and multi-task learning differ in the following way according to Finn [2019]:

- "*The meta-learning problem: Given data / experience on previous tasks, learn a* ***new task*** *more quickly and/or more proficiently*"
- "*The multi-task learning problem: Learn all of the tasks more quickly or more proficiently than learning them independently*"

In our case we perform meta-learning, by sharing statistical strength across multiple CDE problems. Standard CDE models usually require a lot of prior information to be useful in low data settings. In meta-learning however, this prior information is learned through the meta-learning phase, where the model is shown multiple small datasets during training such that given a new unseen dataset, the model is able to estimate the density well.

Before we introduce our method, we would like to outline the meta-learning experimental setup that we will be using throughout the experiments in section 2.5. Our setup is closely related to the one introduced in the papers on Neural Process [Garnelo et al., 2018b,a].

Let us define the set of tasks $\mathcal{T} = \{T_1, \ldots, T_l\}$ to be a set of l conditional density estimation tasks such that T_q corresponds to the dataset $\mathcal{D}^q = \{(x_i^q, y_i^q)\}_{i=1}^{m_q}$, where

$x_i^q \in \mathcal{X}$ and $y_i^q \in \mathcal{Y}$ share the same support across the tasks. During training, we split the task T_l into a "context set" and a "target set". The context set is used to summarize the task, whereas the target set is used to evaluate the summary of the context set. Hence, we use the target set to compute the loss and update the parameters of our model. We train our model jointly across multiple tasks and we can thus use this shared information to make better inference on similar tasks at testing time. In our experiments, during testing time we will only be presented with $m_q^{context}$ samples (i.e. only a context set) and are then asked to do inferences on any given set of target points.

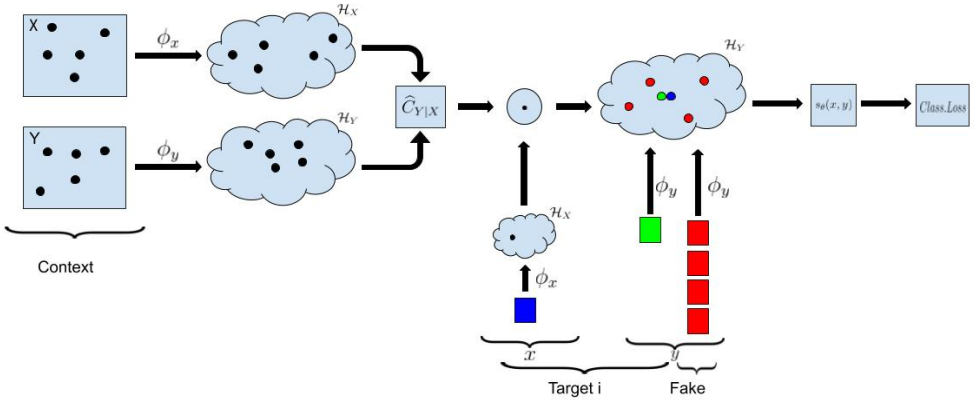

Figure 2.1: The context data is first passed through the feature maps to construct the CMEO ($\widehat{\mathcal{C}}_{Y|X}$), which is then used to project the new target x (dark blue) to $\mathcal{H}_Y$. We then compare this projection to the *True*(green) and *Fake* y's(red). Finally, we can compute the classification loss and back-propagate to update the parameters of the feature maps

2.3 Methodology

2.3.1 Conditional Density Estimation

As described in the previous section, the key ingredient of NCE is a classifier which can discriminate between the samples, i.e. $\{y_i\}_{i=1}^n$, from the true density, in our case the conditional $p(y|x)$, and fake samples, i.e. $\{y_i^f\}_{i=1}^{n\kappa}$, from the fake density $p_f(y)$.

Using the parameterization of Eq.2.4 for our density model, the probabilistic classifier we will adopt is:

$$\begin{aligned} P_\theta(\text{True}|y,x) &= \frac{\exp(s_\theta(x,y)+b_\theta(x))}{\exp(s_\theta(x,y)+b_\theta(x))+\kappa p_f(y)} \\ &= \sigma\left(s_\theta(x,y)+b_\theta(x)-\log(\kappa p_f(y))\right) \end{aligned} \tag{2.6}$$

where $\sigma(t) = 1/(1 + e^{-t})$ is the logistic function. Learning the parameters θ of s_θ and b_θ through this classification allows us to learn the density in Eq.2.4, given that it defines the density by design. In the next section, we discuss the scoring function $s_\theta(x, y)$ and its relation to the feature maps ϕ_x and ϕ_y.

We only have approximations to the Bayes classifier, hence it will be useful following [Gutmann and Hyvärinen, 2012] to model the normalizing constant b_θ separately. We should also note that $b_\theta(x) = -\log \int \exp(s_\theta(x, y'))dy'$ from Eq.2.4. While the contribution $b_\theta(x)$ is directly determined by the choice of s_θ, the calculation of b_θ is computationally intractable, and we therefore model $b_\theta(x)$ separately as it adds an extra independent component of y and is suggested in the original noise contrastive paper [Gutmann and Hyvärinen, 2012]. We empirically note that learning b_θ through a different network helps training and keeps the learned density normalized (see Appendix for details). To make fair comparisons to other methods in terms of loglikelihood, we added an extra post-normalization step which is described in section 2.5. For notation purposes, we collate all parameters of s_θ and b_θ into the parameter set θ.

2.3.2 The choice of the scoring function $s_\theta(x, y)$

We first map x_i and y_i using feature maps $\phi_x : \mathcal{X} \to \mathcal{H}_X$ and $\phi_y : \mathcal{Y} \to \mathcal{H}_Y$ respectively, which will be learned. We should note that we initially explored a fixed kernel choice with a characteristic kernel e.g. Gaussian RBF, but this resulted in poor empirical performance as the RBF was not flexible enough due to its restrictive stationarity constraints (see Appendix for experiments). Hence, in order to facilitate learning of these feature maps, we parametrized them with neural networks with θ. Note that in this case $\langle ., . \rangle_{\mathcal{H}_Y}$ is equivalent to a simple dot product between vectors.

Next, we compute the conditional mean embedding operator (CMEO) $\widehat{\mathcal{C}}_{Y|X} : \mathcal{H}_X \to \mathcal{H}_Y$ given in Eq.2.3. Using $\widehat{\mathcal{C}}_{Y|X}$, we can estimate the conditional mean embedding for any new x^* as

$$\widehat{\mu}_{Y|X=x^*} = \widehat{\mathcal{C}}_{Y|X}\phi_x(x^*). \tag{2.7}$$

Note that $\widehat{\mu}_{Y|X=x^*} \in \mathcal{H}_Y$. Hence we can evaluate the function for any given new $y^* \in \mathcal{Y}$ by using the reproducing property of the RKHS, the linearity of the inner

product and the definition in Eq.2.1,

$$\widehat{\mu}_{Y|X=x^*}(y^*) = \langle \widehat{\mu}_{Y|X=x^*}, \phi_y(y^*) \rangle_{\mathcal{H}_Y} \tag{2.8}$$

$$\approx \int k_y(y^*, y) p(y|x^*) dy. \tag{2.9}$$

Note that when using finite-dimensional feature maps, $\widehat{\mu}_{Y|X=x^*}$ is a vector. Intuitively, since a kernel k_y expresses the similarity between its inputs, we expect $k_y(y^*, y)$ to have high value when y^* is drawn from the true distribution and low value when drawn from the fake distribution p_f, which thus affects the value of the CME accordingly. This suggests the following form of the scoring function for our model:

$$s_\theta(x^*, y^*) = \langle \widehat{\mu}_{Y|X=x^*}, \phi_y(y^*) \rangle_{\mathcal{H}_Y}. \tag{2.10}$$

Furthermore, in order to investigate the importance of the CMEO task representation, we developed a purely neural version of our proposed method, which we call MetaNN. It differs from MetaCDE solely in the task representation. While MetaCDE uses kernel embeddings formalism to represent the task using CMEO calculated on the context points, MetaNN uses the DeepSets [Zaheer et al., 2017] approach, where the context pairs (x_i, y_i) are simply concatenated into a vector, and then passed through a neural network. The outputs are then averaged to obtain the task embedding to which any new x^* is concatenated to obtain the "neural" equivalent to CME. The same training procedure as MetaCDE follows.

We note that the concatenation of x and y encodes the joint distribution, rather than the conditional as in MetaCDE. While such task representation does preserve the relevant information, it is susceptible to changes in the marginal of x across tasks and conditional representations are intuitively better suited for the task of conditional density estimation. The experiments demonstrate the value of combining the CME formalism with neural representations and we obtain significantly better results with MetaCDE compared to MetaNN. Additional details on MetaNN and MetaCDE can be found in our Appendix.

Lastly, we also want to note that there are other choices for s_θ that could have been considered, i.e. ϵ-KDE etc. However, the reason we opted for CME is firstly, its flexibility of parameterization using NN and secondly, its ability and theoretical background of capturing conditional densities in a RKHS.

2.3.3 Training our proposed model

For a given task T_q corresponding to the dataset $\mathcal{D}^q = \{(x_i^q, y_i^q)_{i=1}^{m_q}\}$, we sample a set of fake responses $\{y_{i,j}^f\}_{i=1}^{\kappa}$ from p_f, associated to each y_j. In this case $y_{i,j}^f$ is the i^{th} sample from the fake distribution for data point y_j. We can now train the classifier using the model given in Eq.2.6 by maximizing conditional loglikelihood of the True/Fake labels, or equivalently, by minimizing the logistic loss:

$$\min_{\theta} \sum_{j=1}^{n} \left\{ \log\left(1 + \frac{\kappa p_f(y_j)}{\exp(s_\theta(x_j, y_j) + b_\theta(x_j))}\right) + \sum_{i=1}^{\kappa} \log\left(1 + \frac{\exp(s_\theta(x_j, y_{i,j}^f) + b_\theta(x_j))}{\kappa p_f(y_{i,j}^f)}\right) \right\}. \tag{2.11}$$

After the parameters θ of the classifier have been learned, the conditional density estimates can be read off from Eq.2.4. Note that we need to be able to evaluate the fake density pointwise. Hence we consider a simple kernel density estimator (KDE) of $p(y)$ as our fake density, which we can easily evaluate pointwise. To sample from this fake density p_f, we draw from the empirical distribution of all the y's (context and target) and add Gaussian noise with standard deviation being the bandwidth of the KDE (we use a Gaussian KDE for simplicity). This simple approach yielded good results (better than current SOTA) and hence we leave other methods, i.e. conditioning on x, for future work.

We also note that Eq.2.11 may be of an independent interest when learning feature maps for CMEs, i.e. where the goal is not necessarily density estimation, but other uses of CMEs discussed in [Song et al., 2013]. Even though estimation of CME corresponds to regression in the feature space, it is inappropriate to use the squared error loss of the feature-mapped responses to learn the feature maps themselves. The reason being that the notion of distance in the loss is changing as the feature maps are learned and as a results they are not comparable across different feature maps. In fact, it would be optimal for the feature map ϕ_y to be constant, as the squared error would then be zero, without having learned anything about the relationship between x and y.

2.3.4 Meta-Learning of Conditional Densities

Our proposed method will be trained as described in section 2.2.3 and is illustrated in Figure 2.1, which outlines of one meta-training step for a given task. The figure

		Synthetic	Chemistry	NYC
MetaCDE (Ours)	Loglike	**197.84 ± 22.4**	**-305.49 ± 46.9**	**-1685.52± 608.35**
MetaNN (Ours)	Loglike	132.776±130.87	-317.91±51.3	-2276.55 ± 608.9
	p-value	4.781e-06	1e-03	3.89e-10
Neural Process	Loglike	-81.11±18.5	-426.75± 47.3	-3050.2 ± 822.8
	p-value	<2.2e-16	<2.2e-16	3.89e-10
DDE	Loglike	162.98 ± 69.0	-399.68 ± 41.3	-2236.07 ± 565.9
	p-value	8.14e-07	1.65e-15	3.89e-10
KCEF	Loglike	-388.30 ± 703.1	-724.40 ± 891.6	-1695.89±435.4
	p-value	<2.2e-16	9.72e-14	0.025
LSCDE	Loglike	44.95 ± 74.3	-407.32 ± 80.1	-2748.01 ± 549.2
	p-value	<2.2e-16	2.57e-14	3.89e-10
e-KDE	Loglike	116.31 ± 236.9	-485.10 ± 303.4	-2337.90 ± 501.1
	p-value	2.38e-07	2.94e-14	4.13e-10

Table 2.1: Due to the varying difficulties in tasks, the variance in loglikelihoods is high. Hence we added the p-values of a one-sided Wilcoxon test to show that MetaCDE is significantly outperforming competing methods in terms of loglikelihood

illustrates how the loss is computed and we repeat this step for every training task. We also added a step by step algorithm in Alg.s1.

In this case, the CMEO, $\widehat{\mathcal{C}}_{Y|X}$ will be estimated using the context set, and the CMEs will be evaluated on the target set. The CMEO acts as the task embedding and the CME is used for the inferences on the target set.

During training, for each target example, we sample κ fake samples from $p_f(y)$ and represent them in $\mathcal{H}_Y$ using the feature map ϕ_y to construct s_θ so that Eq.2.6 can be computed for each of these $\kappa + 1$ samples (1 true and κ fakes). By also providing the labels (i.e. *True*(from data)/*Fake*(from noise distribution), we proceed by training the classifier. Thus we learn the parameters θ of neural networks ϕ_x, ϕ_y and b_θ using the objective in Eq.2.11 jointly over all tasks.

The resulting feature maps hence generalize across tasks and can be readily applied to new, previously unseen datasets, where we are simply required to compute the scoring function $s_\theta(x, y)$ and normalization constant $b_\theta(x)$.

2.3.5 Choice of the Fake Distribution in NCE

The choice of the fake distribution plays a key role in the learning process here, especially because we are interested in conditional densities. In particular, if the fake density is significantly different from the marginal density $p(y)$, then our model could learn to distinguish between the fake and true samples of y simply by constructing a "good

enough" model of the marginal density $p(y)$ on a given task while completely ignoring the dependence on x (this will then lead to feature maps that are constant in x).

This becomes obvious if, say, the supports of the fake and the true marginal distribution are disjoint, where clearly no information about x is needed to build a classifier – i.e. the classification problem is "too easy".

Thus, ideally we wish to draw fake samples from the true marginal $p(y)$ in a given task and hence, we propose to use a kernel density estimate (KDE) of y's as our fake density in any given task.

In particular, a kernel density estimator of $p(y)$ is computed on all responses y (context and target) during training. In our experiments we demonstrate that this choice ensures that the fake samples are sufficiently hard to distinguish from the true ones, requiring the model to learn meaningful feature maps which capture the dependence between x and y and are informative for the CDE task.

2.4 Related Work

Noise contrastive estimation for learning representations has been considered in the setting of Natural Language Processing (NLP). Mnih and Teh [2012] and Ma and Collins [2018] only focus on learning discrete distributions in the context of NLP using NCE. They achieved impressive speedups over other word embedding algorithms as they avoid computing the normalizing constant.

More recently, [Van den Oord et al., 2018] introduced a NCE method for representation learning, however, they focused on learning an expressive representation in the unsupervised setting, by optimizing a mutual information objective. Since they focused on representation learning, they did not require to evaluate the fake density point-wise. An alternative method that used the idea of fake examples was [Zhang et al., 2018b], which trained a GAN in order to use the resulting discriminator for few-shot classification. Their focus was on representation learning, rather than density estimation [Goodfellow, 2019].

RKHSs in density models have been proposed, for example [Dai et al., 2018, Arbel and Gretton, 2017] considered training kernel exponential family models, where the main bottleneck was the calculation of the normalizing constant. [Dai et al., 2018] exploited the flexibility of kernel exponential families to learn conditional densities and avoided computing the normalizing constants by solving so-called nested Fenchel

duals. [Arbel and Gretton, 2017] trained kernel exponential family models using score matching criteria, which allowed them to bypass normalizing constant computation. Their method however required computing and storing the first- and second order derivatives of the kernel function for each dimension and each sample, and thus required $\mathcal{O}(n^2d^2)$ memory and $\mathcal{O}(n^3d^3)$ time, for n datapoints in d dimensions.

CME-based sampling methods, such as kernel herding [Chen et al., 2012, Kanagawa, 2016], have also been proposed. These methods are interested in producing representative samples using the CME as a reference. However, we use the CME as a feature to model the conditional density.

Another work that has recently used CME as task embeddings is meta-CGNN [Ton et al., 2021c]. However, they are mainly interested in the causal direction detection settings.

All the above methods assume access to a large dataset to train on, as most of the theoretical guarantees only hold in the large data setting. We however, are interested in scenarios in which we are presented with only little data i.e. around 50 datapoints at test time in our synthetic dataset experiments. Hence, the work that come closest to ours is the Neural Process (NP) [Garnelo et al., 2018b] as they also consider the meta-learning setting for density estimation. However, our method extends on their methodology in two major ways. First of all, we embed our task into an CMEO which takes into account the input covariates and responses in the context sets. The NP on the other hand does not consider this distinction, but rather concatenates them before pushing them through a neural network, thus loosing valuable information on their conditional relationship. Secondly, the NP objective is constrained to Gaussian output conditioned on the latent variable. In contrast, our method is able to deal with any modality of distributions as we consider a nonparametric model for the density.

2.5 Experiments

2.5.1 Experiments on Synthetic Data

To assess the ability of our method in learning the multimodality and heteroscedasticity in the response variable, we construct a synthetic dataset as follows: we sample $y_i \sim \text{Uniform}(0, 1)$ and set $x_i = \cos(ay_i + b) + \epsilon_i$, where a and b vary between tasks, with noise $\epsilon_i \sim \mathcal{N}(0, \sigma^2)$ (See Appendix for examples of tasks). This corresponds to a 90 degree rotated cosine curve with noise, resulting in multimodality of $p(y|x)$.

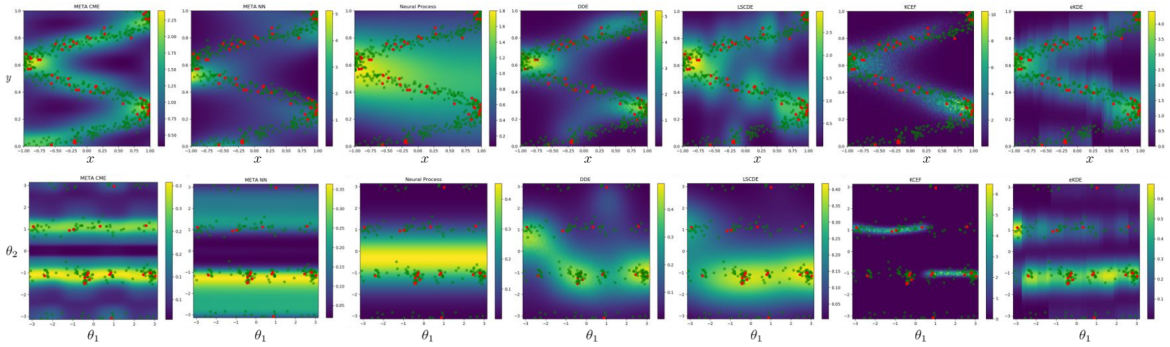

Figure 2.2: Density plots of synthetic data (top) and Chem data (bottom). MetaCDE(ours), MetaNN(ours), NP, DDE, LSCDE, KCEF, ϵ-KDE. Red points: context points. Green points: true density.

Figure 2.2 shows the comparison between our method and a number of alternative conditional density estimations methods, including ϵ-KDE (KDE applied to the ϵ-neighbourhood of x), DDE [Dai et al., 2018], KCEF [Arbel and Gretton, 2017], and LSCDE [Sugiyama et al., 2010]. We also include meta-learning algorithms such as the Neural Process [Garnelo et al., 2018b] and a pure neural network version of our framework, MetaNN.

For the meta-learning algorithms, we use 50 context points and 80 target points for each task. We also fix $\kappa = 10$ as suggested in [Gutmann and Hyvärinen, 2012] for all our experiments. At testing time, we evaluate the method on 100 new tasks with 50 context points each. We simply pass the new context points to our model, which can evaluate the density with a simple forward pass as in Eq.2.4. The non meta-learning baselines are trained on each of the 100 datasets separately. We have included additional experiments with different dataset size, as well as details on the neural network architectures, hyper-parameters and experimental setup for our experiments in the Appendix.

In Table 2.1, we report the mean loglikelihood over the 100 different datasets. Note that the NCE does not give us a perfectly normalized density. To ensure a fair comparison between methods, we post-normalize the densities for all our experiments, i.e. whenever we compute a conditional density, $p(y^*|X = x)$, we create a grid, $\{y_i\}_{i=1}^{100}$, equally spaced evaluations $p(y_i|X = x)$ over the range of the data, and normalize the density to 1 before computing the loglikelihood at y^*. We should also note that the post-normalization is minimal, see Appendix for more details.

The reason for the high variance in some methods stems from the varying difficulty of tasks. Hence, we also report the p-values of the one-sided signed Wilcoxon test. This allows us to confirm that our likelihood of MetaCDE is statistically significantly higher than all the other methods, including meta-learning methods such as NP and MetaNN.

2.5.2 Experiments on NYC taxi data

Next, we illustrate our algorithm on real world data such as the NYC taxi dataset from January 2016[2]. We are interested in estimating conditional densities $p_{\text{pickup}}(\text{dropoff}|\text{tips})$. Hence, for each task, i.e. given the pick up location, our goal is to model the dropoff density conditionally on the tip amount. Different tasks correspond to different pickup locations and thus different conditional densities.

During training, we use 200 context and 300 target points. At testing time, we are given 200 datapoints of dropoff locations for an unseen pickup location and model the conditional density based on those.

In Appendix we added figures to show how the density evolves as the tip amount increases. In particular, we see that given a pickup location in Brooklyn, as the tip amount increases the trips are more likely to end in Manhattan. Note that this pickup location has not been seen during training. We illustrate additional unseen pickup locations in the Appendix as well as additional information on the experimental setup. We compute the loglikelihood on 50 unseen pickup location and perform a one-sided Wilcoxon test as in the above section (see Table 2.1).

2.5.3 Experiments on Ramachandran plots for molecules

Lastly, we apply our algorithm on a challenging chemistry dataset. Finding all energetically favourable conformations for flexible molecular structures in both bound and unbound state is one of the biggest challenges in computational chemistry [Hawkins, 2017] as the number of possibilities increases exponentially with the dimension. The distribution of a pair of dihedral angles in molecules can be represented by *Ramachandran plots* [Mardia, 2013], and the knowledge of the correlated dihedral angles is limited by the library curated by the chemists. Here, we attempt to apply MetaCDE in order to learn richer relationships between dihedral angles, which can improve the current sampling scheme.

[2]Data from: `https://www1.nyc.gov/site/tlc/about/tlc-trip-record-data.page`

Our experimental data was extracted from crystallography database [Gražulis et al., 2011]. A task T_q consists of a pair of dihedral angles defined in a molecular fragment q, which is composed of 2 smaller fragments, F1 and F2. In this case, we have $\mathcal{D}^q = \{(\theta^q_{1,i}, \theta^q_{2,i})_{i=1}^{m_q}\}$, where $\theta^q_{1,i}$'s are the dihedral angle of F1 and $\theta^q_{2,i}$'s are the corresponding dihedral angles of F2. For more clarification, see Appendix. The multimodality arises from the molecular reflectional and rotational symmetry. Even though this dataset is a 1D problem it constitutes an important problem in chemistry and we show that conventional methods are unable to capture the true underlying density of the Ramachandran plots.

In our meta-learning setup we use 20 context and 60 target points. At testing time we are given 20 datapoints. We again fix $\kappa = 10$ as suggested in [Gutmann and Hyvärinen, 2012] and evaluate our method using loglikelihood on 100 examples of pairs of dihedral angles, which are unseen during training. We perform a one-sided signed Wilcoxon test to confirm that MetaCDE achieves a significantly higher held-out mean loglikelihood than other methods, as presented in Table 2.1. Figure 2.2 also shows that our method is able to successfully capture the multimodality in the data (i.e. 2 parallel lines), whereas other methods fail on this task, by either modelling the mean or focusing only on one mode. For more patterns see Appendix.

2.6 Conclusions and Future Work

We introduced a novel method for conditional density estimation in a meta-learning setting. We applied our method to a variety of synthetic and real-world data, with strong performance on applications in computational chemistry and NYC taxi data. Owing to the meta-learning framework, the experiments indicate that the developed method is able to capture correct density structure even when presented with small sample sizes at testing time. Similarly to the Neural Process [Garnelo et al., 2018b], our method is able to construct a task embedding. In our case however, the embedding of each task takes the form of a conditional mean embedding operator, computed with feature maps learned using noise contrastive estimation.

In this work we only considered low-dimensional problems (1D and 2D) just like in NP [Garnelo et al., 2018b,a], mainly because even in this setting, standard method still fail. We have demonstrated significant advantages of our method MetaCDE against the competition in important real-world applications such as the Ramachandran

plots in chemistry. One weakness of the proposed method is that it would not scale well in the dimension of y given the post-normalization/NCE needed. We leave this to future work to amend.

Lastly, an interesting avenue would be in modelling conditional distributions in the reinforcement learning setting. In particular, [Lyle et al., 2019] have shown the benefits of using distributional perspective on reinforcement learning as opposed to only modelling expectations of returns received by the agents.

2.7 Acknowledgments

We would like to thanks Anthony Caterini, Qinyi Zhang, Emilien Dupont, David Rindt, Robert Hu, Leon Law, Jin Xu, Edwin Fong and Kaspar Martens for helpful discussions and feedback. JFT is supported by the EPSRC and MRC through the OxWaSP CDT programme (EP/L016710/1). YWT and DS are supported in part by Tencent AI Lab and DS is supported in part by the Alan Turing Institute (EP/N510129/1). YWT's research leading to these results has received funding from the European Research Council under the European Union's Seventh Framework Programme (FP7/2007-2013) ERC grant agreement no. 617071.

Appendix

2.A Synthetic dataset setup and further experiments

In order to measure the ability of our method metaCDE to learn multimodality and heteroscedasticity in the response variable, we construct the datasets as follows: we sample $y_i \sim \text{Uniform}(0, 1)$ and set $x_i = \cos(ay_i + b) + \epsilon_i$, where a and b vary between tasks, with noise $\epsilon_i \sim \mathcal{N}(0, \sigma^2)$. Here we sample a from $U(8, 12)$ and phase vary from $U(0, \pi)$. Intuitively, this just corresponds to rotated sine curves as can be seen in the examples in Section 2.E of the Appendix. (see most left column for best density estimate of the true density.)

In this experiment we are given a variable number of context points during testing time ranging from $15, 30$ and 50. This is done to investigate the ability of metaCDE to adapt even in small dataset situations.

MetaCDE/Neural Process/MetaNN are trained with $15, 30$ and 50 context points on the tasks and with 80 target points during training. At testing time, we simply pass the data through our model without having to retrain on the new unseen dataset. Note that we report again the p-values of the Wilcoxon signed one-sided test and we can see that as we decrease the context points, our methods is significantly outperforming the other methods. See Section 2.E of Appendix for additional

Each of the non meta learning models DDE, KCEF, ϵ-KDE, LSCDE are trained separately on each new dataset as they cannot share information between tasks.

2.A.1 Model specifications

For our MetaCDE we used a 3-hidden layer Neural Network with *tanh* activation functions and *Adam* optimizer for all of our feature maps ϕ_x, ϕ_y, b_θ. We cross-validate on held out dataset, over 32 and 64 hidden nodes per layer and $\lambda = 1.0, 0.1, 0.01$ for

the regularization parameter. We fixed the learning rate at 1e-3. We also set $\kappa = 10$ as suggested by Gutmann and Hyvärinen [2012].

- KCEF: we used the CV function that was in built in their Github repository (optimizing the parameters from a range $[1e\text{-}1, 10]$ `https://github.com/MichaelArbel/KCEF`) as well as consulted the authors to make sure we are using their method correctly.
- LSCDE: We CV for σ in logspace$(-3, 5, 20)$ and λ in logspace$(-5, 5, 20)$
- ϵ-KDE: We CV over ϵ in linspace$(0.1, 1, 15)$ and bandwidth in linspace$(0.01, 1, 15)$
- DDE: We CV over the bandwidth of 0.5 and 1.0

NOTE: We also tried to use a standard RBF kernel to compute our CMEO and CME with 50 context points on the synthetic data, however, these results were not up to par with what we got with NN. We searched for lengthscales over a range 0.01 to 10, however, the results were not comparable on synth data (84.65 +-21.15), see Table in section 2.E for comparison.

The reason for this is first of all, the stationarity of the Gaussian kernel and secondly, only having very little data to construct a good CMEO. Using NN and CMEs we were able to capture the density by training it across tasks and hence learning more efficient embeddings than a gaussian kernel could. In addition, deep kernels have recently shown impressive results [Liu et al., 2020] and hence using a NN as a feature map is well justified.

2.B Neural Network version of our method (MetaNN)

Furthermore, in order to investigate the importance of the CMEO task representation, we developed a purely neural version of our proposed method, which we call MetaNN. It differs from MetaCDE solely in the task representation. While MetaCDE uses kernel embeddings formalism to represents the task using CMEO calculated on the context points, MetaNN uses the DeepSets [Zaheer et al., 2017] approach, where the context pairs (x_i, y_i) are simply concatenated into a vector, and then passed through a neural network. The outputs are then averaged to obtain the task embedding to which any new x_{target} is concatenated to obtain the "neural" equivalent to CME.

This neural representation is then being pushed through a Feed forward network in order to have the same dimension as $\phi_y(y_{target})$. This ensures that we can take the inner product to compute s_θ.

The same training procedure as MetaCDE follows. We note that the concatenation of x and y encodes the joint distribution, rather than the conditional as in MetaCDE. While such task representation does preserve the relevant information, it is susceptible to changes in the marginal of x across tasks and conditional representations are intuitively better suited for the task of conditional density estimation. The experiments demonstrate the value of combining the CME formalism with neural representations and we obtain significantly better results with MetaCDE compared to MetaNN. In our experiments we simply used a 3 layer MLP and cross-validate over either 32, 64 hidden nodes and learning rates of 1e-3 or 1e-4.

2.B.1 Comparison of MetaNN to MetaCDE

Next we will compare the MetaNN and MetaCDE algorithms in order to investigate the importance of the conditional mean embedding operator. As mentioned above, we have now swapped out the computation of the CMEO for a NN for MeatNN. This representation is then concatenated with the new x_{target} to give us an element in $\mathcal{H}_Y$. We test out MetaNN on the synthetic dataset described in the experimental section.

MetaNN will perform worse on 50 context points as show in the main text but better on 15, where MetaNN achieves a log-likelihood of 114.43 ± 26.44, whereas MetaCDE achieved only 51.73 ± 10.43 (see Figure 2.B.1). This can be explained by two factors.

Firstly, a lower number of context points might give us a worse estimate of the conditional mean embedding operator. Secondly, we note that it takes significantly more task examples for the MetaNN to achieve the performance and hence this might have been due to the limited variety in the training task *i.e.* variation in the range of the period and phase parameter. Hence we conjectured that MetaNN might have just memorized the tasks well.

Therefore we have ran additional experiments to on a harder synthetic dataset where we now sample a from $U(4, 14)$ and phase vary from $U(-\pi, \pi)$ (i.e. more variety in the tasks than before where we sampled a from $U(8, 12)$ and phase from $U(0, \pi)$). In this case, MetaNN seems to completely fail and not able to learn anything useful at all. As the tasks in this case are more variable (see figure 2.B.2). Hence, we have not included MetaNN in the below figures when comparing with other conditional density methods.

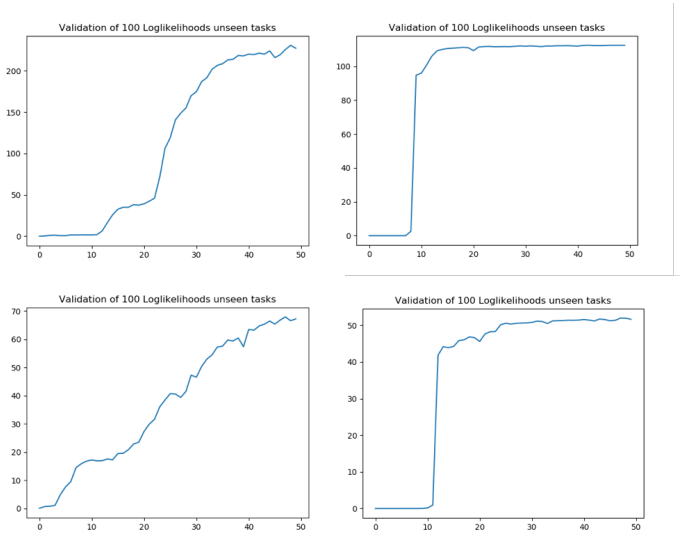

Figure 2.B.1: Figure illustrating how MetaNN performs better in low context points settings but seems to learn slower than MetaCDE. Top Row: 30 context points (left) MetaNN (right) MetaCDE; Bottom row: 15 context points (left) MetaNN (right) MetaCDE (x-axis represents 1 unit=10k tasks, y-axis loglikelihood)

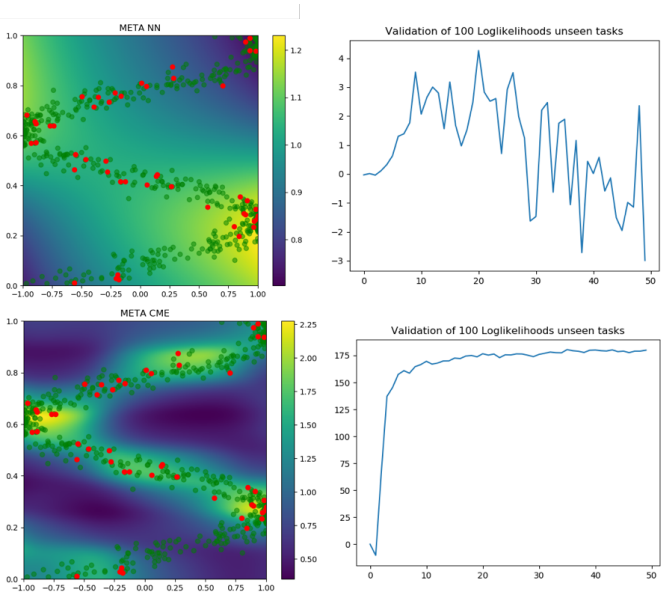

Figure 2.B.2: Figure illustrating how MetaNN fails when task become more variable. Top Row: MetaNN; Bottom row: MetaCDE (x-axis represents 1 unit=10k tasks, y-axis loglikelihood)

To further investigate this phenomenon, we have created a new task based on samples on Gaussian Processes (GPs). Here we sample 2 GPs with an Gaussian kernel with lengthscale 1 as well as a uniform random variable from $q \sim U(1, 3)$. We then added u to one of the sampled GPs and hence created a multimodal dataset in y (see figure 2.B.3). This task has a lot more variability than the previous synthetic dataset task. Below, we illustrate how MetaCDE is still able to perform well whereas MetaNN completely fails to learn anything useful. This illustrates that CMEO includes useful additional inductive biases in our model. In particular, by using a CMEO, we explicitly tell the model which entries are covariates and which are responses and that the relevant property of the data for this task is the conditional distribution.

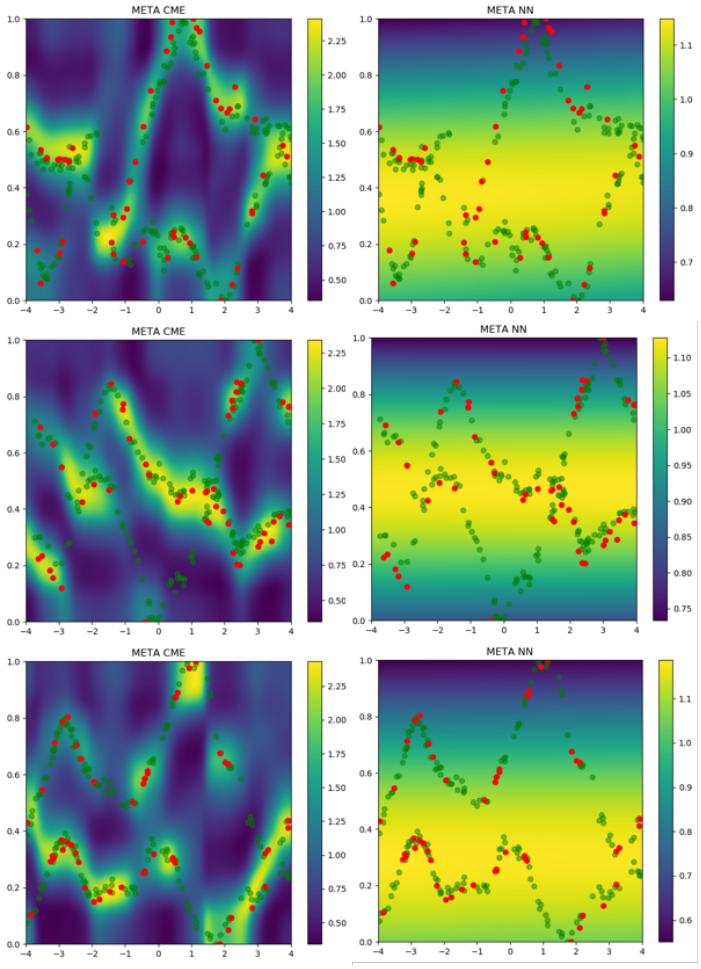

Figure 2.B.3: Density maps of the GP example (Left)MetaCDE (Right)(MetaNN)

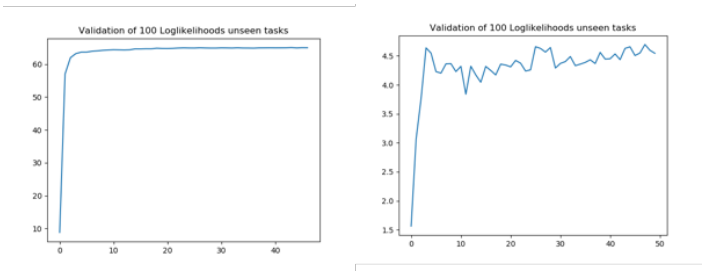

Figure 2.B.4: Evolution of the log-likelihood (x-axis represents 1 unit=10k tasks, y-axis loglikelihood) (Left)MetaCDE (Right)(MetaNN)

2.C Normalization network b_θ

First of all, we want to note that learning the b_θ is not a novel idea but in fact has been proposed by the seminal paper on noise contrastive estimation [Gutmann and Hyvärinen, 2012]. Gutmann and Hyvärinen [2012] note that one can actually learn the normalization constant and show empirically that that the learnt density is actually close to a properly normalized density. Intuitively, the reason this happens is as follows.

$$P_\theta(\text{True}|y,x) := \frac{p_\theta(y|x)}{p_\theta(y|x) + \kappa p_f(y)}. \tag{2.12}$$

Assuming that we have a Bayes optimal classifier above and that p_f is normalized (we choose this distribution so we know it is normalized), then assuming $p_\theta(y|x)$ was not normalized by a factor of γ then we could just divide numerator and denominator by γ. This would hence just correspond to modifying κ to κ/γ. However the Bayes optimal classifier sees exactly κ more fake samples then real ones and hence in the case Bayes optimality was reached, we should have γ close to 1, i.e. normalized $p_\theta(y|x)$. We even show in experiments that the normalziation needed is minimal (see Figure 2.C.1). Additionally, we also noted that using b_θ actually helped in terms of stability in our training as it gives an extra degree of flexibility.

Next, we note that $b_\theta(x)$ depends on $s_\theta(x,y)$ and hence is task/context set dependent. Modelling $b_\theta(x)$ as a neural network that only takes input x would not be task dependent, as it would be the same for each task and only depend on x. Therefore, we set the input of b_θ to be the CME $\widehat{\mu}_{Y=y|X=x}$, which thus incorporates all task information compactly (as $\widehat{\mu}_{Y=y|X=x}$ is computed using the CMEO). We have done further studies of this normalization network and show how it effectively normalizes the density approximately.

To ensure a fair comparison between methods, we post-normalize the densities for all our experiments, i.e. whenever we compute a conditional density, $p(y^*|X = x)$, we first create a grid, $\{y_i\}_{i=1}^{100}$, and consider equally spaced evaluations $p(y_i|X = x)$ over the range of the data, and use them to re-normalize the density model before evaluating performance by computing the loglikelihood at y^*.

We note that most methods are already producing density models that are very close to being normalized so the effect of the post-normalization is minimal.

Fig. 2.C.1 illustrates the post-normalization in our GP experiments (i.e. the closer the line is to one the less we need to normalize our density), demonstrating that our approaches are able to learn an approximately normalized density. We however still perform the post normalization in order to remain fair compared to other methods. Another thing to note is that including the information of the CME $b_\theta(x)$ compared to not using it (no CME), allows us to have even less normalization necessary when learning the density. We have also found that the normalization network helps greatly in keeping the learned density approximately normalized.

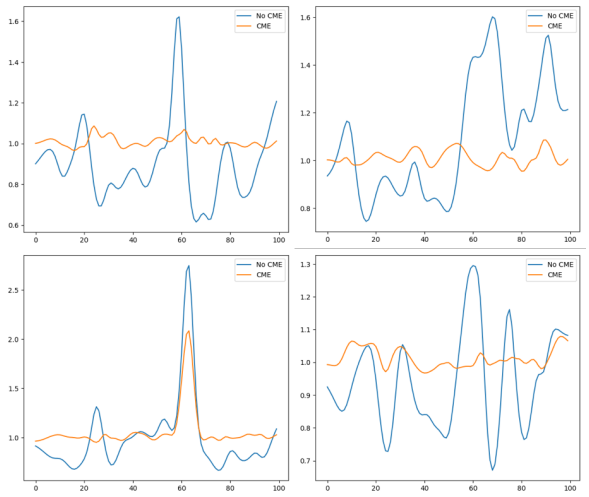

Figure 2.C.1: Post normalization needed after learning the density. Comparison shows the regimes where b_θ network includes CME as an input and the one where it does not.

2.D Choice of the Fake Distribution in NCE

The choice of the fake distribution plays a key role in the learning process here, especially because we are interested in conditional densities. In particular, if the fake density is significantly different from the marginal density $p(y)$, then our model could learn to distinguish between the fake and true samples of y simply by constructing a "good enough" model of the marginal density $p(y)$ on a given task while completely ignoring the dependence on x (this will then lead to feature maps that are constant in x).

This becomes obvious if, say, the supports of the fake and the true marginal distribution are disjoint, where clearly no information about x is needed to build a classifier – i.e. the classification problem is "too easy".

Thus, ideally we wish to draw fake samples from the true marginal $p(y)$ in a given task. While we could achieve this by drawing a y paired to another x, i.e. from the empirical distribution of pooled ys in a given task, recall that we also require to be able to compute the fake density pointwise. Hence, we propose to use a kernel density estimate (KDE) of y's as our fake density in any given task.

In particular, a kernel density estimator of $p(y)$ is computed on all responses y (context and target). To sample from the this fake density, we draw from the empirical distribution of pooled y's and add Gaussian noise with standard deviation being the bandwidth of the KDE (we are using a Gaussian KDE for simplicity here; other choices of kernels are of course possible with appropriate modification of the type of noise). As our experiments demonstrate, this choice ensures that the fake samples are sufficiently hard to distinguish from the true ones, requiring the model to learn meaningful feature maps which capture the dependence between x and y and are informative for the CDE task.

Furthermore, we want to note that we chose κ i.e. number of fakes samples to be 10, mainly because in the original noise contrastive paper, they use also used 10 and we noticed in our experiments that even when increasing κ the performance didn't increase by much and hence just fixed $\kappa = 10$.

Finally, we note that in principle it is possible to consider families of fake distributions which depend on the conditioning variable x. We do not explore this direction here and will leave this for future work.

2.E Illustration of Synthetic dataset

In this section we illustrate that our proposed method, metaCDE, performs well even when data becomes scarce. Hence we applied our method on several tasks, where we have 50, 30 and 15 datapoints as context. In each of the experiments we also plotted the corresponding density, the method gave us and we can clearly see that only metaCDE is able to recover the "rotated sine curve" (synthetic data experiments) in all 3 cases, which is also shown in the loglikelihood estimates.

2.E.1 Using 50 context points

	MetaCDE	NP	DDE	LSCDE	KCEF	ϵ-KDE
Sythn. Data Mean over 100 log-likelihoods	**197.84 ± 22.45**	-81.11±18.53	162.98 ± 69.01	44.95 ± 74.36	-388.30 ± 703.17	116.31 ± 236.99
Sythn. Data P-value for Wilcoxon test	NA	< 2.2e-16	8.144e-07	<2.2e-16	< 2.2e-16	2.384e-07

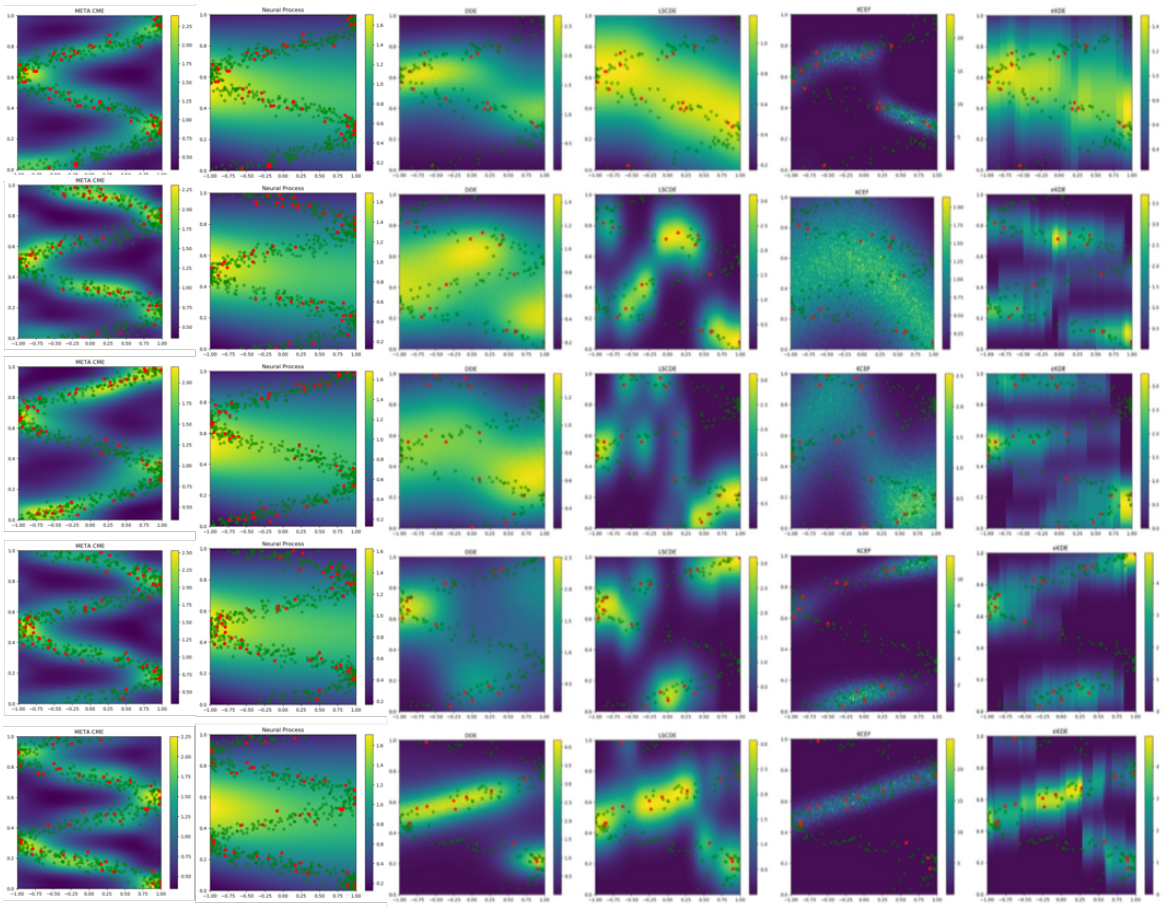

Figure 2.E.1: In Order (synthetic dataset): MetaCDE (ours), NP, DDE, LSCDE, KCEF, ϵ-KDE
The red dots are the context/training points and the green dots are points from the true density.

2.E.2 Using 30 context points

	MetaCDE	NP	DDE	LSCDE	KCEF	ϵ-KDE
Sythn. Data Mean over 100 log-likelihoods	**113.27 ± 17.36**	-48.98±12.26	64.61 ± 54.33	-23.02 ± 65.31	-233.38 ± 528.99	29.64 ± 195.49
Sythn. Data P-value for Wilcoxon test	NA	< 2.2e-16	4.577e-14	<2.2e-16	< 2.2e-16	4.917e-13

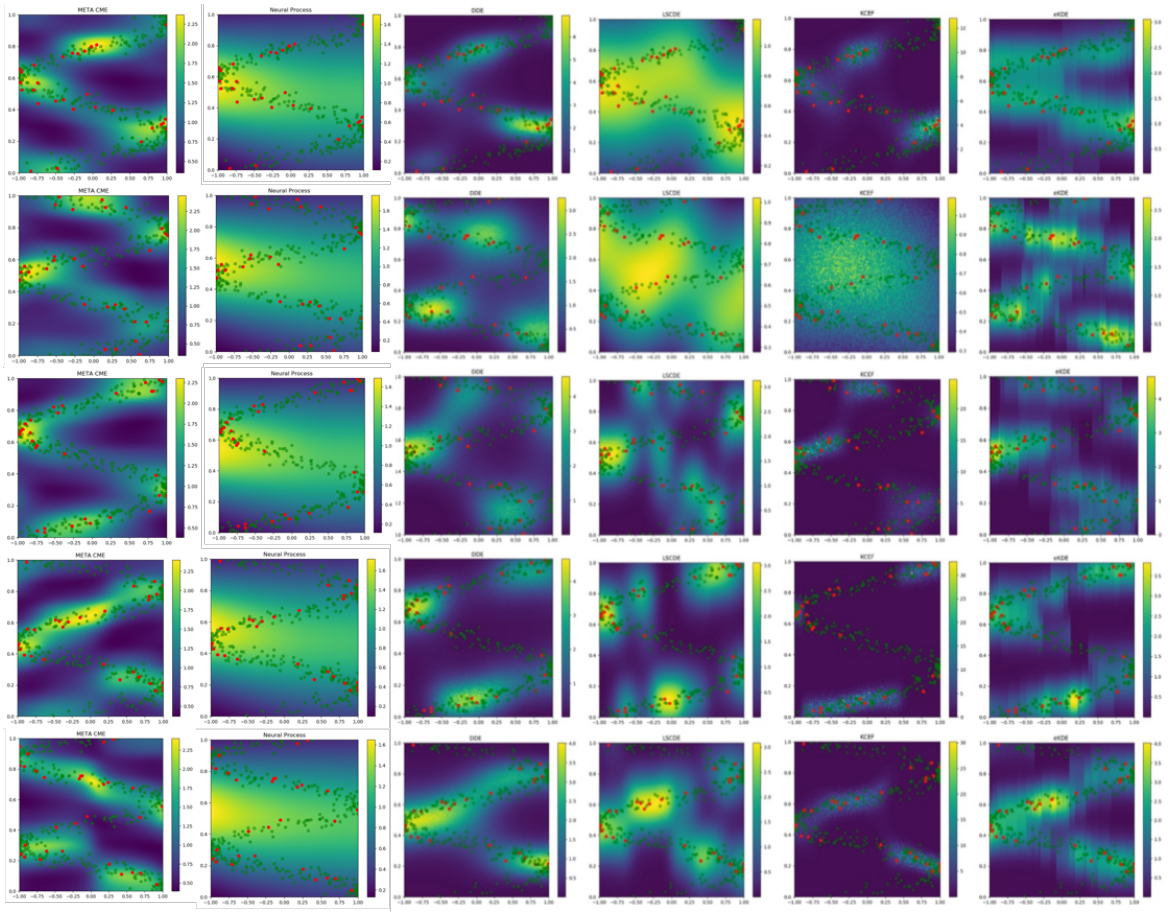

Figure 2.E.2: In Order (synthetic dataset): MetaCDE (ours), NP, DDE, LSCDE, KCEF, ϵ-KDE
The red dots are the context/training points and the green dots are points from the true density.

2.E.3 Using 15 context points

	MetaCDE	NP	DDE	LSCDE	KCEF	ϵ-KDE
Sythn. Data Mean over 100 log-likelihoods	**51.73 ± 10.48**	-24.39±8.20	0.58 ± 40.70	-57.99 ± 59.13	-142.19 ± 259.59	-87.50 ± 224.13
Sythn. Data P-value for Wilcoxon test	NA	< 2.2e-16	4.577e-14	<2.2e-16	< 2.2e-16	< 2.2e-16

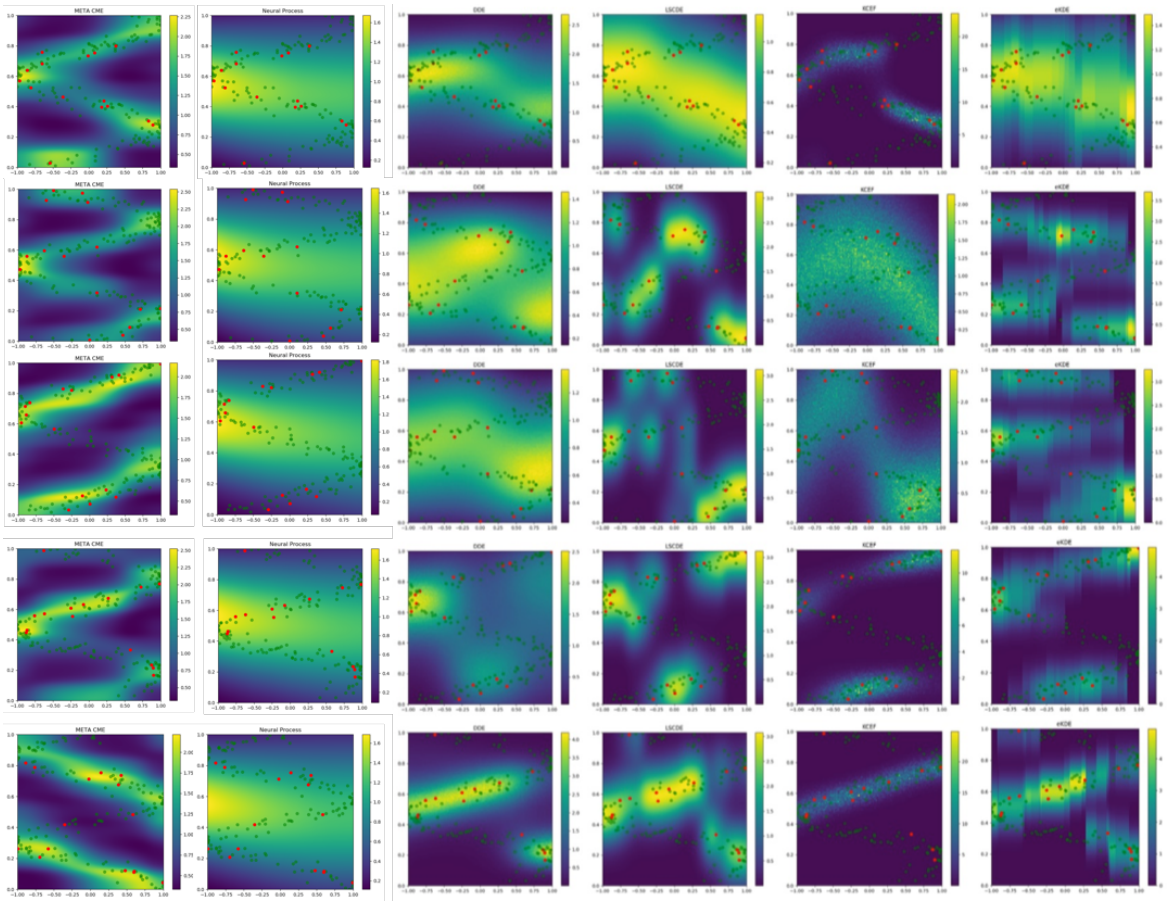

Figure 2.E.3: In Order (synthetic dataset): MetaCDE (ours), NP, DDE, LSCDE, KCEF, ϵ-KDE
The red dots are the context/training points and the green dots are points from the true density.

2.F Further insight to the Ramachandran plots

2.F.1 Further details on Ramachandran plots

To better understand the problem that we are tackling, consider the images below which represent different molecules with their respective fragments whose dihedral angle chemist measure. "*A dihedral angle is the angle between two intersecting planes. In chemistry, it is the angle between planes through two sets of three atoms, having two atoms in common. - Wikipedia*".

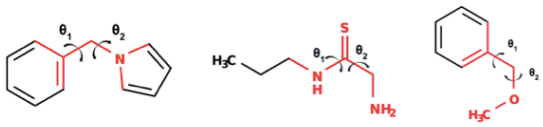

Figure 2.F.1: Example of molecules and the respective angles θ_1, θ_2 that we are trying to model. From this image it also becomes apparent how the multimodality arises as there are clearly some spatial symmetries. In the density plots below, the x-axis is θ_1 and the y-axis is θ_2

2.F.2 Additional information on the experimental setup

In this experiment, we look into the Ramachandran plots for molecules. Each plot indicates the energetically stable region of a pair of correlated dihedral angles in the molecule. Specifically, we are interested in estimating the distributions of these correlated dihedral angles. We compute the conditional density for each correlated dihedral angles, given 20 context points at testing time. For our meta-learning training we use 20 context points and 60 targets points.

Note that the data was extracted from crystallography database [Gražulis et al., 2011]. It is possible that some specific pairs of dihedral angles are rarely seen in the dataset, Hence, we may obtain a conditional density with high probability on the region without any observations in some cases. This is reasonable as the database covered only a small part of the chemical space and some potential area could be overlooked. Given that we assume that the support of our conditioning variable x ranges from $[-\pi, \pi)$, we will inevitable also compute conditional distribution on areas where the configurations are not defined and hence the densities in those areas can be safely ignored as a computational chemist would not have queried these configurations in the first place.

2.F.3 Model specifications

For our MetaCDE we used a 3 hidden layer NN with *tanh* activation functions for all of our feature maps. We cross validate over 32 and 64 hidden nodes per layer and $\lambda = 1.0, 0.1$ for the regularization parameter and over $1.0, 0.5, 0.3$ for the KDE lengthscale. We fix the learning rate at $1e$-3 and set $\kappa = 10$.

- KCEF: we used the CV function that was in built in their Github repository (optimizing the parameters from a range $[1e-1, 10]$ `https://github.com/MichaelArbel/KCEF`)
- LSCDE: We CV for σ in logspace$(-3, 5, 20)$ and λ in logspace$(-5, 5, 20)$
- ϵ-KDE: We CV over ϵ in linspace$(0.5, 3, 15)$ and bandwidth in linspace$(0.01, 3, 15)$
- DDE: We CV over bandwidth of 0.5 and 1.0

NOTE: Furthermore it looks like our method is not able to always capture the true trend given the limited amount of data. However, it seems to be able to capture some interesting **multimodal** patterns that would be useful to scientist to include in their models. Recently, there has been work done on these Ramachandran plots for Molecules by handcrafting the density maps. Our model allows us to compute the density maps without prior knowledge.

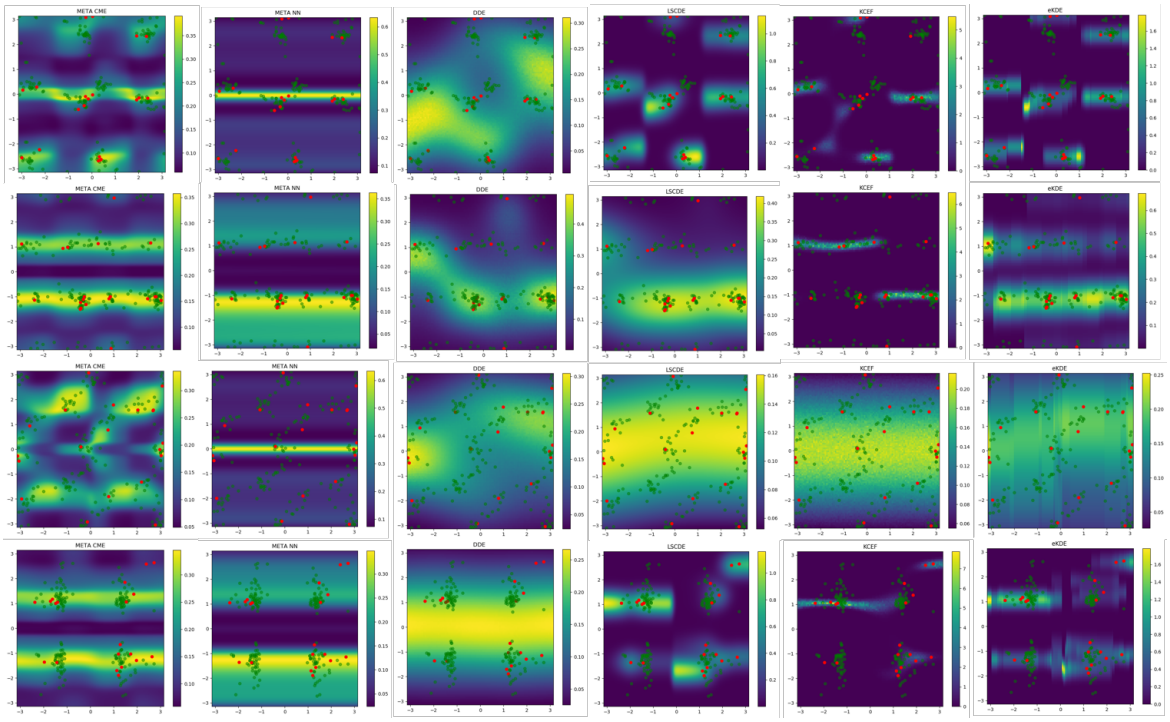

Figure 2.F.2: In Order (synthetic dataset): MetaCDE (ours), MetaNN (ours), NP, DDE, LSCDE, KCEF, ϵ-KDE
The red dots are the context/training points and the green dots are points from the true density.

2.G Illustration of the NYC taxi dataset

2.G.1 Experimental Setup

We have extracted the publicly available dataset from the website [3]. We have first of all restricted ourselves to drop-off locations in from -74.1 to -73.7 in longitude and 40.6 to 40.9 in latitude. Next we have given our meta learning model 200 datapoints for context during training and 300 for target. At testing time we are presented with 200 context points and are required to compute the conditional density given a tip. In this case each task is one specific pickup location. Again, we are using a 3-hidden layer NN and CV over $32, 64, 128$ nodes and $\lambda = 0.1$ or 1.0 and over $1.0, 0.5, 0.3$ for the KDE lengthscale. We use the *Adam* optimizer and fixed the learning rate to $1e-3$. We also set $\kappa = 10$.

2.G.2 Note on the dataset

In the main text we have seen how the drop-off density changes as we increase the amount of tips. This move of density illustrates well the data itself, as one is more likely to pay higher tips for longer journeys. Below we have plotted the drop-off locations of one specific pickup location colored with the respective tips paid.

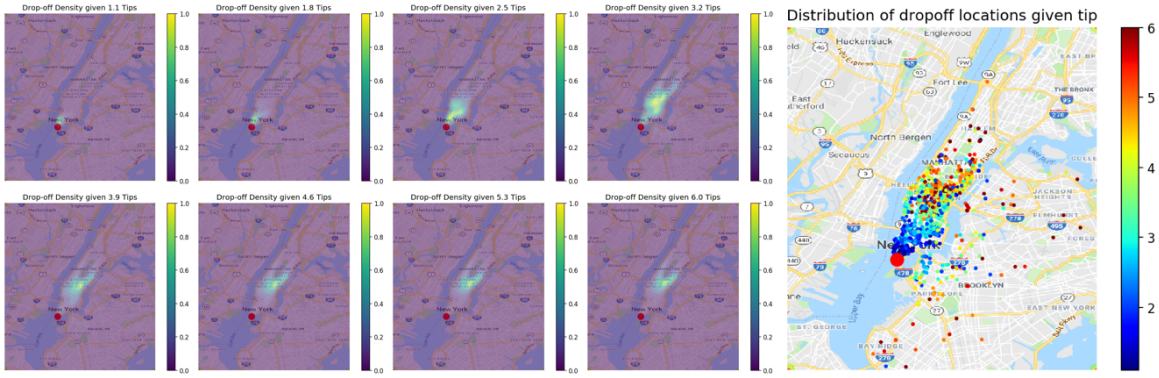

Figure 2.G.1: Density of the dropoff locations with increase in tips (right) the context and target points corresponding to the pickup locations (Big red dot the the pickup location)

[3]Data has been taken from: https://www1.nyc.gov/site/tlc/about/tlc-trip-record-data.page

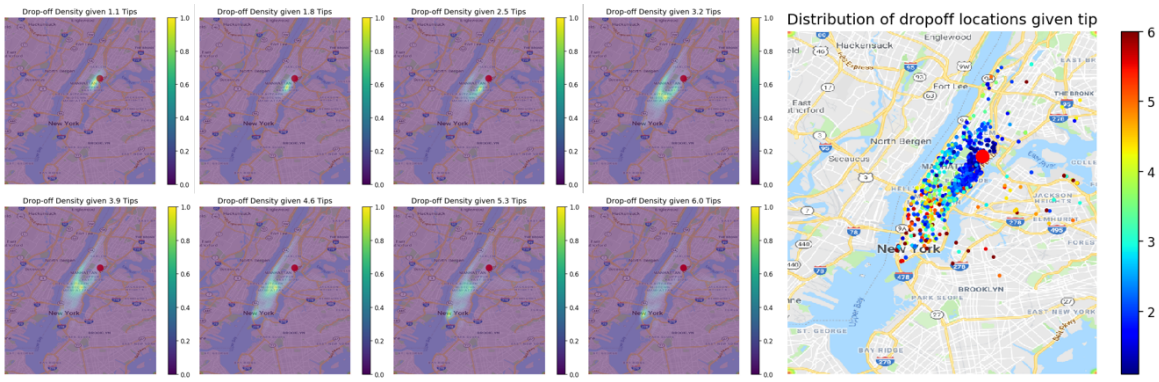

Figure 2.G.2: Density of the dropoff locations with increase in tips (right) the context and target points corresponding to the pickup locations (Big red dot the the pickup location)

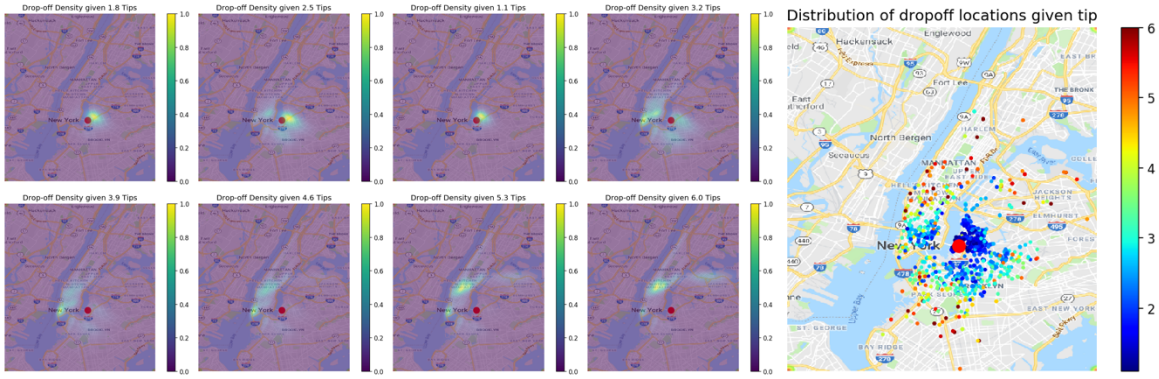

Figure 2.G.3: Density of the dropoff locations with increase in tips (right) the context and target points corresponding to the pickup locations (Big red dot the the pickup location)

Statement of Authorship for joint/multi-authored papers for PGR thesis

To appear at the end of each thesis chapter submitted as an article/paper

The statement shall describe the candidate's and co-authors' independent research contributions in the thesis publications. For each publication there should exist a complete statement that is to be filled out and signed by the candidate and supervisor **(only required where there isn't already a statement of contribution within the paper itself).**

Title of Paper	Noise Contrastive Meta Learning For Conditional Density Estimation using Kernel Mean Embeddings
Publication Status	xPublished □ Accepted for Publication □Submitted for Publication □Unpublished and unsubmitted work written in a manuscript style
Publication Details	***Jean-Francois Ton, Lucian CHAN, Yee Whye Teh, Dino Sejdinovic*** *Proceedings of The 24th International Conference on Artificial Intelligence and Statistics*, PMLR 130:1099-1107, 2021.

Student Confirmation

Student Name:	Jean-Francois Ton		
Contribution to the Paper	In this work, I was responsible for all the experiments. Together with DS and YW we discussed potential avenues of how to extend the Neural Process [1]. We then concluded that a Noise contrastive method with Kernel Mean embeddings would be an appropriate method to use after I mentioned how one could use the mean embedding operator as a task embedding. Lucian Chan provided the chemistry dataset. Both my supervisors also helped with the write up. [1] Neural Processes Garnelo et al. 2018		
Signature		Date	30/03/2022

Supervisor Confirmation

By signing the Statement of Authorship, you are certifying that the candidate made a substantial contribution to the publication, and that the description described above is accurate.

Supervisor name and title:	Professor Dino Sejdinovic		
Supervisor comments	All above seems accurate and fair.		
Signature		Date	30/03/2022

This completed form should be included in the thesis, at the end of the relevant chapter.

3

Meta Learning for Causal Direction

Abstract

The inaccessibility of controlled randomized trials due to inherent constraints in many fields of science has been a fundamental issue in causal inference. In this paper, we focus on distinguishing the cause from effect in the bivariate setting under limited observational data. Based on recent developments in meta learning as well as in causal inference, we introduce a novel generative model that allows distinguishing cause and effect in the small data setting. Using a learnt task variable that contains distributional information of each dataset (task), we propose an end-to-end algorithm that makes use of similar training datasets at test time. We demonstrate our method on various synthetic as well as real-world data and show that it is able to maintain high accuracy in detecting directions across varying dataset sizes.

3.1 Introduction

Discovering causal links between variables has been a long standing problem in many areas of science. Ideally all experiments are in a randomized controlled environment where each variable can be accounted for separately. However in most cases, this is impossible due to physical, financial or ethical reasons. The problem of determining causal direction becomes even more apparent, when trying to understand the true generating process of data. Although modern machine learning models are able to achieve impressive performances in capturing complex nonlinear relationships among variables, many of them do not take into account causal structure, which might lead to generalization errors when faced with different data than seen during training.

Hence, in recent years, researchers have focused on inferring causal relations from observational data and have developed many different algorithms. One of the major approaches is constraint-based methods, which analyze the conditional independence among variables to determine the causal graph up to a Markov equivalence class under certain assumptions [Neuberg, 2003]; in addition to early example of the PC algorithm [Spirtes et al., 2000], there are also nonlinear methods for capturing independence [Sun et al., 2007b,a, Zhang et al., 2011].

Another category is score-based methods which use search algorithms to find the best causal graph with respect to such a score as BIC [Chickering, 2002]. These methods are, however, often unable to determine the correct structure and can be computationally very expensive. There are also hybrid methods which mitigate such difficulty [Tsamardinos et al., 2006].

A new line of research has taken specific interest in the bivariate case, i.e., the cause-effect inference, where one decides between causal hypotheses "$X \to Y$" and "$Y \to X$" [Hoyer et al., 2009, Goudet et al., 2017, Mitrovic et al., 2018, Wu and Fukumizu, 2020]. In this setting, methods that exploit the inherent asymmetries between cause and effect are the most prominent. The data is analysed under the *Functional Causal Model* (FCM, [Pearl, 2009]) formalism, following respective model assumptions. Due to the intrinsic asymmetry of the problem, several statistics have been proposed to infer the direction of the cause-effect pairs [Shimizu et al., 2006, Hoyer et al., 2009, Mooij et al., 2016, Mitrovic et al., 2018].

Most relevant prior work to this paper is the framework of Causal Generative Neural Networks (CGNN, [Goudet et al., 2017]). When applied to cause-effect inference, CGNN

learns a generative model using a neural network for each direction and compares their fitting to determine the causal directionality. Many advanced methods including CGNN, however, assume an access to a large dataset to make use of strong learning models such as neural networks. This in turn may lead significantly degraded performances in small data settings encountered in practical problems [Look and Riedelbauch, 2018]. We demonstrate this phenomenon in our experimental section. In addition, training a model for every dataset may cause significantly slow inference time by the computational burden of neural networks.

In this paper, we revisit the problem of cause-effect inference from the viewpoint of empirical learning. Specifically, we are interested in learning from many examples of cause-effect datasets together with their true causal directions. Our purpose is to develop a method for using this empirical knowledge on causal directions effectively, when making cause-effect inference on a new unseen and purely observational dataset.

Learning-based cause-effect inference has been already explored in the literature. For instance, Randomized Causation Coefficient (RCC) [Lopez-Paz et al., 2015a] and Neural Causation Coefficient (NCC) [Lopez-Paz et al., 2017] make a learning-based binary classifier for causal directions. RCC and NCC, however, require to synthesize vast amounts of problem specific data pairs up to $10,000$ [Lopez-Paz et al., 2017]. Other examples, such as NonSENS [Monti et al., 2019] and Causal Mosaic [Wu and Fukumizu, 2020], aim to recover a FCM using nonlinear independent component analysis. An important assumption for these methods is availability of multiple datasets sharing the same causal mechanism and the same exponential family of latent variables. We then need to assume or select datasets to satisfy this.

Different from these works, aiming at alleviating the problem of small data, we consider methods of meta learning by introducing a dataset-feature extractor. The feature represents the distributional information of each dataset, aiming to encode similar causal mechanisms of datasets into similar features. For this purpose, we employ two approaches: the formalism of kernel mean embeddings [Muandet et al., 2017] and DeepSets [Zaheer et al., 2017].

We propose a neural network-based generative model that trains jointly on all the training datasets. The model has an encoder-decoder architecture, which has been employed successfully by meta learning frameworks [Garnelo et al., 2018b,a]; the encoder gives the dataset-features, and the decoder realizes a generative model or FCM, which is accompanied with the Feature-wise Linear Modulation layers (FiLM [Perez

et al., 2018]) to adapt the generator to the dataset at hand. With this meta-learning architecture, the proposed method is able to determine the cause-effect direction efficiently for new unseen, possibly small, and purely observational datasets.

The contributions of this work can be summarized as follows:

- We introduce a new meta learning algorithm that can leverage similar datasets for unseen causal pairs in causal direction discovery.
- We exploit structural asymmetries with an adaptive generative model, thus avoiding the need to retrain at test time.
- High performance on small dataset sizes can be achieved by virtue of meta learning.

3.2 Meta Learning for Detecting Causal Direction

We first give a brief summary of FCM and explain the proposed method including its building blocks.

3.2.1 Functional Causal Model

Functional Causal Models (FCM) have been widely used when conducting causal inference. Formally, a FCM on a random vector $X = (X_1, \ldots, X_d)$ is a triplet $C = (\mathcal{G}, f, \mathcal{E})$, where $\mathcal{G}$ is the causal graph, f are the structural functions and $\mathcal{E}$ are the exogenous noise such that:

$$X_i = f_i(X_{Pa(i;\mathcal{G})}, Z_i), \quad Z_i \sim \mathcal{E}, \text{ for } i = 1, \ldots, d, \tag{3.1}$$

where X_i are the observed variables, Z_i are the exogenous variables, $Pa(i;\mathcal{G})$ being the parents of X_i and f_i being the mechanism linking the cause and the effect. Under this formulation, there is clear asymmetry between cause and effect, given that cause is used to infer the effect, and hence numerous work has been done exploiting this fact.

For inferring causal direction $X \to Y$ for bivariate (X, Y), we can consider only the FCM $Y = f(X, Z)$, where Z and X are independent. Among other inference methods, CGNN [Goudet et al., 2017] uses neural networks to train the mechanism f for a dataset. More precisely, given dataset $\mathcal{D} = \{(x_j, y_j)\}_{j=1}^m$, we generate Z_j by the standard normal distribution $N(0, 1)$, and train neural network f so that the

distribution of $\{(X_j, Y_j)\}_j$ be close to that of $\{(X_j, f(X_j, Z_j)\}_j$. The difference of the distributions is measured by the Maximum Mean Discrepancy (MMD, [Gretton et al., 2012]). CGNN learns two models $\hat{Y} = f_y(X, Z)$ and $\hat{X} = f_x(Y, Z)$, and chooses a better fit to determine the direction.

Unlike CGNN, which trains networks f_y and f_x for each dataset, our method considers a single neural network working for all the datasets in a cause-effect database $\{\mathcal{D}_i\}_{i=1}^N$ where $\mathcal{D}_i = \{(X_j^i, Y_j^i)\}_{j=1}^{m_i}$ is a dataset in the database. We assume that the causal direction is known for all $\mathcal{D}_i$ during training. More specifically, given $X^i \to Y^i$ is the true causal direction, we wish to create a single suitable model $F(X, Z)$ based on neural networks so that the distribution of $\mathcal{D}_i$ is approximately the same as that of $\{(X_j^i, \hat{Y}_j^i)\}$ for *any* i, where $\hat{Y}_j^i = F(X_j^i, Z_j^i)$.

This approach involves obvious difficulty, since a wide-variety of cause-effect relations must be learnt by a single network. Naïvely training a single model jointly over all the different dataset does not yield desired performance as we demonstrate in our ablation study. In order to achieve successful training, we introduce two novel and crucial components:

1. **Dataset-feature:** This feature C represents the causal mechanism of each dataset as the distributional information on the dataset $\mathcal{D}$. The feature will be used to adapt our single network efficiently at test time.

2. **FiLM layers:** To adapt the base neural network to each dataset using the dataset-features C, the FiLM layers enable us to adapt the weights of our network to a given new dataset $\mathcal{D}$ quickly at test time.

Together, with these two additional apparatus, we are able to train our model across datasets and therefore harness information from all the datasets together, instead of treating them independently as it is done for example in CGNN.

In the next section we will briefly give an high level overview of meta learning and then afterwards describe the ways we are able to capture distributional information of dataset $\mathcal{D}_i$, by leveraging, the well studied area of conditional mean embeddings (CME) [Song et al., 2013] as well as DeepSets [Zaheer et al., 2017].

3.2.2 Meta Learning

Meta learning is an ever growing area in machine learning as it allows a model to extract information from similar problems/datasets and use this prior information on new unseen datasets. This is achieved, by sharing statistical strength across several causal inference problems. Standard methods usually require a lot of prior information to be useful in small data setting, i.e. knowing it is a linear model etc. Meta learning however learns this prior information through the meta learning training phase, where during training the model is presented with several small datasets, that allows the model to perform well on new unseen dataset.

In our setting, we are interested in learning the causal direction on new unseen datasets, given a set of datasets where we know the causal direction. This allows us to make efficient use of the information across all the datasets, which is contrary to most cause-effect methods with the exception of (Lopez-Paz et al. [2015a,b], etc) that treat datasets independently and have to be retrained for each new dataset.

3.2.3 Dataset features via DeepSets

DeepSets [Zaheer et al., 2017] have been used as task embedding C_i in previous meta learning literature [Garnelo et al., 2018a,b, Xu et al., 2019]. Using a neural network $\phi_{x,y}$, the task embedding is defined by

$$C_i = \frac{1}{m_i} \sum_{j=1}^{m} \phi_{x,y}([x_j, y_j]). \tag{3.2}$$

DeepSets is a simple and flexible approach to encoding sets into vectors, which is also permutation invariant. The latter is important as we do not want the embeddings to change solely based on the order of the elements in the dataset. [Zaheer et al., 2017, Wagstaff et al., 2022] show that Deep Sets are universal approximators for continuous set functions if model's latent space is sufficiently high-dimensional. However, given that we use a concatenation of both x and y in DeepSets we do not encode the conditional distribution information but rather the joint. Therefore, in this chapter, we also consider conditional mean embeddings as dataset-features.

3.2.4 Dataset features via Conditional Mean Embeddings (CME)

Kernel mean embeddings of distributions provide a powerful framework for representing probability distributions [Song et al., 2013, Muandet et al., 2017]. Given sets $\mathcal{X}$ and $\mathcal{Y}$, with a distribution P over the random variables (X, Y) taking values in $\mathcal{X} \times \mathcal{Y}$, the conditional mean embedding (CME) of the conditional density $p(y|x)$, is defined as:

$$\mu_{Y|X=x} := \mathbb{E}_{Y|X=x}[\phi_y(Y)] = \int_{\mathcal{Y}} \phi_y(y)p(y|x)dy. \tag{3.3}$$

where ϕ_y is the feature map associated to the reproducing kernel Hilbert space (RKHS) of Y, $\mathcal{H}_Y$. Intuitively, the equation above allows us to represent a probability distribution $p(y|x)$ in a function space such as a RKHS, by taking the expectation under $p(y|x)$ of the features $\phi_y(y) \in \mathcal{H}_Y$. Hence, for each value of the conditioning variable x, we obtain $\mu_{Y|X=x} \in \mathcal{H}_Y$.

Following [Song et al., 2013], the CME can be associated with the operator $\mathcal{C}_{Y|X} : \mathcal{H}_X \to \mathcal{H}_Y$, known as the conditional mean embedding operator (CMEO), which satisfies

$$\mu_{Y|X=x} = \mathcal{C}_{Y|X}\phi_x(x) \tag{3.4}$$

where $\mathcal{C}_{Y|X} := \mathcal{C}_{YX}\mathcal{C}_{XX}^{-1}$ with $\mathcal{C}_{YX} := \mathbb{E}_{Y,X}[\phi_y(Y) \otimes \phi_x(X)]$ and $\mathcal{C}_{XX} := \mathbb{E}_{X,X}[\phi_x(X) \otimes \phi_x(X)]$.

As a result, the finite sample estimator of $\mathcal{C}_{Y|X}$ based on the dataset $\{(x_j, y_j)\}_{j=1}^n$ can be written as

$$\widehat{\mathcal{C}}_{Y|X} = \Phi_y(K + \lambda I)^{-1}\Phi_x^T \tag{3.5}$$

where $\Phi_y := (\phi_y(y_1), \ldots, \phi_y(y_n))$ and $\Phi_x := (\phi_x(x_1), \ldots, \phi_x(x_n))$ are the feature matrices, $K := \Phi_x^T\Phi_x$ is the kernel matrix with entries $K_{i,j} = k_x(x_i, x_j) := \langle \phi_x(x_i), \phi_x(x_j) \rangle$, and $\lambda > 0$ is a regularization parameter.

Hence $\widehat{\mu}_{Y|X=x} = \widehat{\mathcal{C}}_{Y|X}\phi_x(x)$ simplifies to a weighted sum of the feature maps of the observed points y_i:

$$\widehat{\mu}_{Y|X=x} = \sum_{i=1}^{n} \beta_i(x)\phi_y(y_i) = \Phi_y\beta(x), \tag{3.6}$$

$$\beta(x) = (\beta_1(x), \ldots, \beta_n(x))^T = (K + \lambda I)^{-1}K_{:x}, \tag{3.7}$$

where $K_{:x} = (k_x(x, x_1), \ldots, k_x(x, x_n))^T$.

In fact, when using finite-dimensional feature maps, the conditional mean embedding operator is simply a solution to a vector-valued ridge regression problem (regressing $\phi_y(y)$ to $\phi_x(x)$), which allows computation scaling linearly in the number of observations n.

The Woodbury matrix identity allows us to have computations of either order $\mathcal{O}(n^3)$ or $\mathcal{O}(d^3) + \mathcal{O}(d^2 n)$, where d is the dimension of the feature map ϕ_x. In our case, given that we are in the meta learning setting, the dataset size n is usually rather small and hence the CME can be efficiently computed.

The CME is a canonical way for capturing conditional densities and thus the mechanism in a functional causal model under certain data generating and identifiability assumptions. We give further motivations on why we use CMEO as dataset-features in the next few sections.

3.2.5 A Feature-wise Linear Modulation (FiLM)

We propose an architecture that is able to adapt the network $f(X, Z)$, i.e. a FCM, with the dataset-feature by using the Feature-wise Linear Modulation (FiLM) [Perez et al., 2018].

The FiLM layers are known to allow network adaptation to new environments quickly without adding further model parameters. They have been shown to work effectively in various tasks of computer vision [Perez et al., 2018] and regression [Requeima et al., 2019]. In essence, the FiLM layers work as follows: given a conditioning variable C (this may be the label for image classification) and l_a being the a^{th} layer of a network, the FiLM layer FL_a, constructed by a neural network, adapts l_a to l_a^{FL} by

$$(\beta_a, \gamma_a) = FL_a(C), \qquad l_a^{FL} = \beta_a + \gamma_a \circ l_a, \tag{3.8}$$

where $\circ$ is the element-wise multiplication. Intuitively, the FiLM layer learns shift and scale parameters, conditioned on C, for any given layer l_a. We shall use the CME or DeepSets for C.

3.2.6 Proposed method

We propose a new meta learning algorithm, *meta-CGNN*, which works for cause-effect inference given cause-effect training database $\mathbb{D} = \{\mathcal{D}_i\}_{i=1}^N$ with known directionality, where $\mathcal{D}_i = \{(x_j^i, y_j^i)\}_{j=1}^{m_i}$. Without loss of generality, we assume $X^i \to Y^i$ for any

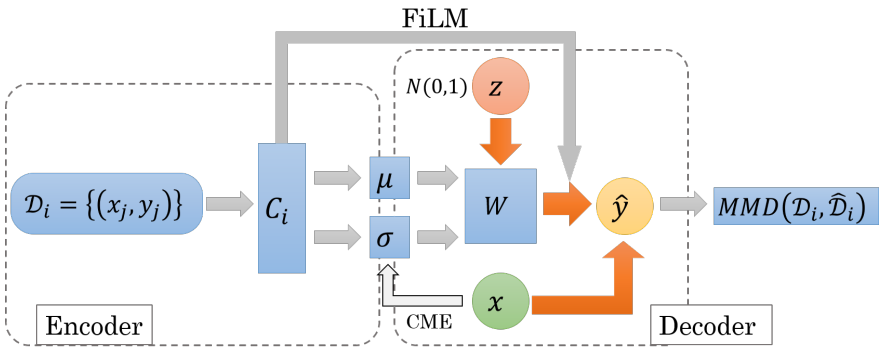

Figure 3.2.1: Proposed meta-CGNN Algorithm for only one dataset $\mathcal{D}_i$ in the mini-batch

dataset i. The FCM is trained on this database $\mathbb{D}$, and is used to infer the causal direction for an unseen dataset $\mathcal{D}_{test}$.

Overview: We use the popular encoder-decoder based architecture for meta learning, which is similar to the Neural Process [Garnelo et al., 2018b]. The encoder first maps the dataset $\mathcal{D}_i$ into a dataset-feature C_i, which is given as an input to two further neural networks, (1) FiLM network and (2) amortization network. The FiLM network, operates as described in the above, by producing shift and scale parameters that allow us to adapt the decoder D_{FL} accordingly. The amortization network outputs $(\mu(C), \sigma(C))$, with which we use to modulate the latent random variable $Z_j^i \sim N(0,1)$ to $W_j^i := \mu(C_i) + \sigma(C_i)Z_j^i$.

The decoder network, D_{FL}, then maps (X, W) to $\hat{Y}$ so that the distribution of $\{(X_j^i, \hat{Y}_j^i)\}_j$ is close to $\{(X_j^i, Y_j^i)\}_j$. By using an encoder network, which trains across datasets jointly, we are able to share distributional information between datasets (tasks) and apply it to a new unseen task. See Figure 3.2.1 and Algorithm 3 for a detailed breakdown of *meta-CGNN*.

The overall functional causal model in the proposed meta-CGNN is thus

$$\hat{y}_j = F\left((x_j, z_j); C\right) \text{ , where } z_j \sim N(0,1), \quad z_j \perp\!\!\!\perp x_j. \tag{3.9}$$

Encoder: For representing dataset-specific distributional information or mechanism for each task, we consider both the CME and DeepSets approach.

As argued by [Mitrovic et al., 2018], under identifiablity assumptions, CMEs can hold critical information for causal-effect inference by representing the mechanism in the FCM. The claim is that the Kolmogorov complexity [Grünwald et al., 2008] of the mechanism tends to be larger in the anti-causal direction than it is in the causal

direction. Hence, we expect CME to capture relevant distributional information, and inform adapting our generative model to the task at hand.

More concretely, we use conditional mean embeddings as follows. We first compute the CMEO $\mathcal{C}_{Y|X}$ from $\mathcal{D}_i$ as described in (Eq 3.5), and use it to obtain the CME for each datapoint using:

$$C_{i,j} = \mathcal{C}_{Y|X}\phi_x(x_j). \tag{3.10}$$

where, ϕ_x is the feature vector such that $k(x_i, x_k) = \langle \phi_x(x_j), \phi_x(x_k) \rangle$, k being the RBF kernel. To obtain a finite dimensional representation efficiently, we will use Random Fourier Features [Rahimi and Recht, 2008] to approximate the CME. Throughout the paper we will use 100 features as we work on problems with at most 1500 datapoints. According to [Li et al., 2019b], we need approximately $\sqrt{N}$, where N is the number of datapoints (see Algorithm 3). For DeepSets the dataset feature C_i is simply defined in (Eq 3.2).

Decoder: For the generative part of our model, the dataset-feature C_i or $C_{i,j}$ gives modulation through the FiLM and amortization network. Both FiLM and amortization networks take as input C_i (DeepSets) or $C_{i,j}$ (CME). The FiLM layer is able to adapt the weights of the decoder network depending on the distributional feature of a dataset. This is crucial for a single network to learn FCMs of all the datasets. Naïvely training a single network over multiple datasets resulted in poor performance (see ablation study). The amortization network works on $z \sim N(0, 1)$ similarly to FiLM. It can be interpreted as the adaption of latent Gaussian distribution; $p(w|C_i)$ with $W^i = \mu(C_i) + \sigma(C_i)Z$ regarded as a new latent variable for the dataset $\mathcal{D}_i$.

Together with the FiLM layer we construct a decoder which generates data from the conditional distribution, by firstly sampling from $p(w|\mathcal{D}_i)$ and concatenate w with x before pushing it through the decoder D_{FL}. This novel architecture, allows us to model and more importantly, **sample** from the conditional distribution of unseen task quickly and efficiently. In the next section, we will describe how we will make use of the samples to train our networks.

$$\hat{y}_j = D_{FL}\left((x_j, w_j); C\right) \text{ , where } w_j \sim N(\mu(C), \sigma(C)) \tag{3.11}$$

Training: The objective function of training is similar to [Goudet et al., 2017]; by sampling $\{\hat{y}_j^i\}_{j=1}^{m_i}$ from (3.11) and estimating the Maximum Mean Discrepancy (MMD)

[Gretton et al., 2012] between the sampled data $\widehat{\mathcal{D}_i} = \{(x_j^i, \hat{y}_j^i)\}_{j=1}^{m_i}$ and the original data $\mathcal{D}_i = \{(x_j^i, y_j^i)\}_{j=1}^{m}$. MMD is a popular metric to measure the distance between two distributions. It has been widely used in two-sample tests [Gretton et al., 2012] as well as in applications of Generative Adversarial Network (GAN) [Li et al., 2017].

More formally, the MMD estimator between two datasets $\mathcal{U} = \{u_i\}_{i=1}^m$ and $\mathcal{V} = \{v_i\}_{i=1}^n$ is, given by

$$\begin{aligned}\widehat{\mathrm{MMD}}^2(\mathcal{U}, \mathcal{V}) = &\frac{1}{m(m-1)} \sum_{i=1}^{m} \sum_{j \neq i}^{m} k(u_i, u_j) \\ &+ \frac{1}{n(n-1)} \sum_{i=1}^{n} \sum_{j \neq i}^{n} k(v_i, v_j) \\ &- \frac{2}{mn} \sum_{i=1}^{m} \sum_{j=1}^{n} k(u_i, v_j).\end{aligned}$$

We use Gaussian kernel, which is characteristic [Sriperumbudur et al., 2011a]. With k being the Gaussian kernel, the above expression is differentiable and thus can be optimized as already demonstrated in various works such as [Goudet et al., 2017, Li et al., 2017]. A drawback of using MMD as a loss function however is that it scales quadratically in the sample size. We can again use Random Fourier Features [Rahimi and Recht, 2008, Lopez-Paz et al., 2015a], which give us linear-time estimators of MMD.

Using stochastic mini-batches of q datasets $\{\mathcal{D}_i\}_{i=1}^q$, the objective function to minimize is

$$LOSS = \sum_{i=1}^{q} \widehat{\mathrm{MMD}}^2(\widehat{\mathcal{D}_i}, \mathcal{D}_i) \tag{3.12}$$

This joint training, similar to that used in other encoder-decoder architectures for meta learning [Garnelo et al., 2018a,b, Xu et al., 2019], allows us to utilize the information of all the available training datasets in a single generative model. This is in stark contrast to CGNN [Goudet et al., 2017], which trains a separate generative model for each dataset. As noted in [Goudet et al., 2017] training these neural networks separately can be very costly and one of the major drawbacks of the CGNN. Hence *meta-CGNN* aims to alleviate this constraint by training over datasets jointly. This considerably speeds up the model inference time as *meta-CGNN* does not need to retrain the network for a new task.

Inference of Causal Direction: After training, when we wish to infer the causal direction for a new dataset $\mathcal{D}_{test} = \{(x_j, y_j)\}_{j=1}^m$, we feed both of $\mathcal{D}^{xy} = \{(x_j, y_j)\}_{j=1}^m$

and $\mathcal{D}^{yx} = \{(y_j, x_j)\}_{j=1}^m$ into the trained model and estimate the MMDs between the generated samples and the true ones, i.e., $\mathcal{M}_{xy} := \widehat{\text{MMD}}(\mathcal{D}^{xy}, \hat{\mathcal{D}}^{x\hat{y}})$ and $\mathcal{M}_{yx} := \widehat{\text{MMD}}(\mathcal{D}^{yx}, \hat{\mathcal{D}}^{y\hat{x}})$. If $\mathcal{M}_{xy} < \mathcal{M}_{yx}$, we deduce that $X \to Y$ as it agrees better with a postulated FCM than $Y \to X$, and similarly $Y \to X$ if $\mathcal{M}_{xy} > \mathcal{M}_{yx}$. Intuitively, this means that we choose the direction of whose samples that match the ground truth best. Note that we assume that the supports over x and y are the same in our setting.

3.3 Related work

There have been already some works that exploit the asymmetry by taking a closer look at decomposing the joint distribution $P(X, Y)$ into either $P(Y|X)P(X)$ or $P(X|Y)P(Y)$. Relevant to this work is the approach using the asymmetry in terms of functional causal models (FCMs). Some of the previous methods make strong assumptions on the model; LiNGAM [Shimizu et al., 2006] considers linear non-Gaussian model for finding causal structure. For nonlinear relations, [Hoyer et al., 2009] discusses nonlinear additive noise models, and [Zhang and Hyvarinen, 2012] invertible interactions between the covariates and the noise models. There are other methods to consider the nonlinear models such as Gaussian process regression [Stegle et al., 2010].

Information theory gives an alternative view on asymmetry using Kolmogorov complexity, following the postulate that the mechanism in the causal direction should be less complex than the one in the anti-causal direction. Several papers have proposed to approximate or use certain proxies for intractable Kolmogorov complexity [Janzing and Scholkopf, 2010, Lemeire and Dirkx, 2006, Daniusis et al., 2012, Mitrovic et al., 2018].

The method that comes closest to ours is CGNN [Goudet et al., 2017]. However, our meta-CGNN method differs from CGNN in a multitude of aspects.

1. Our method employs meta learning, while CGNN only considers one dataset at a time. Hence CGNN is not able leverage similarity between datasets. A naïve way of training the CGNN jointly over datasets as in (Eq 3.12) was analysed in our ablation study and performed poorly.

2. CGNN averages over 32-64 separately trained generative networks per direction, which is computationally very expensive, i.e. training up to 128 models per dataset. In contrast, we merely train the model with four random initializations

Method	Inference Time
ANM	<1sec CPU
CDS	<1sec CPU
RECI	<1sec CPU
ICGI	<1sec CPU
RCC	<1sec CPU
CGNN	24 mins GPU
meta-CGNN (DeepSets)	<1min GPU
meta-CGNN (CME)	<1min GPU

Table 3.3.1: Time needed during inference i.e. to determine the causal direction for a new dataset at test time. *meta-CGNN* is a lot faster at inference time than CGNN, which is crucial, given that CGNN needs to be trained from scratch for every new dataset.

and average over the resulting MMDs for each dataset, thereby achieving similar results to CGNN for larger datasets and significantly better for smaller ones.

3. CGNN needs to be trained separately for every new dataset, which in practice can be slow as well as computationally expensive. *meta-CGNN* does not need to be trained for a new dataset and can give a causal direction through a simple forward pass at the test time. This is a crucial difference which allows much faster inference once a new dataset is presented to the model.

Lastly, there have recently been other applications of meta learning to causal inference. MetaCI [Sharma et al., 2019] uses a MAML-style[Finn et al., 2017] (optimization based) learner but mainly deals with counterfactual inference rather than causal directionality.

Bengio et al. [2019] also consider a meta learning method using FCM, based on the principle that the model assuming true causal direction can be "adapted faster" than the one assuming the anti-causal. However, their method is designed for different settings. Firstly, the test distribution comes from a perturbation of the training distribution, which is modeled by a known parametric family. The choice of the model is not trivial for continuous domains in real-world data. Secondly, for training of neural networks, they assume to have access to a large training dataset around 3000 for a single mechanism, which is different from our setting of small data. On another note [Ke et al., 2019] also considers meta learning in the causal setting however, they do not focus on the bivariate settings, which is our main objective.

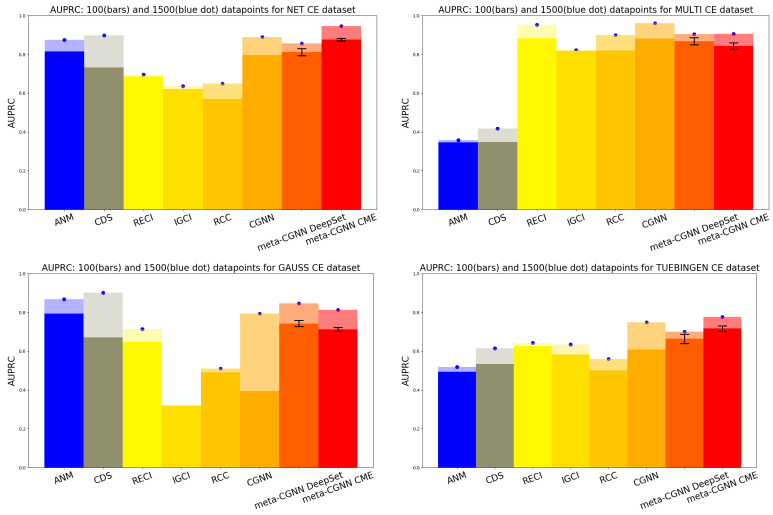

Figure 3.3.1: AUPRC for Net, Multi, Gauss and Tuebingen dataset. The thin colored bars with blue dots represent the AUPRC with 1500 datapoints, whereas the thicker barplots are with 100 datapoints. The proposed *meta-CGNN* shows among the best results in all the cases. Note that the other methods show significant degradation for some datasets or small data size.

Algorithm 3 *meta-CGNN* with CME

Input: $\mathcal{D} = \{\mathcal{D}_i\}_{i=1}^N$ Datasets w/ causal direction label $\mathcal{D}^{test} = \{\mathcal{D}^{test}\}$ Datasets w/o causal direction label

Output: Causal direction for $\mathcal{D}^{test}$

for $i \leftarrow 1$ to N **do**
  Compute CMEO: $\mathcal{C}_{Y|X}$ from $\mathcal{D}_i$ (Eq.3.5)
  for $j \leftarrow 1$ to m **do**
    Compute dataset-feature i.e $C_{i,j} = \mu_{Y|X=x_j^i}$
    $W_j^i := \mu(C_{i,j}) + \sigma(C_{i,j}) Z_j^i$, s.t. $Z_j^i \sim N(0,1)$
    $\hat{y}_j^i = D_{FL}\left((x_j^i, W_j^i); C\right) \rightarrow \widehat{\mathcal{D}}_i$ with FiLM dec.
  end for
end for
$LOSS = \sum_{i=1}^N \widehat{\text{MMD}}^2(\widehat{\mathcal{D}}_i, \mathcal{D}_i)$
Back-propagate through $LOSS$ until convergence
Compute $\mathcal{M}_{xy} := \widehat{\text{MMD}}(\mathcal{D}^{xy}, \hat{\mathcal{D}}^{x\hat{y}})$ and $\mathcal{M}_{yx} := \widehat{\text{MMD}}(\mathcal{D}^{yx}, \hat{\mathcal{D}}^{y\hat{x}})$
Check $\mathcal{M}_{xy} < \mathcal{M}_{yx}$ or $\mathcal{M}_{xy} > \mathcal{M}_{yx}$

3.4 Experiments

3.4.1 Synthetic Datasets

For the synthetic experiments we use three different types of datasets taken from [Goudet et al., 2017], each of which exhibits a distinct cause-effect (CE) mechanism. The *CE-Net* contains 300 cause-effect pairs with random distributions for the causes and random Neural Networks as mechanisms to create the effects. The *CE-Gauss* also contains 300 data pairs, where the cause is sampled from a random mixture of Gaussians and the mechanism is drawn from a Gaussian process [Mooij et al., 2016]. Lastly, *CE-Multi* datasets take the cause from Gaussian distributions and the mechanisms are built using random linear and polynomial functions. They also include multiplicative and additive noise before or after the causal mechanism, making the task harder. In addition, to confirm the advantage of meta learning, we use two different data size regimes: 1500 and 100 datapoints.

We measure the performance of distinguishing the causal direction using the Area Under the Precision Recall Curve (AUPRC), which is the same metric used in [Goudet et al., 2017] (accuracy is presented in the Supplementary material). With AUPRC, we are able to take into account the confidence of an algorithm, thus allowing models not to commit to a prediction if not certain.

For *meta-CGNN*, we use 100 datasets for training and the remaining 200 for testing. We average the MMD of each dataset over 4 independent runs in order to get our final prediction. At testing time, we only need to do a simple forward pass through our model, as our model has been trained to adapt to new dataset quickly and efficiently. Hence it takes **<1 minute** for each new dataset at inference time. This is contrary to CGNN that needs to be trained on each new dataset separately. Note that CGNN [Goudet et al., 2017] significantly benefits from averaging their model over multiple runs i.e. around 32-64 different runs, which takes about **24 minutes** per dataset. This would be infeasible without high-performance computing environments as we have hundreds of datasets to do inference on. Hence we have restricted ourselves to averaging CGNN over 12 runs in the comparisons, which is still computationally very heavy (See Table 3.3.1).

Regarding architectures, we use 2 hidden layers with ReLU activation function for the FiLM, amortization and encoder network. For the decoder we use a 1 hidden layer with ReLU activation function.

As noted in [Goudet et al., 2017], the number of hidden nodes in the decoder is very important; a network that is too small is not able to realize the mapping properly, while too big network tends to overfit so that both directions have low MMD values. Hence for our experiments, for simplicity, we solely cross-validated over the number of decoder nodes $[5, 40]$ [Goudet et al., 2017] by leaving out a few datasets at training aside for validation. Around 40 nodes for the 1-hidden layer was optimal for [Goudet et al., 2017]. See Supplements for further experimental details.

In addition, following [Goudet et al., 2017], for our loss function, we use a sum over Gaussian kernel MMDs with bandwidth $\eta \in \{0.005, 0.05, 0.25, 0.5, 1, 5, 50\}$ together with an Adam optimizer [Kingma and Ba, 2014] and a fixed learning rate 0.01. For the mini-batch size we fix q to 10. These were parameters we fixed in the beginning and seemed to work well in our experiments.

We compare *meta-CGNN* against several competing ones that have open source codes; the methods are

1. Additive noise model (ANM) [Mooij et al., 2016] with Gaussian process regression and HSIC test.

2. Information Geometric Causal Inference (IGCI) [Daniusis et al., 2012] with entropy estimator and Gaussian reference measure

3. Conditional Distribution Similarity statistic (CDS) [Fonollosa, 2019], which analyses the variance of the conditional distributions

4. Regression Error based Causal Inference (RECI) [Blöbaum et al., 2019], which analyses the residual of each direction using a polynomial fit

5. Randomized Causation Coefficient [Lopez-Paz et al., 2015a], which builds a synthetic classification problem and use CME as the feature.

6. Causal Generative Neural Networks (CGNN) [Goudet et al., 2017]

We use the implementation by [Kalainathan and Goudet, 2020] which provides a GitHub repository toolbox for the above mentioned methods. In order to keep the comparisons fair, we use the same 200 data pairs as the one that our *meta-CGNN* is tested on. Finally, in order to demonstrate the effect of each building block in our model, we have also conducted an ablation study, highlighting the importance

of the task embeddings and FiLM layer. Further experimental details can be found in the Supplementary Material.

#datapoints	Gauss		Multi		Net	
	FiLM	noFiLM	FiLM	noFiLM	FiLM	noFiLM
1500	**0.78**	0.73	**0.75**	0.46	**0.70**	0.55
500	**0.80**	0.72	**0.73**	0.49	**0.70**	0.50
100	**0.73**	0.64	**0.72**	0.63	**0.72**	0.50

Table 3.4.1: Accuracy of the proposed method with DeepSet embedding with and without the FiLM layer

3.4.2 Tuebingen Cause-Effect dataset

As a real-world example, we use the popular Tuebingen benchmark [Mooij et al., 2016], from which we take 99 bivariate datasets. We use a similar setup as in the synthetic experiments, and the only difference is that we employ 5-fold cross-validation for training and testing, i.e. We train on 4 of the folds and test on the last one. We repeat this procedure such that each fold has been the test fold at one point, while the remaining were acting as training. That way we obtain a prediction on each of the 99 datasets. We repeat this 3 times with different random splits and report the results in Figure 3.3.1, where we also check the performance on decreasing size of samples in a dataset.

3.4.3 Results

Figure 3.3.1 illustrates how both our *meta-CGNN* algorithms are the only ones that can retain high AUPRC across different datasets, as well as dataset sizes, i.e. 1500 and 100 datatpoints. **NOTE:** *meta-CGNN* performs well in all the small dataset settings, while remaining computationally more efficient at inference time, in contrast to CGNN, which has significantly worse result in the small data for *CE-Gauss* and *CE-Tueb* dataset and requires training the whole model at test time. Another important point to notice is that similarly to CGNN, *meta-CGNN* does not have any assumptions on the data and hence can be used in a variety of datasets, while retaining good performance. This does not hold for other methods.

Algorithms such as ANM [Mooij et al., 2016], for example, seems to do reasonably well on the *CE-Net* and *CE-Gauss* dataset, but completely fails in the *CE-Multi* and, more importantly, on the real-world Tuebingen dataset. This occurs mainly because

of the strict assumptions which ANM imposes on the FCM. Similarly, RECI which performs well on the *CE-Multi*, but not on the *CE-Net*, *CE-Gauss* and the Tuebingen dataset. Similar situation holds true for the remaining competing methods.

Our proposed method is amongst the top performing ones and is consistently doing well for both 1500 and 100 datapoints settings. These results illustrate that *meta-CGNN* is able to retain high performance, even when faced with small data, by leveraging the meta-learning setting. In order to understand the components of *meta-CGNN* better we conducted an ablation study in the next section.

Lastly, we want to emphasise that *meta-CGNN* does not need to be trained for each test dataset, but instead only needs a forward pass through the generative model to determine the causal direction, which makes it vastly more computationally efficient than CGNN, while attaining higher performance (see Table 3.3.1).

3.4.4 Ablation study

In this section, we study the necessity of task embedding in our model and show that not every model can be trained in this joint dataset fashion. To this end, we used a CGNN [Goudet et al., 2017] that has been trained exactly like our *meta-CGNN*, i.e. *meta-CGNN* where we removed the amortization network and the FiLM layer. In this case, only the decoder was trained jointly across the datasets and we see that this naïve version of extending CGNN does not perform well, with accuracy hovering around 55-60%, which are close to chance level, for the Tuebingen dataset with at most 1500 datapoints per dataset. It behaves similarly on the synthetic data with a **chance level accuracy** for *CE-Net* dataset with 1500 datapoints.

Next, we investigate the importance of the FiLM layer and run the experiments from our experiment section using DeepSets embeddings, with and without the FiLM layer while keeping the amortization network. We see from Table 3.B.1 a significant performance boost when using the FiLM as it allows the decoder at test time more flexibility to adapt, instead of relying solely on the amortization network. This shows the importance of each component in the proposed architecture and that adapting cause-effect methods into the meta learning setting is a non-trivial task. We see similar trends for *meta-CGNN* using CME. Lastly, we have also tried to ablate for the amortization network and noticed a similar/slightly higher performance with the amortization network in *meta-CGNN*.

3.5 Conclusion

We introduced a novel meta learning algorithm for cause-effect inference which performs well even in the small data regime, in sharp contrast to the existing methods. By leveraging a dataset-feature extractor that can be learned during training, we are able to efficiently adapt our model at test time to new previously unseen datasets by using amortization and FiLM layers. We also demonstrate the utility of using conditional mean embeddings, as they allow us to capture the distribution and adapt the model at test time.

In addition, we extended the framework of Causal Generative Neural Network (CGNN) [Goudet et al., 2017] by learning a single generative network, readily adaptable for new datasets, vastly alleviating the computation burden of CGNNs. In particular, instead of having to train multiple models on each dataset separately, our proposed methods are able to achieve similar or better performance than existing methods with simple forward passes through our generative network at test time.

Recently there has been an increase in interest in causality topics in reinforcement learning [Zhu and Chen, 2020, Buesing et al., 2019, Dasgupta et al., 2019], where the proposed methods may also be applicable. Assuming that we have a skeleton graph of the relevant quantities in the RL model, *meta-CGNN* could be used to efficiently infer causal direction of unoriented edges.

3.6 Acknowledgements

We would like to thank Pengzhou (Abel) Wu for interesting discussions. JFT is supported by the EPSRC and MRC through the OxWaSP CDT programme EP/L016710/1. KF is supported by SPS KAKENHI 18K19793 and JST CREST JPMJCR2015.

Appendix

3.A Appendix

3.A.1 Ablation study

In this section, we study the necessity of task embedding in our model and show that not every model can be trained in this joint dataset fashion. To this end, we used a CGNN [Goudet et al., 2017] that has been trained exactly like our *meta-CGNN*, i.e. *meta-CGNN* where we removed the amortization network and the FiLM layer. In this case, only the decoder was trained jointly across the datasets and we see that this naïve version of extending CGNN does not perform well, with accuracy hovering around 55-60%, which are close to chance level, for the Tuebingen dataset with at most 1500 datapoints per dataset. It behaves similarly on the synthetic data with a **chance level accuracy** for *CE-Net* dataset with 1500 datapoints.

Next, we investigate the importance of the FiLM layer and run the experiments from our experiment section using DeepSets embeddings, with and without the FiLM layer while keeping the amortization network. We see from Table 3.B.1 a significant performance boost when using the FiLM as it allows the decoder at test time more flexibility to adapt, instead of relying solely on the amortization network. This shows the importance of each component in the proposed architecture and that adapting cause-effect methods into the meta learning setting is a non-trivial task. We see similar trends for *meta-CGNN* using CME. Lastly, we have conduct an ablation study for the amortization network and noticed a similar/slightly higher performance with the amortization network in *meta-CGNN*.

3.B Additional details on experiments setup

In this section of the Appendix we will explain the experimental setup in more detail. For all our synthetic datasets, we have 300 datasets. We then use 100 pairs to train

our meta-CGNN algorithm and test on the remaining 200. For the standard methods, we only consider the 200 testing datasets as they cannot incorporate information from different datasets. Each dataset then has either 1500 or 100 datapoints.

Note that for the Tuebingen dataset contains 99 bivariate datasets. We use the same setup as in [Goudet et al., 2018], which means that datasets have at most 1500 or at most 100 datapoints. We use a similar setup as in the synthetic experiments, and the only difference is that we employ 5-fold cross-validation for training and testing. To be concrete, we train on 4 of the folds and test on the last one. We repeat this procedure such that each fold has been the test fold at one point, while the remaining were acting as training. That way we can obtain fair estimates of the directions. We repeat these experiments 3 times with different splits.

Next we provide the neural network architecture for meta-CGNN as well as hyper-parameters. For simplicity, just as in [Goudet et al., 2018], we use a fixed learning rate of 0.01, Adam optimizer, and up to 500 epochs. We used a mini-batch size of 10, i.e. 10 datasets for each gradient update. In terms of the neural network architectures we used a 2-hidden layer network for the encoder, FiLM and amortization network. For the decoder we follow [Goudet et al., 2018] and use a 1-hidden layer NN. All activations used are ReLU.

For the encoder network (meta-CGNN DeepSet), we fix $[40, 10]$ hidden nodes per layer. For the hyper-parameters λ of the Conditional Mean Embedding Operator (CMEO) as well as lengthscale l for the Random Fourier Features embedding we used $\lambda = 1$ and $l = 1$. These were the default settings and worked well. In the future we will cross-validate over these to see if there are any performance gains.

For the amortization network we have $[40, 20]$ hidden nodes layer and output μ_z and σ_z which are both $1D$. In our case the latent variable Z is always one dimensional. For the FiLM layer we have $[40, 40]$ hidden nodes and output 2x decoder hidden nodes. This is because we need a shift and a scale parameter for the hidden layer of the decoder. In terms of the decoder, we cross-validate and choose either 40 or 5 hidden nodes. The reason we chose values around 40 was because we took [Goudet et al., 2018] as a starting point and it worked well, around 40.

#datapoints	Gauss		Multi		Net	
	FiLM	noFiLM	FiLM	noFiLM	FiLM	noFiLM
1500	**0.78**	0.73	**0.75**	0.46	**0.70**	0.55
500	**0.80**	0.72	**0.73**	0.49	**0.70**	0.50
100	**0.73**	0.64	**0.72**	0.63	**0.72**	0.50

Table 3.B.1: Accuracy of the proposed method with DeepSet embedding with and without the FiLM layer

3.C AUPRC and Accuracy of the methods

Here we present the AURPC (area under the precision recall curve) as well as accuracy of the proposed methods as well as competing methods. Note that we report the weighted accuracy for the Tuebingen dataset as in [Mooij et al., 2016] to account for similarities in the datasets. We note that the accuracy follows a similar trend to the AUPRC, however we opted to show the AUPRC in the main text as it is more indicative of the confidence for each prediction. AUPRC also allows us to take into account the fact that a method might be uncertain about its prediction, i.e. it is better to give no answer than a wrong answer.

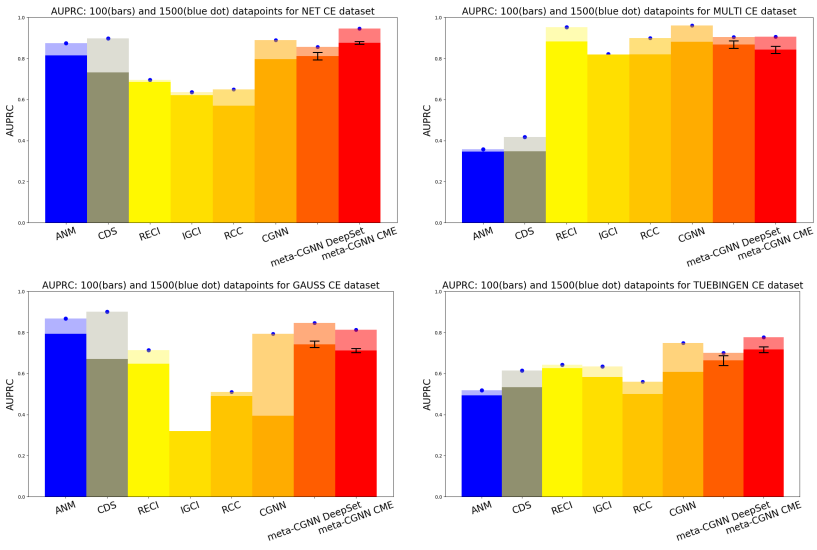

Figure 3.C.1: AUPRC for Net, Multi, Gauss and Tuebingen dataset. The blue dots represent the AUPRC with 1500 datapoints, whereas the barplots are with 100 datapoints. Note that the proposed meta-CGNN is in most cases significantly better that standard CGNN, which is trained individually on each dataset. In addition, meta-CGNN even though not the best on each dataset remains comparable across different datasets and sees the smallest drop in performance when reducing the dataset size.

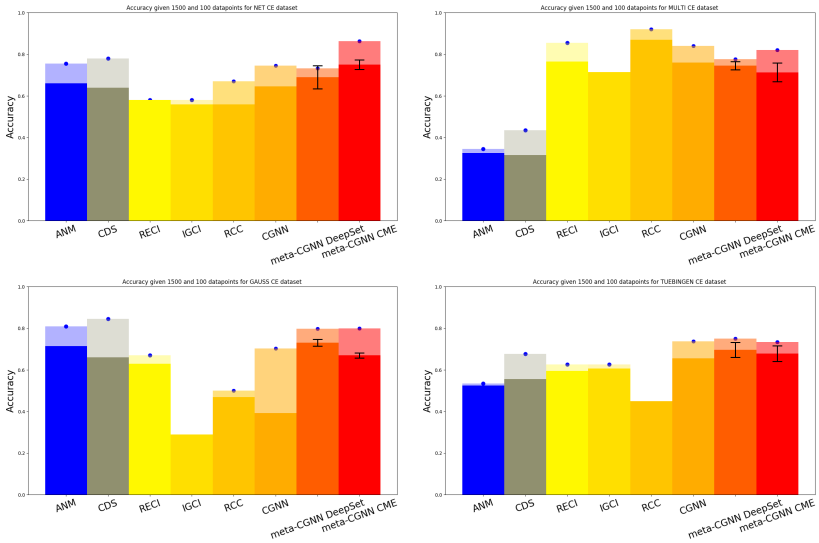

Figure 3.C.2: ACC for Net, Multi, Gauss and Tuebingen dataset. The blue dots represent the ACC with 1500 datapoints, whereas the barplots are with 100 datapoints. Note that the proposed meta-CGNN is in most cases significantly better that standard CGNN, which is trained individually on each dataset. In addition, meta-CGNN even though not the best on each dataset remains comparable across different datasets and sees the smallest drop in performance when reducing the dataset size.

3.D Note on competing methods and RCC

For the competing methods, we used the excellent *Causal discovery toolbox* by [Kalainathan and Goudet, 2020], which is provided in an open source Github repository, with all the popular methods implemented. We have done minimal changes to the algorithms as one can easily streamline the experiments by just inserting the datasets into their framework. The only thing we changed was to use 12 instead of 32 or 64 different models for CGNN [Goudet et al., 2018] and kept the hidden layer size as recommended by Goudet et al. to around 40 . The rest remained the same as described in the CGNN paper. The results we got on the 1500 datapoints are comparable to the one noted in the paper with a slight degradation of the performance due to less model averaging, i.e. 12 instead 32 or 64, which was not feasible due to our computational constraints.

In addition, we want to note that CGNN requires about 24 min per dataset at testing time, whereas our method can do inference in less than 1min. This is a big computational saving at test time while retaining similar or better performance.

ACC	Gauss	Multi	Net	Tueb
1500	0.50	0.92	0.67	0.39 ± 0.02
100	0.47	0.87	0.56	0.45 ± 0.03

AUPRC	Gauss	Multi	Net	Tueb
1500	0.51	0.90	0.65	0.56 ± 0.01
100	0.49	0.82	0.57	0.50 ± 0.03

Table 3.D.1: Results for RCC method. As we can see RCC also struggles with *CE-Gauss* dataset and seems to reasonable well on the *CE-Multi* dataset.

.

Lastly, we also investigated the performance in particular the performance of RCC (randomized causation coefficient) method [Lopez-Paz et al., 2015a] in our setting. RCC also uses the formalism of kernel mean embeddings (approximated using random features) in order to build a classifier for causal direction. However, it is data hungry and typically requires a lot of generated synthetic data to train the classifier, as noted in [Wu and Fukumizu, 2020]. Using again the toolbox by [Kalainathan and Goudet, 2020], we use the same procedure as in meta-CGNN, where 100 of the the synthetic datasets are used for training and remaining ones for testing. For the Tuebingen dataset, we again perform a 5-fold cross-validation, repeating the experiment three times, and reporting mean and standard deviation in Table 3.D.1. The performance of RCC is poor except for the *CE-Multi* dataset where most other methods also did reasonably well, thus showing is that using CME alone does not solve the problem.

Statement of Authorship for joint/multi-authored papers for PGR thesis

To appear at the end of each thesis chapter submitted as an article/paper

The statement shall describe the candidate's and co-authors' independent research contributions in the thesis publications. For each publication there should exist a complete statement that is to be filled out and signed by the candidate and supervisor **(only required where there isn't already a statement of contribution within the paper itself).**

Title of Paper	Meta Learning for Causal Direction
Publication Status	xPublished □ Accepted for Publication □Submitted for Publication □Unpublished and unsubmitted work written in a manuscript style
Publication Details	Ton, J.-F., Sejdinovic, D., & Fukumizu, K. (2021). Meta Learning for Causal Direction. *Proceedings of the AAAI Conference on Artificial Intelligence*, *35*(11), 9897-9905.

Student Confirmation

Student Name:	Jean-Francois Ton		
Contribution to the Paper	In this work, I was responsible for all the experiments. Together with KF we discussed potential avenues of how to extend causality to the meta learning setting. I then proposed to extend my previous work using Kernel mean embeddings to the causality setting. KF and DS provided valuable insight and also helped writing the paper.		
Signature		Date	30/03/2022

Supervisor Confirmation

By signing the Statement of Authorship, you are certifying that the candidate made a substantial contribution to the publication, and that the description described above is accurate.

Supervisor name and title:	Professor Dino Sejdinovic		
Supervisor comments	All above seems accurate and fair.		
Signature		Date	30/03/2022

4

BAYESIMP: Uncertainty Quantification for Causal Data Fusion

Abstract

While causal models are becoming one of the mainstays of machine learning, the problem of uncertainty quantification in causal inference remains challenging. In this paper, we study the causal data fusion problem, where datasets pertaining to multiple causal graphs are combined to estimate the average treatment effect of a target variable. As data arises from multiple sources and can vary in quality and quantity, principled uncertainty quantification becomes essential. To that end, we introduce Bayesian Interventional Mean Processes, a framework which combines ideas from probabilistic integration and kernel mean embeddings to represent interventional distributions in the reproducing kernel Hilbert space, while taking into account the uncertainty within each causal graph. To demonstrate the utility of our uncertainty estimation, we apply our method to the Causal Bayesian Optimisation task and show improvements over state-of-the-art methods.

4.1 Introduction

Causal inference has seen a significant surge of research interest in areas such as healthcare [Thompson, 2019], ecology [Courtney et al., 2017], and optimisation [Aglietti et al., 2020a]. However, data fusion, the problem of merging information from multiple data sources, has received limited attention in the context of causal modelling, yet presents significant potential benefits for practical situations [Meng et al., 2020, Singh et al., 2019b]. In this work, we consider a causal data fusion problem where two causal graphs are combined for the purposes of inference of a target variable (see Fig.4.1.1). In particular, our goal is to quantify the uncertainty under such a setup and determine the level of confidence in our treatment effect estimates.

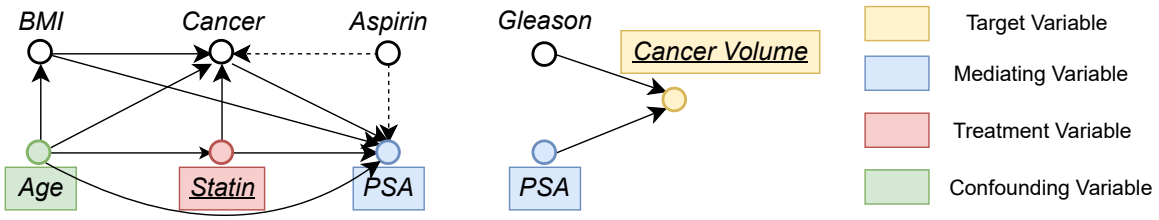

Figure 4.1.1: Example problem setup: Causal graphs collected in two separate medical studies i.e. [Ferro et al., 2015] and [Stamey et al., 1989]. (Left) $\mathcal{D}_1$: Data describing the causal relationships between statin level and Prostate Specific Antigen (PSA). (Right) $\mathcal{D}_2$: Data from a prostate cancer study for patients about to receive a radical prostatectomy. Goal: **Model** $\mathbb{E}[\boldsymbol{Cancer\ \ Volume}|\boldsymbol{do(Statin)}]$ while also quantifying its uncertainty.

Let us consider the motivating example in Fig.4.1.1, where a medical practitioner is investigating how *prostate cancer volume* is affected by a *statin* drug dosage. We consider the case where the doctor only has access to two separate medical studies describing the quantities of interest. On one hand we have observational data, from one medical study $\mathcal{D}_1$ [Thompson, 2019], describing the causal relationship between *statin* level and *prostate specific antigen (PSA)*, and on the other hand we have observational data, from a second study $\mathcal{D}_2$ [Stamey et al., 1989], that looked into the link between *PSA* level and *prostate cancer volume*. The goal is to model the **interventional effect** between our target variable (*cancer volume*) and the treatment variable (*statin*). This problem setting is different from the standard observational scenario as it comes with the following challenges:

- **Unmatched data:** Our goal is to estimate $\mathbb{E}[cancer\ volume|do(statin)]$ but the

observed *cancer volume* is not paired with *statin* dosage. Instead, they are related via a mediating variable *PSA*.

- **Uncertainty quantification:** The two studies may be of different data quantity/quality. Furthermore, a covariate shift in the mediating variable, i.e. a difference between its distributions in two datasets, may cause inaccurate extrapolation. Hence, we need to account for uncertainty in both datasets.

Formally, let X be the treatment (*Statin*), Y be the mediating variable (*PSA*) and T our target (*cancer volume*), and our aim is to estimate $\mathbb{E}[T|do(X)]$ i.e. the expected value of T when intervening on X Pearl [1995]. The problem of unmatched data in a similar context has been previously considered by [Singh et al., 2019b] using a two-staged regression approach ($X \to Y$ and $Y \to T$). However, uncertainty quantification, despite being essential if our estimates of interventional effects will guide decision-making, has not been previously explored. In particular, it is crucial to quantify the uncertainty in both stages as this takes into account the lack of data in specific parts of the space. Given that we are using different datasets for each stage, there are also two sources of epistemic uncertainties (due to lack of data) as well as two sources of aleatoric uncertainties (due to inherent randomness in Y and T) [Hüllermeier and Waegeman, 2021] . It is thus natural to consider regression models based on Gaussian Processes (GP) [Rasmussen and Williams, 2005a], as they are able to model both types of uncertainties. However, as GPs, or any other standard regression models, are designed to model conditional expectations only and will fail to capture the underlying distributions of interest (e.g. if there is multimodality in Y as discussed in [Ton et al., 2021a]). This is undesirable since, as we will see, interventional effect estimation requires accurate estimates of distributions. While one could in principle resort to density estimation methods, this becomes challenging since we typically deal with a number of conditional/ interventional densities.

In this paper, we introduce the framework of *Bayesian Interventional Mean Processes* (BayesIMP) to circumvent the challenges in the causal data fusion setting described above. BayesIMP considers kernel mean embeddings [Muandet et al., 2017] for representing distributions in a reproducing kernel Hilbert space (RKHS), in which the whole arsenal of kernel methods can be extended to probabilistic inference (e.g. kernel Bayes rule [Fukumizu et al., 2010], hypothesis testing [Zhang et al., 2018a], distribution regression [Law et al., 2018]). Specifically, BayesIMP uses kernel

mean embeddings to represent the interventional distributions and to analytically marginalise out Y, hence accounting for aleatoric uncertainties. Further, BayesIMP uses GPs to estimate the required kernel mean embeddings from data in a Bayesian manner, which allows to quantify the epistemic uncertainties when representing the interventional distributions. To illustrate the quality of our uncertainty estimates, we apply BayesIMP to Causal Bayesian Optimisation [Aglietti et al., 2020b], an efficient heuristic to optimise objective functions of the form $x^* = \arg\min_{x \in \mathcal{X}} \mathbb{E}[T|do(X) = x]$. Our contributions are summarised below:

1. We propose a novel *Bayesian Learning of Conditional Mean Embedding* (BayesCME) that allows us to estimate conditional mean embeddings in a Bayesian framework.

2. Using BayesCME, we propose a novel *Bayesian Interventional Mean Process* (BayesIMP) that allows us to model interventional effect across causal graphs without explicit density estimation, while obtaining uncertainty estimates for $\mathbb{E}[T|do(X) = x]$.

3. We apply BayesIMP to Causal Bayesian Optimisation, a problem introduced in [Aglietti et al., 2020b] and show significant improvements over existing state-of-the-art methods.

Note that [Bareinboim and Pearl, 2016] also considered a causal fusion problem but with a different objective. They focused on extrapolating experimental findings across treatment domains, i.e. inferring $\mathbb{E}[Y|do(X)]$ when only data from $p(Y|do(S))$ is observed, where S is some other treatment variable. In contrast, we focus on modelling combined causal graphs, with a strong emphasis on uncertainty quantification. While [Singh et al., 2020] considered mapping interventional distributions in the RKHS to model quantities such as $\mathbb{E}[T|do(X)]$, they only considered a frequentist approach, which does not account for epistemic uncertainties.

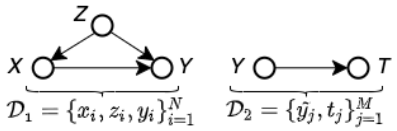

Figure 4.1.2: A general two stage causal learning setup.

Notations. We denote X, Y, Z as random variables taking values in the non-empty sets $\mathcal{X}, \mathcal{Y}$ and $\mathcal{Z}$ respectively. Let $k_x : \mathcal{X} \times \mathcal{X} \to \mathbb{R}$ be positive definite kernels on X with an associated RKHS $\mathcal{H}_{k_x}$. The corresponding canonical feature map $k_x(x', \cdot)$ is denoted as $\phi_x(x')$. Analogously for Y and Z.

In the simplest setting, we observe i.i.d samples $\mathcal{D}_1 = \{x_i, y_i, z_i\}_{i=1}^N$ from joint distribution $\mathbb{P}_{XYZ}$ which we concatenate into vectors $\mathbf{x} := [x_1, ..., x_N]^\top$. Similarly for $\mathbf{y}$ and $\mathbf{z}$. For this work, X is referred as *treatment variable*, Y as *mediating variable* and Z as *adjustment variables* accounting for confounding effects. With an abuse of notation, features matrices are defined by stacking feature maps along the columns, i.e $\Phi_{\mathbf{x}} := [\phi_x(x_1), ..., \phi_x(x_N)]$. We denote the Gram matrix as $K_{\mathbf{xx}} := \Phi_{\mathbf{x}}^\top \Phi_{\mathbf{x}}$ and the vector of evaluations $k_{x\mathbf{x}}$ as $[k_x(x, x_1), ..., k_x(x, x_N)]$. We define $\Phi_{\mathbf{y}}, \Phi_{\mathbf{z}}$ analogously for $\mathbf{y}$ and $\mathbf{z}$.

Lastly, we denote $T = f(Y) + \epsilon$ as our *target variable*, which is modelled as some noisy evaluation of a function $f : \mathcal{Y} \to \mathcal{T}$ on Y while ϵ being some random noise. For our problem setup we observe a second dataset of i.i.d realisations $\mathcal{D}_2 = \{\tilde{y}_j, t_j\}_{j=1}^M$ from the joint $\mathbb{P}_{YT}$ independent of $\mathcal{D}_1$. Again, we define $\tilde{\mathbf{y}} := [\tilde{y}_1, ..., \tilde{y}_M]^\top$ and $\mathbf{t} := [t_1, ..., t_M]^\top$ just like for $\mathcal{D}_1$. See Fig.4.1.2 for illustration.

4.2 Background

Representing interventional distributions in an RKHS[1] has been explored in different contexts [Muandet et al., 2021, Singh et al., 2020, Mitrovic et al., 2018]. In particular, when the treatment is continuous, [Singh et al., 2020] introduced the *Interventional Mean Embeddings* (IMEs) to model densities in an RKHS by utilising smoothness across treatments. Given that IME is an important building block to our contribution, we give it a detailed review by first introducing the key concepts of *do*-calculus [Pearl, 1995] and conditional mean embeddings [Song et al., 2013].

[1]We refer the reader to a detailed review of RKHS methods provided in the Appendix of [Singh et al., 2019b]

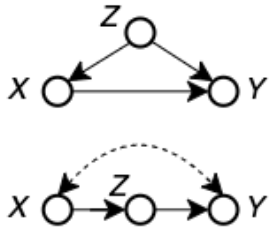

Figure 4.2.1: (Top) Backdoor adjustment (Bottom) Front-door adjustment, dashed edges denote unobserved confounders.

4.2.1 Interventional distribution and *do*-calculus

In this work, we consider the structural causal model [Pearl, 1995] (SCM) framework, where a causal directed acyclic graph (DAG) $\mathcal{G}$ is given and encodes knowledge of existing causal mechanisms amongst the variables in terms of conditional independencies. Given random variables X and Y, a central question in interventional inference [Pearl, 1995] is to estimate the distribution $p(Y|do(X) = x)$, where $\{do(X) = x\}$ represents an intervention on X whose value is set to x. Note that this quantity is not directly observed given that we are usually only given observational data, i.e, data sampled from the conditional $p(Y|X)$ but not from the interventional density $p(Y|do(X))$. However, Pearl [Pearl, 1995] developed *do*-calculus which allows us to estimate interventional distributions from purely observational distributions under the identifiability assumption. Here we present the backdoor and front-door adjustments, which are the fundamental components of DAG based causal inference.

The backdoor adjustment is applicable when there are observed confounding variables Z between the cause X and the effect Y (see Fig. 4.2.1 (Top)). In order to correct for this confounding bias we can use the following equation, adjusting for Z as $p(Y|do(X) = x) = \int_{\mathcal{Z}} p(Y|X = x, z)p(z)dz$.

The front-door adjustment applies to cases when confounders are unobserved (see Fig. 4.2.1 (Bottom)). Given a set of front-door adjustment variables Z, we can again correct the estimate for the causal effect from X to Y with $p(Y|do(X) = x) = \int_{\mathcal{Z}} \int_{\mathcal{X}} p(Y|x', z)p(z|X = x)p(x')dx'dz$.

We rewrite the above formulae in a more general form as we show below. For the remainder of the paper we will opt for this notation:

$$p(Y|do(X) = x) = \mathbb{E}_{\Omega_x}[p(Y|\Omega_x)] = \int p(Y|\Omega_x)p(\Omega_x)d\Omega_x \tag{4.1}$$

For backdoor we have $\Omega_x = \{X = x, Z\}$ and $p(\Omega_x) = \delta_x p(Z)$ where δ_x is the Dirac measure at $X = x$. For front-door, $\Omega_x = \{X', Z\}$ and $p(\Omega_x) = p(X')p(Z|X = x)$.

4.2.2 Conditional Mean Embeddings

Kernel mean embeddings of distributions provide a powerful framework for representing probability distributions [Muandet et al., 2017, Song et al., 2013] in an RKHS. In particular, we work with conditional mean embeddings (CMEs) in this paper. Given random variables X, Y with joint distribution $\mathbb{P}_{XY}$, the conditional mean embedding with respect to the conditional density $p(Y|X = x)$, is defined as:

$$\mu_{Y|X=x} := \mathbb{E}_{Y|X=x}[\phi_y(Y)] = \int_{\mathcal{Y}} \phi_y(y)p(y|X = x)dy \tag{4.2}$$

CMEs allow us to represent the distribution $p(Y|X = x)$ as an element $\mu_{Y|X=x}$ in the RKHS $\mathcal{H}_{k_y}$ without having to model the densities explicitly. Following [Song et al., 2013], CMEs can be associated with a Hilbert-Schmidt operator $\mathcal{C}_{Y|X} : \mathcal{H}_{k_x} \to \mathcal{H}_{k_y}$, known as the conditional mean embedding operator, which satisfies $\mu_{Y|X=x} = \mathcal{C}_{Y|X}\phi_x(x)$ where $\mathcal{C}_{Y|X} := \mathcal{C}_{YX}\mathcal{C}_{XX}^{-1}$ with $\mathcal{C}_{YX} := \mathbb{E}_{Y,X}[\phi_y(Y) \otimes \phi_x(X)]$ and $\mathcal{C}_{XX} := \mathbb{E}_{X,X}[\phi_x(X) \otimes \phi_x(X)]$ being the covariance operators. As a result, the finite sample estimator of $\mathcal{C}_{Y|X}$ based on the dataset $\{\mathbf{x}, \mathbf{y}\}$ can be written as:

$$\hat{\mathcal{C}}_{Y|X} = \Phi_{\mathbf{y}}(K_{\mathbf{xx}} + \lambda I)^{-1}\Phi_{\mathbf{x}}^T \tag{4.3}$$

where $\lambda > 0$ is a regularization parameter. Note that from Eq.4.3, [Grünewälder et al., 2012] showed that the CME can be interpret as a vector-valued kernel ridge regressor (V-KRR) i.e. $\phi_x(x)$ is regressed to an element in $\mathcal{H}_{k_y}$. This is crucial as CMEs allow us to turn the integration, in Eq.4.2, into a regression task and hence remove the need for explicit density estimation. This insight is important as it allows us to derive analytic forms for our algorithms. Furthermore, the regression formalism of CMEs motivated us to derive a Bayesian version of CME using vector-valued Gaussian Processes (V-GP), see Sec.4.3.

4.2.3 Interventional Mean Embeddings

Interventional Mean Embeddings (IME) [Singh et al., 2020] combine the above ideas to represent interventional distributions in RKHSs. We derive the front-door adjustment embedding here but the backdoor adjustment follows analogously. Denote $\mu_{Y|do(X)=x}$ as

the IME corresponding to the interventional distribution $p(Y|do(X) = x)$, which can be written as:

$$\mu_{Y|do(X)=x} := \int_{\mathcal{Y}} \phi_y(y) p(y|do(X) = x) dy = \int_{\mathcal{X}} \int_{\mathcal{Z}} \underbrace{\Big(\int_{\mathcal{Y}} \phi_y(y) p(y|x', z) dy \Big)}_{\text{CME } \mu_{Y|X=x,Z=z}} p(z|x) p(x') dz dx'$$

using the front-door formula with adjustment variable Z, and rearranging the integrals. By definition of CME $\int \phi_y(y) p(y|x', z) dy = C_{Y|X,Z}(\phi_x(x') \otimes \phi_z(z))$ and linearity of integration, we have

$$= C_{Y|X,Z}\Big(\underbrace{\int_{\mathcal{X}} \phi_x(x') p(x') dx'}_{=\mu_X} \otimes \underbrace{\int_{\mathcal{Z}} \phi_z(z) p(z|x) dz}_{=\mu_{Z|X=x}} \Big) = C_{Y|X,Z}\Big(\mu_X \otimes \mu_{Z|X=x}\Big)$$

Using notations from Sec.4.2.1, embedding interventional distributions into an RKHS is as follows.

Proposition 1. *Given an identifiable do-density of the form* $p(Y|do(X) = x) = \mathbb{E}_{\Omega_x}[p(Y|\Omega_x)]$*, the general form of the empirical interventional mean embedding is given by,*

$$\hat{\mu}_{Y|do(X)=x} = \Phi_Y (K_{\Omega_x} + \lambda I)^{-1} \Phi_{\Omega_x}(x)^\top \tag{4.4}$$

where $K_{\Omega_x} = K_{XX} \odot K_{ZZ}$ *and* $\Phi_{\Omega_x}(x)$ *is derived depending on* $p(\Omega_x)$*. In particular, for backdoor adjustments,* $\Phi^{(bd)}_{\Omega_x}(x) = \Phi_X^\top k_X(x, \cdot) \odot \Phi_Z^\top \hat{\mu}_z$ *and for front-door* $\Phi^{(fd)}_{\Omega_x}(x) = \Phi_X^\top \hat{\mu}_X \odot \Phi_Z^\top \hat{\mu}_{Z|X=x}$*.*

4.3 Our Proposed Method

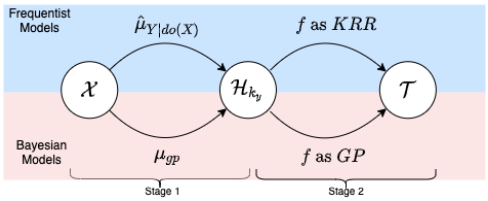

Figure 4.3.1: Two-staged causal learning problem

METHODS	Stage 1	Stage 2
IME [Singh et al., 2020]	KRR	KRR
IMP (Ours)	KRR	GP
BayesIME (Ours)	GP	KRR
BayesIMP (Ours)	GP	GP

Table 4.3.1: Summary of our proposed methods

Two-staged Causal Learning. Given two independent datasets $\mathcal{D}_1 = \{(x_i, z_i, y_i)\}_{i=1}^N$ and $\mathcal{D}_2 = \{(\tilde{y}_j, t_j)\}_{j=1}^M$, our goal is to model the average average of T when intervening on variable X, i.e model $g(x) = \mathbb{E}[T|do(X) = x]$. Note that the target variable T and the treatment variable X are never jointly observed. Rather, they are linked via a mediating variable Y observed in both datasets. In our problem setting, we make the following two assumptions: **(A1)** The treatment only affects the target through the mediating variable, i.e $P(T|do(X), Y) = p(T|Y)$, in other words, that in the true data generating model P(X, Y, Z, T), all causal paths from X to T are mediated through Y. and **(A2)** Function f given by $f(y) = \mathbb{E}[T|Y = y]$ belongs to an RKHS $\mathcal{H}_{k_y}$.

We can thus express the average treatment effect as:

$$g(x) = \mathbb{E}[T|do(X) = x] = \int_{\mathcal{Y}} \underbrace{\mathbb{E}[T|do(X) = x, Y = y]}_{=\mathbb{E}[T|Y=y], \text{ since } T \perp\!\!\!\perp do(X)|Y} p(y|do(X) = x)dy \tag{4.5}$$

$$= \int_{\mathcal{Y}} f(y)p(y|do(X) = x)dy = \langle f, \mu_{Y|do(X)=x} \rangle_{\mathcal{H}_{k_y}}. \tag{4.6}$$

The final expression decomposes the problem of estimating g into that of estimating the IME $\mu_{Y|do(X)}$ (which can be done using $\mathcal{D}_1$) and that of estimating the integrand $f : \mathcal{Y} \to \mathcal{T}$ (which can be done using $\mathcal{D}_2$). Each of these two components can either be estimated using a GP or KRR approach (See Table 4.3.1). Furthermore, the reformulation as an RKHS inner product is crucial, as it circumvents the need for density estimation as well as the need for subsequent integration in Eq.4.6. Rather, the main parts of the task can now be viewed as two instances of regression (recall that mean embeddings can be viewed as vector-valued regression).

To model g and quantify its uncertainty, we propose 3 GP-based approaches. While the first 2 methods, *Interventional Mean Process* (IMP) and *Bayesian Interventional Mean Embedding* (BayesIME) are novel derivations that allow us to quantify uncertainty from either one of the datasets, we treat them as intermediate yet necessary steps to derive our main algorithm, *Bayesian Interventional Mean Process* (BayesIMP), which allows us to quantify uncertainty from both sources in a principled way. For a summary of the methods, see Fig.4.3.1 and Table 4.3.1. All derivations are included in the appendix.

***Interventional Mean Process*:** Firstly, we train f as a GP using $\mathcal{D}_2$ and model $\mu_{Y|do(X)=x}$ as V-KRR using $\mathcal{D}_1$. By drawing parallels to Bayesian quadrature [Briol et al., 2019] and conditional mean process introduced in [Chau et al., 2021a], the integral of interest $g(x) = \int f(y)p(y|do(X) = x)dy$ will be a GP indexed by the

treatment variable X. We can then use the empirical embedding $\hat{\mu}_{Y|do(X)}$ learnt in $\mathcal{D}_1$ to obtain an analytic mean and covariance of g.

Bayesian Interventional Mean Embedding: Next, to account for the uncertainty from $\mathcal{D}_1$, we model f as a KRR and $\mu_{Y|do(X)=x}$ using a V-GP. We introduce our novel *Bayesian Learning of Conditional Mean Embeddings* (BayesCME), which uses a *nuclear dominant kernel* [Lukić and Beder, 2001] construction, similar to [Flaxman et al., 2016], to ensure that the inner product $\langle f, \mu_{Y|do(X)=x} \rangle$ is well-defined. As the embedding is a GP, the resulting inner product is also a GP and hence takes into account the uncertainty in $\mathcal{D}_1$.(See Prop. 4).

Bayesian Interventional Mean Process: Lastly, in order to account for uncertainties coming from both $\mathcal{D}_1$ and $\mathcal{D}_2$, we combine ideas from the above IMP and BayesIME. We place GPs on both f and $\mu_{Y|do(X)}$ and use their inner product to model $\mathbb{E}[T|do(X)]$. Interestingly, the resulting uncertainty can be interpreted as the sum of uncertainties coming from IMP and BayesIME with an additional interaction term (See Prop.5).

4.3.1 Interventional Mean Process

Firstly, we consider the case where f is modelled using a GP and $\mu_{Y|do(X)=x}$ using a V-KRR. This allows us to take into account the uncertainty from $\mathcal{D}_2$ by modelling the relationship between Y and T using in a GP. Drawing parallels to Bayesian quadrature [Briol et al., 2019] where integrating f with respect to a marginal measure results into a Gaussian random variable, we integrate f with respect to a conditional measure, thus resulting in a GP indexed by the conditioning variable. Note that [Chau et al., 2021a] studied this GP in a non-causal setting, for a very specific downscaling problem. In this work, we extend their approach to model uncertainty in the causal setting. The resulting mean and covariance are then estimated analytically, i.e without integrals, using the empirical IME $\hat{\mu}_{Y|do(X)}$ learnt from $\mathcal{D}_1$.

Proposition 2 (IMP). *Given dataset $D_1 = \{(x_i, y_i, z_i)\}_{i=1}^N$ and $D_2 = \{(\tilde{y}_j, t_j)\}_{j=1}^M$, if f is the posterior GP learnt from $\mathcal{D}_2$, then $g = \int f(y)p(y|do(X))dy$ is a GP $\mathcal{GP}(m_1, \kappa_1)$ defined on the treatment variable X with the following mean and covariance estimated*

using $\hat{\mu}_{Y|do(X)}$,

$$m_1(x) = \langle \hat{\mu}_{Y|do(x)}, m_f \rangle_{\mathcal{H}_{k_y}} = \Phi_{\Omega_x}(x)^\top (K_{\Omega_x} + \lambda I)^{-1} K_{\mathbf{y}\tilde{\mathbf{y}}} (K_{\tilde{\mathbf{y}}\tilde{\mathbf{y}}} + \lambda_f I)^{-1} \mathbf{t} \tag{4.7}$$

$$\kappa_1(x, x') = \hat{\mu}^\top_{Y|do(x)} \hat{\mu}_{Y|do(x')} - \hat{\mu}^\top_{Y|do(x)} \Phi_{\tilde{\mathbf{y}}} (K_{\tilde{\mathbf{y}}\tilde{\mathbf{y}}} + \lambda I)^{-1} \Phi^\top_{\tilde{\mathbf{y}}} \hat{\mu}_{Y|do(x')} \tag{4.8}$$

$$= \Phi_{\Omega_x}(x)^\top (K_{\Omega_x} + \lambda I)^{-1} \tilde{K}_{\mathbf{yy}} (K_{\Omega_x} + \lambda I)^{-1} \Phi_{\Omega_x}(x') \tag{4.9}$$

where $\hat{\mu}_{Y|do(x)} = \hat{\mu}_{Y|do(X)=x}$, $K_{\tilde{\mathbf{y}}\mathbf{y}} = \Phi^\top_{\tilde{\mathbf{y}}} \Phi_{\mathbf{y}}$, m_f *and* $\tilde{K}_{\mathbf{yy}}$ *are the posterior mean function and covariance of* f *evaluated at* $\mathbf{y}$ *respectively.* $\lambda > 0$ *is the regularisation of the CME.* $\lambda_f > 0$ *is the noise term for GP* f*.* Ω_x *is the set of variables as specified in Prop.1.*

Summary: The posterior covariance between x and x' in IMP can be interpreted as the similarity between their corresponding empirical IMEs $\hat{\mu}_{Y|do(X)=x}$ and $\hat{\mu}_{Y|do(X)=x'}$ weighted by the posterior covariance $\tilde{K}_{\mathbf{yy}}$, where the latter corresponds to the uncertainty when modelling f as a GP in $\mathcal{D}_2$. However, since f only considers uncertainty in $\mathcal{D}_2$, we need to develop a method that allows us to quantify uncertainty when learning the IME from $\mathcal{D}_1$. In the next section, we introduce a Bayesian version of CME, which then lead to BayesIME, a remedy to this problem.

4.3.2 Bayesian Interventional Mean Embedding

To account for the uncertainty in $\mathcal{D}_1$ when estimating $\mu_{Y|do(X)}$, we consider a GP model for CME, and later extend to the interventional embedding IME. We note that Bayesian formulation of CMEs has also been considered in [Hsu et al., 2018], but with a specific focus on discrete target spaces.

Bayesian learning of conditional mean embeddings with V-GP. As mentioned in Sec.4.2, CMEs have a clear "feature-to-feature" regression perspective, i.e $\mathbb{E}[\phi_y(Y)|X = x]$ is the result of regressing $\phi_y(Y)$ onto $\phi_x(X)$. Hence, we consider a vector-valued GP construction to estimate the CME.

Let $\mu_{gp}(x, y)$ be a GP that models $\mu_{Y|X=x}(y)$. Given that $f \in \mathcal{H}_{k_y}$, for $\langle f, \mu_{gp}(x, \cdot) \rangle_{\mathcal{H}_{k_y}}$ to be well defined, we need to ensure $\mu_{gp}(x, \cdot)$ is also restricted to $\mathcal{H}_{k_y}$ for any fixed x. Consequently, we cannot define a $\mathcal{GP}(0, k_x \otimes k_y)$ prior on μ_{gp} as usual, as draws from such prior will almost surely fall outside $\mathcal{H}_{k_x} \otimes \mathcal{H}_{k_y}$ [Lukić and Beder, 2001]. Instead we define a prior over $\mu_{gp} \sim \mathcal{GP}(0, k_x \otimes r_y)$, where r_y is a *nuclear dominant kernel* [Lukić and Beder, 2001] over k_y, which ensures that samples paths of μ_{gp} live in $\mathcal{H}_{k_x} \otimes \mathcal{H}_{k_y}$ almost surely. In particular, we follow a similar construction as in [Flaxman

et al., 2016] and model r_y as $r_y(y_i, y_j) = \int k_y(y_i, u)k_y(u, y_j)\nu(du)$ where ν is some finite measure on Y. Hence we can now setup a vector-valued regression in $\mathcal{H}_{k_y}$ as follows:

$$\phi_y(y_i) = \mu_{gp}(x_i, \cdot) + \lambda^{\frac{1}{2}}\epsilon_i \tag{4.10}$$

where $\epsilon_i \sim \mathcal{GP}(0, r)$ are independent noise functions. By taking the inner product with $\phi_y(y')$ on both sides, we then obtain $k_y(y_i, y') = \mu_{gp}(x_i, y') + \lambda^{\frac{1}{2}}\epsilon_i(y')$. Hence, we can treat $k(y_i, y_j)$ as noisy evaluations of $\mu_{gp}(x_i, y_j)$ and obtain the following posterior mean and covariance for μ_{gp}.

Proposition 3 (**BayesCME**). *The posterior GP of μ_{gp} given observations $\{\mathbf{x}, \mathbf{y}\}$ has the following mean and covariance:*

$$m_\mu((x, y)) = k_{x\mathbf{x}}(K_{\mathbf{xx}} + \lambda I)^{-1}K_{\mathbf{yy}}R_{\mathbf{yy}}^{-1}r_{\mathbf{y}y} \tag{4.11}$$

$$\kappa_\mu((x, y), (x', y')) = k_{xx'}r_{y,y'} - k_{x\mathbf{x}}(K_{\mathbf{xx}} + \lambda I)^{-1}k_{\mathbf{x}x'}r_{\mathbf{y}y}R_{\mathbf{yy}}^{-1}r_{\mathbf{y}y'} \tag{4.12}$$

In addition, the following marginal likelihood can be used for hyperparameter optimisation,

$$-\frac{N}{2}\Big(\log|K_{\mathbf{xx}} + \lambda I| + \log|R|\Big) - \frac{1}{2}\operatorname{Tr}\Big((K_{\mathbf{xx}} + \lambda I)^{-1}K_{\mathbf{yy}}R_{\mathbf{yy}}^{-1}K_{\mathbf{yy}}\Big) \tag{4.13}$$

Note that in practice we fix the lengthscale of k_y and r_y when optimising the above likelihood. This is to avoid trivial solutions for the vector-valued regression problem as discussed in [Ton et al., 2021a]. The Bayesian version of the IME is derived analogously and we refer the reader to appendix due to limited space.

Finally, with V-GPs on embeddings defined, we can model $g(x)$ as $\langle f, \mu_{gp}(x, \cdot)\rangle_{H_{k_y}}$, which due to the linearity of the inner product, is itself a GP. Here, we first considered the case where f is a KRR learnt from $\mathcal{D}_2$ and call the model BayesIME.

Proposition 4 (**BayesIME**). *Given dataset $D_1 = \{(x_i, y_i, z_i)\}_{i=1}^N$ and $D_2 = \{(\tilde{y}_j, t_j)\}_{j=1}^M$, if f is a KRR learnt from $\mathcal{D}_2$ and $\mu_{Y|do(X)}$ modelled as a V-GP using $\mathcal{D}_1$, then $g = \langle f, \mu_{Y|do(X)}\rangle \sim \mathcal{GP}(m_2, \kappa_2)$ where,*

$$m_2(x) = \Phi_{\Omega_x}(x)^\top(K_{\Omega_x} + \lambda I)^{-1}K_{\mathbf{yy}}R_{\mathbf{yy}}^{-1}R_{\mathbf{y}\tilde{\mathbf{y}}}A \tag{4.14}$$

$$\kappa_2(x, x') = B\Phi_{\Omega_x}(x)^\top\Phi_{\Omega_x}(x) - C\Phi_{\Omega_x}(x)^\top(K_{\Omega_x} + \lambda I)^{-1}\Phi_{\Omega_x}(x') \tag{4.15}$$

where $A = (K_{\tilde{\mathbf{y}}\tilde{\mathbf{y}}} + \lambda_f I)^{-1}\mathbf{t}$, $B = A^\top R_{\tilde{\mathbf{y}}\tilde{\mathbf{y}}}A$ and $C = A^\top R_{\tilde{\mathbf{y}}\mathbf{y}}R_{\mathbf{yy}}^{-1}R_{\mathbf{y}\tilde{\mathbf{y}}}A$

Summary: Constants B and C in κ_2 can be interpreted as different estimation of $||f||_{\mathcal{H}_{k_y}}$, i.e the RKHS norm of f. As a result, for problems that are "harder" to learn in $\mathcal{D}_2$, i.e. corresponding to larger magnitude of $||f||_{\mathcal{H}_{k_y}}$, will result into larger values of B and C. Therefore the covariance κ_2 can be interpreted as uncertainty in $\mathcal{D}_1$ scaled by the difficulty of the problem to learn in $\mathcal{D}_2$.

4.3.3 Bayesian Interventional Mean Process

To incorporate both uncertainties in $\mathcal{D}_1$ and $\mathcal{D}_2$, we combine ideas from IMP and BayesIME to estimate $g = \langle f, \mu_{Y|do(X)} \rangle$ by placing GPs on both f and $\mu_{Y|do(X)}$. Again as before, a nuclear dominant kernel r_y was used to ensure the GP f is supported on $\mathcal{H}_{k_y}$. For ease of computation, we consider a finite dimensional approximation of the GPs f and $\mu_{Y|do(X)}$ and estimate g as the RKHS inner product between them. In the following we collate $\mathbf{y}$ and $\tilde{\mathbf{y}}$ into a single set of points $\hat{\mathbf{y}}$, which can be seen as landmark points for the finite approximation [Trecate et al., 1999]. We justify this in the Appendix.

Proposition 5 (**BayesIMP**). *Let f and $\mu_{Y|do(X)}$ be GPs learnt as above. Denote $\tilde{f}$ and $\tilde{\mu}_{Y|do(X)}$ as the finite dimensional approximation of f and $\mu_{Y|do(X)}$ respectively. Then $\tilde{g} = \langle \tilde{f}, \tilde{\mu}_{Y|do(X)} \rangle$ has the following mean and covariance:*

$$m_3(x) = E_x K_{\mathbf{y}\hat{\mathbf{y}}} K_{\hat{\mathbf{y}}\hat{\mathbf{y}}}^{-1} R_{\hat{\mathbf{y}}\tilde{\mathbf{y}}} (R_{\tilde{\mathbf{y}}\tilde{\mathbf{y}}} + \lambda_f I)^{-1} \mathbf{t} \tag{4.16}$$

$$\kappa_3(x, x') = \underbrace{E_x \Theta_1^\top \tilde{R}_{\hat{\mathbf{y}}\hat{\mathbf{y}}} \Theta_1 E_{x'}^\top}_{\text{Uncertainty from } \mathcal{D}_1} + \underbrace{\Theta_2^{(a)} F_{xx'} - \Theta_2^{(b)} G_{xx'}}_{\text{Uncertainty from } \mathcal{D}_2} + \underbrace{\Theta_3^{(a)} F_{xx'} - \Theta_3^{(b)} G_{xx'}}_{\text{Uncertainty from Interaction}} \tag{4.17}$$

where $E_x = \Phi_{\Omega_x}(x)^\top (K_{\Omega_x} + \lambda I)^{-1}, F_{xx'} = \Phi_{\Omega_x}(x)^\top \Phi_{\Omega_x}(x'), G_{xx'} = \Phi_{\Omega_x}(x)^\top (K_{\Omega_x} + \lambda I)^{-1} \Phi_{\Omega_x}(x')$, and $\Theta_1 = K_{\hat{\mathbf{y}}\hat{\mathbf{y}}}^{-1} R_{\hat{\mathbf{y}}\mathbf{y}} R_{\mathbf{y}\mathbf{y}}^{-1} K_{\mathbf{y}\mathbf{y}}$, $\Theta_2^{(a)} = \Theta_4^\top R_{\hat{\mathbf{y}}\hat{\mathbf{y}}} \Theta_4, \Theta_2^{(b)} = \Theta_4^\top R_{\hat{\mathbf{y}}\mathbf{y}} R_{\mathbf{y}\mathbf{y}}^{-1} R_{\mathbf{y}\hat{\mathbf{y}}} \Theta_4$ and $\Theta_3^{(a)} = tr(K_{\hat{\mathbf{y}}\hat{\mathbf{y}}}^{-1} R_{\hat{\mathbf{y}}\hat{\mathbf{y}}} K_{\hat{\mathbf{y}}\hat{\mathbf{y}}}^{-1} \bar{R}_{\hat{\mathbf{y}}\hat{\mathbf{y}}}), \Theta_3^{(b)} = tr(R_{\hat{\mathbf{y}}\mathbf{y}} R_{\mathbf{y}\mathbf{y}}^{-1} R_{\mathbf{y}\hat{\mathbf{y}}} K_{\hat{\mathbf{y}}\hat{\mathbf{y}}}^{-1} \bar{R}_{\hat{\mathbf{y}}\hat{\mathbf{y}}} K_{\hat{\mathbf{y}}\hat{\mathbf{y}}}^{-1})$ and $\Theta_4 = K_{\hat{\mathbf{y}}\hat{\mathbf{y}}}^{-1} R_{\hat{\mathbf{y}}\tilde{\mathbf{y}}} (K_{\tilde{\mathbf{y}}\tilde{\mathbf{y}}} + \lambda_f)^{-1} \mathbf{t}$. $\bar{R}_{\hat{\mathbf{y}}\hat{\mathbf{y}}}$ is the posterior covariance of f evaluated at $\hat{\mathbf{y}}$

Summary: While the first two terms in κ_3 resemble the uncertainty estimates from IMP and BayesIME, the last term acts as an extra interaction between the two uncertainties from $\mathcal{D}_1$ and $\mathcal{D}_2$. We note that unlike IMP and BayesIME, $\tilde{g}$ from Prop.5 is not a GP as inner products between Gaussian vectors are not Gaussian. Nonetheless, the mean and covariance can be estimated.

4.4 Experiments

In this section, we first present ablation studies on how our methods would perform under settings where we have missing data parts at different regions of the two datasets. We then demonstrate BayesIMP's proficiency in the Causal Bayesian Optimisation setting.

In particular, we compare our methods against the sampling approach considered in [Aglietti et al., 2020b]. [Aglietti et al., 2020b] start by modelling $f : Y \to T$ as GP and estimate the density $p(Y|do(X))$ using a GP along with do-calculus. Then given a treatment x, we obtain L samples of y_l and R samples of f_r from their posterior GPs. The empirical mean and standard deviation of the samples $\{f_r(y_l)\}_{l=1,r=1}^{L,R}$ can now be taken to estimate $\mathbb{E}[T|do(X) = x]$ as well as the correspondingly uncertainty. We emphasize that this point estimation requires repeated sampling and is thus inefficient compared to our approaches, where we explicitly model the uncertainty as covariance function.

Ablation study. In order to get a better intuition into our methods, we will start off with a preliminary example, where we investigate the uncertainty estimates in a toy case. We assume two simple causal graphs $X \to Y$ for $\mathcal{D}_1$ and $Y \to T$ for $\mathcal{D}_2$ and the goal is to estimate $\mathbb{E}[T|do(X) = x]$ (generating process given in the appendix). We compare our methods from Sec.4.3 with the sampling-based uncertainty estimation approach described above. In Fig.4.4.1 we plot the mean and the 95% credible interval of the resulting GP models for $\mathbb{E}[T|do(X) = x]$. On the x-axis we also plotted a histogram of the treatment variable x to illustrate its density.

From Fig.4.4.1(a), we see that the uncertainty for sampling is rather uniform across the ranges of x despite the fact we have more data around $x = 0$. This is contrary to our methods, which show a reduction of uncertainty at high x density regions. In particular, $x = -5$ corresponds to an extrapolation of data, where x gets mapped to a region of y where there is no data in $\mathcal{D}_2$. This fact is nicely captured by the spike of credible interval in Fig.4.4.1(c) since IMP utilises uncertainty from $\mathcal{D}_2$ directly. Nonetheless, IMP failed to capture the uncertainty stemming from $\mathcal{D}_1$, as seen from the fact that the credible interval did not increase as we have less data in the region $|x| > 5$. In contrast, BayesIME (Fig.4.4.1(d)) gives higher uncertainty around low x density regions but failed to capture the **extrapolation** phenomenon. Finally, BayesIMP Fig.4.4.1(e) seems to inherit the desirable characteristics from both IMP and BayesIME, due to

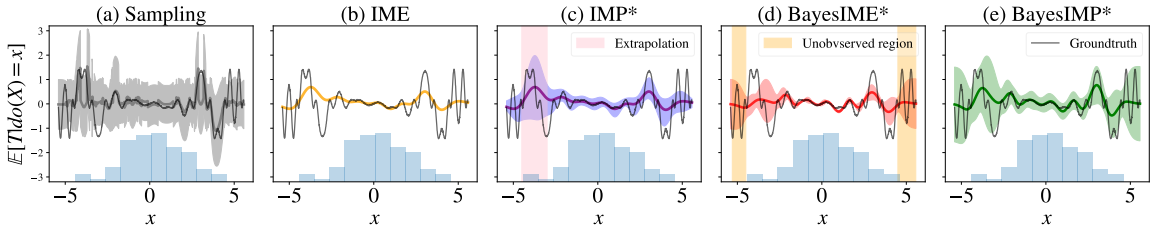

Figure 4.4.1: Ablation studies of various methods in estimating uncertainties for an illustrative experiment. $*$ indicates our methods. $N = M = 100$ data points are used. Uncertainty from sampling gives a uniform estimate of uncertainty and IME does not come with uncertainty estimates. We see IMP and BayesIME covering different regions of uncertainty while BayesIMP takes the best of both worlds.

taking into account uncertainties from both $\mathcal{D}_1$, $\mathcal{D}_2$. Hence, in the our experiments, we focus on BayesIMP and refer the reader to the appendix for the remaining methods.

BayesIMP for Bayesian Optimisation (BO). We now demonstrate, on both synthetic and real-world data, the usefulness of the uncertainty estimates obtained using our methods in BO tasks. Our goal is to utilise the uncertainty estimates to direct the search for the optimal value of $\mathbb{E}[T|do(X) = x]$ by querying as few values of the treatment variable X as possible, i.e. we want to optimize for $x^* = \arg\max_{x\in\mathcal{X}} \mathbb{E}[T|do(X) = x]$. For the first synthetic experiment (see Fig.4.4.2 (Top)), we will use the following two datasets: $\mathcal{D}_1 = \{x_i, u_i, z_i, y_i\}_{i=1}^N$ and $\mathcal{D}_2 = \{\tilde{y}_j, t_j\}_{j=1}^M$. Note that BayesIMP from Prop.5 is not a GP as inner products between Gaussian vectors are not Gaussian. Nonetheless, with the mean and covariance estimated, we will use moment matching to construct a GP out of BayesIMP for posterior inference. At the start, we are given $\mathcal{D}_1$ and $\mathcal{D}_2$, where these observations are used to construct a GP prior for the interventional effect of X on T, i.e $\mathbb{E}[T|do(X) = x]$, to provide a "warm" start for the BO.

Again we compare BayesIMP with the sampling-based estimation of $\mathbb{E}[T|do(X)]$ and its uncertainty, which is exactly the formulation used in the Causal Bayesian Optimisation algorithm (CBO) [Aglietti et al., 2020b]. In order to demonstrate how BayesIMP performs in the multimodal setting, we will be considering the case where we have the following distribution on Y i.e. $p(y|u, z) = \pi p_1(y|u, z) + (1-\pi)p_2(y|u, z)$ where Y is a mixture and $\pi \in [0, 1]$. These scenarios might arise when there is an unobserved binary variable which induces a switching between two regimes on how Y depends on (U, Z). In this case, the GP model of [Aglietti et al., 2020b] would only capture the conditional expectation of Y with an inflated variance, leading to slower convergence

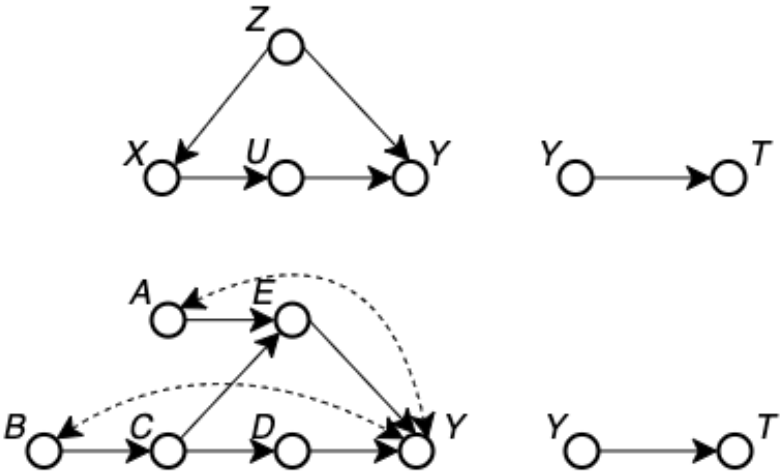

Figure 4.4.2: Illustration of synthetic data experiments.

and higher variance in the estimates of the prior as we will show in our experiments. Throughout our experiments, similarly to [Aglietti et al., 2020b], we will be using the expected improvement (EI) acquisition function to select the next point to query.

Synthetic data experiments. We compare BayesIMP to CBO as well as to a simple GP with no learnt prior as baseline. We will be using $N = 100$ datapoints for $\mathcal{D}_1$ and $M = 50$ datapoints $\mathcal{D}_2$. We ran each method 10 times and plot the resulting standard deviation for each iteration in the figures below. The data generation and details were added in the Appendix. We see from the Fig.4.4.3 that BayesIMP is

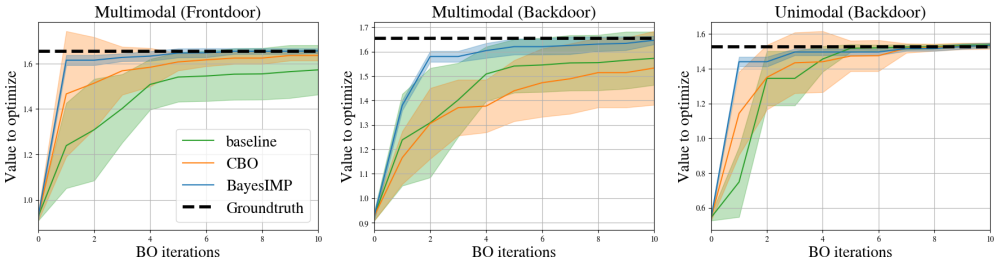

Figure 4.4.3: We are interested in finding the maximal value of $\mathbb{E}[T|do(X) = x]$ with as few BO iterations as possible. We ran experiments with **multimodality** in Y. (Left) Using front-door adjustment (Middle) Using backdoor adjustment (Right) Using backdoor adjustment (**unimodal** Y)

able to find the maxima much faster and with smaller standard deviations, than the current state-of-the-art method, CBO, using both front-door and backdoor adjustments (Fig.4.4.3(Right, Middle)). Given that our method uses more flexible representations

of conditional distributions, we are able to circumvent the multimodality problem in Y. In addition, we also consider the unimodal version, i.e. $\pi = 0$ (see right Fig.4.4.3). We see that the performance of CBO improves in the unimodal setting, however BayesIMP still converges faster than CBO even in this scenario.

Next, we consider a harder causal graph (see Fig.4.4.2 (Bottom)), previously considered in [Aglietti et al., 2020b]. We again introduce multimodality in the Y variable in order to explore the case of more challenging conditional densities. We see from Fig.4.4.4 (Left, Middle), that BayesIMP again converges much faster to the true optima than CBO [Aglietti et al., 2020b] and the standard GP prior baseline. We note that the fast convergence of BayesIMP throughout our experiments is not due to simplicity of the underlying BO problems. Indeed, the BO with a standard GP prior requires significantly more iterations. It is rather the availability of the observational data, allowing us to construct a more appropriate prior, which leads to a "warm" start of the BO procedure.

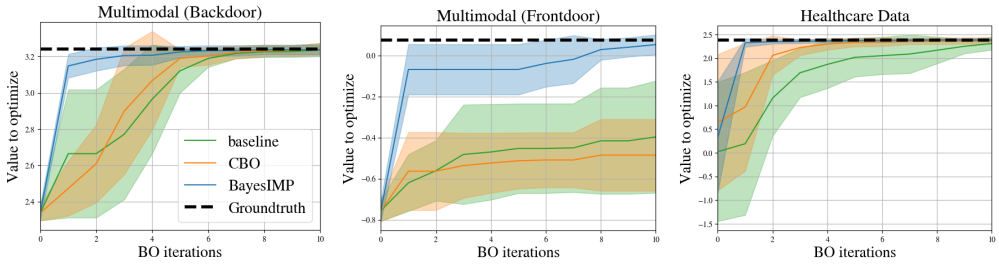

Figure 4.4.4: (Left) Experiments where we are interested in $\mathbb{E}[T|do(D) = d]$ with **multimodal** Y, (Middle) Experiments where we are interested in $\mathbb{E}[T|do(E) = e]$ with **multimodal** Y, (Right) Experiments on **healthcare data** where we are interested in $\mathbb{E}[\textit{Cancer Volume}|do(\textit{Statin})]$.

Healthcare experiments. We conclude with a healthcare dataset corresponding to our motivating medical example in Fig.4.1.1. The causal mechanism graph, also considered in the CBO paper [Aglietti et al., 2020b], studies the effect of certain drugs (Aspirin/Statin) on Prostate-Specific Antigen (PSA) levels [Ferro et al., 2015]. In our case, we modify *statin* to be continuous, in order to optimize for the correct drug dosage. However, in contrast to [Aglietti et al., 2020b], we consider a second experimental dataset, arising from a different medical study, which looks into the connection between *PSA* levels and *cancer volume* amount in patients [Stamey et al., 1989]. Similar to the original CBO paper [Aglietti et al., 2020b], given that interventional data is hard

to obtain, we construct data generators based on the true data collected in [Stamey et al., 1989]. This is done by firstly fitting a GP on the data and then sampling from the posterior (see Appendix for more details). Hence this is the perfect testbed for our model where we are interested in $\mathbb{E}[\textit{Cancer Volume}|\textit{do(Statin)}]$. We see from Fig.4.4.4 (Right) that BayesIMP again converges to the true optima faster than CBO hence allowing us to find the fastest ways of optimizng *cancer volume* by requesting much less interventional data. This could be critical as interventional data in real-life situations can be very expensive to obtain.

4.5 Discussion and Conclusion

In this chapter we propose BayesIMP for quantifying uncertainty in the setting of causal data fusion. In particular, our proposed method BayesIMP allows us to represent interventional densities in the RKHS without explicit density estimation, while still accounting for epistemic and aleatoric uncertainties. We demonstrated the quality of the uncertainty estimates in a variety of Bayesian optimization experiments, in both synthetic and real-world healthcare datasets, and achieve significant improvement over current SOTA in terms of convergence speed. However, we emphasize that BayesIMP is not designed to replace CBO but rather an alternative model for interventional effects.

In the future, we would like to improve BayesIMP over several limitations. As in [Aglietti et al., 2020b], we assumed full knowledge of the underlying causal graph, which might be limiting in practice. Furthermore, as the current formulation of BayesIMP only allows combination of two causal graphs, we hope to generalise the algorithm into arbitrary number of graphs in the future. Causal graphs with recurrent structure will be an interesting direction to explore. Lastly, we would also like to include a cost function as in [Aglietti et al., 2020b] to constrain the search space to the most sensible solutions.

4.6 Acknowledgements

The authors would like to thank Bobby He, Robert Hu, Kaspar Martens, Jake Fawkes and Joost van Amersfoort for helpful comments. SLC and JFT are supported by the EPSRC and MRC through the OxWaSP CDT programme EP/L016710/1. YWT and DS are supported in part by Tencent AI Lab and DS is supported in part by the Alan Turing Institute (EP/N510129/1). YWT's research leading to these results has

received funding from the European Research Council under the European Union's Seventh Framework Programme (FP7/2007-2013) ERC grant agreement no. 617071.

Appendix

4.A Additional background on backdoor/front-door adjustments

In causal inference, we are often times interested in the interventional distributions i.e $p(y|do(x))$ rather than $p(y|x)$, as the former allows us to account for confounding effects. In order to obtain the interventional density $p(y|do(x))$, we resort to *do*-calculus [Pearl, 1995]. Here below we write out the definition for the 2 most crucial formulaes; the front-door and backdoor adjustments, with which we are able to recover the interventional density using only the conditional ones.

4.A.1 Back-door Adjustment

The key intuition of back-door adjustments is to find/adjust a set of confounders that are unaffected by the treatment. We can then study the effect of the treatment has to the target.

Definition 7 (Back-Door)**.** *A set of variables Z satisfies the backdoor criterion relative to an ordered pair of variables X_i, X_j in a DAG G if:*

1. *no node in Z is a descendant of X_i; and*

2. *Z blocks every path between X_i and X_j that contains an arrow into X_i*

Similarly, if X and Y are two disjoint subsets of nodes in G, then Z is said satisfy the back-door criterion relative to (X, Y) if it satisfies the criterion relative to any pair (X_i, X_j) such that $X_i \in X$ and $X_j \in Y$

Now with a given set Z that satisfies the back-door criterion, we apply the backdoor adjustment,

Theorem 3 (Back-Door Adjustment)**.** *If a set of variables Z satisfies the back-door criterion relative to (X, Y), then the causal effect of X on Y is identifiable and is given by the formula*

$$P(y|do(X) = x) = \int_z p(y|x, z)p(z)dz \tag{4.18}$$

4.A.2 Front-door Adjustment

Front-door adjustment deals with the case where confounders are unobserved and hence the backdoor adjustment is not applicable.

Definition 8 (Front-door)**.** *A set of variables Z is said to satisfy the front-door criterion relative to an ordered pair of variables (X, Y) if:*

1. *Z intercepts all directed paths from X to Y;*
2. *there is no back-door path from X to Z; and*
3. *all back-door paths from Z to Y are blocked by X*

Again, with an appropriate front-door adjustment set Z, we can identify the do density using the front-door adjustment formula.

Theorem 4 (Front-Door Adjustment)**.** *If Z satisfies the front-door criterion relative to (X, Y) and if $P(x, z) > 0$, then the causal effect of X on Y is identifiable and is given by the formula:*

$$p(y|do(X) = x) = \int_z p(z|x) \int_{x'} p(y|x', z)p(x')dx'dz \tag{4.19}$$

4.B Derivations

4.B.1 CMP Derivation

Proposition 2. *Given dataset* $D_1 = \{(x_i, y_i, z_i)\}_{i=1}^N$ *and* $D_2 = \{(\tilde{y}_j, t_j)\}_{j=1}^M$*, if* f *is the posterior GP learnt from* $\mathcal{D}_2$*, then* $g = \int f(y)p(y|do(X))dy$ *is a GP* $\mathcal{GP}(m_1, \kappa_1)$ *defined on the treatment variable* X *with the following mean and covariance estimated using* $\hat{\mu}_{Y|do(X)}$ *,*

$$m_1(x) = \langle \hat{\mu}_{Y|do(x)}, m_f \rangle_{\mathcal{H}_{k_y}} = \Phi_{\Omega_x}(x)^\top (K_{\Omega_x} + \lambda I)^{-1} K_{\mathbf{y}\tilde{\mathbf{y}}} (K_{\tilde{\mathbf{y}}\tilde{\mathbf{y}}} + \lambda_f I)^{-1} \mathbf{t} \tag{4.20}$$

$$\kappa_1(x, x') = \hat{\mu}_{Y|do(x)}^\top \hat{\mu}_{Y|do(x')} - \hat{\mu}_{Y|do(x)}^\top \Phi_{\tilde{\mathbf{y}}} (K_{\tilde{\mathbf{y}}\tilde{\mathbf{y}}} + \lambda I)^{-1} \Phi_{\tilde{\mathbf{y}}}^\top \hat{\mu}_{Y|do(x')} \tag{4.21}$$

$$= \Phi_{\Omega_x}(x)^\top (K_{\Omega_x} + \lambda I)^{-1} \tilde{K}_{\mathbf{yy}} (K_{\Omega_x} + \lambda I)^{-1} \Phi_{\Omega_x}(x') \tag{4.22}$$

where $\hat{\mu}_{Y|do(x)} = \hat{\mu}_{Y|do(X)=x}$*,* $K_{\tilde{\mathbf{y}}\mathbf{y}} = \Phi_{\tilde{\mathbf{y}}}^\top \Phi_{\mathbf{y}}$*,* m_f *and* $\tilde{K}_{\mathbf{yy}}$ *are the posterior mean function and covariance of* f *evaluated at* $\mathbf{y}$ *respectively.* $\lambda > 0$ *is the regularisation of the CME.* $\lambda_f > 0$ *is the noise term for GP* f*.* Ω_x *is the set of variables as specified in Prop.1.*

Proof for Proposition 2. Integral operator preserves Gaussianity under mild conditions (see conditions [Chau et al., 2021a]), therefore

$$g(x) = \int f(y) dP(y|do(X) = x) \tag{4.23}$$

is also a Gaussian. For a standard GP prior $f \sim GP(0, k_y)$ and data $D_E = \{(\tilde{y}_j, t_j)\}_{j=1}^M$, standard conjugacy results for GPs lead to the posterior GP with mean $\bar{m}(y) = k_{y\tilde{\mathbf{y}}}(K_{\tilde{\mathbf{y}}\tilde{\mathbf{y}}} + \lambda_f I)^{-1}\mathbf{t}$ and covariance $\bar{k_y}(y, y') = k_y(y, y') - k_{y\tilde{\mathbf{y}}}(K_{\tilde{\mathbf{y}}\tilde{\mathbf{y}}} + \lambda_f I)^{-1} k_{\tilde{\mathbf{y}}y}$. Similar to [Briol et al., 2019], repeated application of Fubini's theorem yields:

$$\mathbb{E}_f[g(x)] = \mathbb{E}_f\left[\int f(y) dP(y|do(X) = x)\right] = \int \mathbb{E}_f[f(y)] dP(y|do(X) = x) \tag{4.24}$$

$$= \int \bar{m}(y) dP(y|do(X) = x) = \langle \bar{m}, \hat{\mu}_{Y|do(X)=x} \rangle \tag{4.25}$$

$$cov(g(x), g(x')) = \int\int cov(f(y), f(y')) dP(y|do(X) = x) dP(y'|do(X) = x) \tag{4.26}$$

$$= \int\int \bar{k}_y(y, y') dP(y|do(x)) dP(y'|do(x')) \tag{4.27}$$

$$= \langle \mu_{Y|do(x)}, \mu_{Y|do(x')} \rangle - \hat{\mu}_{Y|do(x)}^\top \Phi_{\tilde{\mathbf{y}}} (K_{\tilde{\mathbf{y}}\tilde{\mathbf{y}}} + \lambda I)^{-1} \Phi_{\tilde{\mathbf{y}}}^\top \hat{\mu}_{Y|do(x')} \tag{4.28}$$

$$= \Phi_{\Omega_x}(x)^\top (K_{\Omega_x} + \lambda I)^{-1} \tilde{K}_{\mathbf{yy}} (K_{\Omega_x} + \lambda I)^{-1} \Phi_{\Omega_x}(x') \tag{4.29}$$

□

4.B.2 Choice of Nuclear Dominant Kernel

Recall in section 4.3.2, we introduced the nuclear dominant kernel r_y to ensure samples of $\mu_{gp} \sim GP(0, k_x \otimes r_y)$ are supported in $\mathcal{H}_{k_x} \otimes \mathcal{H}_{k_y}$ with probability 1. In the following we will present the analytic form of the nuclear dominant kernel we used in this paper, which is the same as the formulation introduced in Appendix A.2 and A.3 of [Flaxman et al., 2016]. Pick k_y as the RBF kernel, i.e

$$k_y(y, y') = \exp\Big(-\frac{1}{2}(y - y')^\top \Sigma_\theta (y - y')\Big) \tag{4.30}$$

where Σ_θ is covariance matrix for the kernel k_y. The nuclear dominant kernel construction from [Flaxman et al., 2016] then yield the following expression:

$$r_y(y, y') = \int k_y(y, u) k_y(u, y') \nu(du) \tag{4.31}$$

where ν is some finite measure. If we pick $\nu(du) = \exp(\frac{||u||_2^2}{2\eta^2})du$, then we have

$$r_y(y, y') = (2\pi)^{D/2} |2\Sigma_\theta^{-1} + \eta^{-2} I|^{-1/2} \exp\Big(-\frac{1}{2}(y - y')^\top (2\Sigma_\theta)^{-1}(y - y')\Big) \tag{4.32}$$

$$\times \exp\Big(-\frac{1}{2}\Big(\frac{y + y'}{2}\Big)^\top \Big(\frac{1}{2}\Sigma_\theta + \eta^2 I\Big)^{-1}\Big(\frac{y + y'}{2}\Big)\Big) \tag{4.33}$$

4.B.3 BayesCME derivations

Proposition 3. *The posterior GP of μ_{gp} given observations $\{\mathbf{x}, \mathbf{y}\}$ has the following mean and covariance:*

$$m_\mu((x, y)) = k_{x\mathbf{x}}(K_{\mathbf{xx}} + \lambda I)^{-1} K_{\mathbf{yy}} R_{\mathbf{yy}}^{-1} r_{\mathbf{y}y} \tag{4.34}$$

$$\kappa_\mu((x, y), (x', y')) = k_{xx'} r_{y,y'} - k_{x\mathbf{x}}(K_{\mathbf{xx}} + \lambda I)^{-1} k_{\mathbf{x}x'} r_{y\mathbf{y}} R_{\mathbf{yy}}^{-1} r_{\mathbf{y}y'} \tag{4.35}$$

In addition, the following marginal likelihood can be used for hyperparameter optimisation,

$$-\frac{N}{2}\Big(\log|K_{\mathbf{xx}} + \lambda I| + \log|R|\Big) - \frac{1}{2}\operatorname{tr}\Big((K_{\mathbf{xx}} + \lambda I)^{-1} K_{\mathbf{yy}} R_{\mathbf{yy}}^{-1} K_{\mathbf{yy}}\Big) \tag{4.36}$$

Proof of Proposition 3. Recall the Bayesian formulation of CME corresponds to the following model,

$$\begin{aligned} \mu_{gp} &\sim GP\left(0, k_x \otimes r_y\right), \\ k_y\left(y_i, y'\right) &= \mu_{gp}\left(x_i, y'\right) + \lambda^{1/2}\epsilon_i\left(y'\right) \end{aligned}$$

with $\epsilon_i \sim GP(0, r_y)$ independently across i. Now consider $k_y(y_i, y_j)$ as noisy evaluations of $\mu_{gp}(x_i, y_j)$, we have the predictive posterior mean as

$$\begin{aligned}
&\operatorname{vec}\left(r_{\mathbf{yy}} k_{x\mathbf{x}}\right)^{\top} \left(K_{\mathbf{xx}} \otimes R_{\mathbf{yy}} + \lambda I \otimes R_{\mathbf{yy}}\right)^{-1} \operatorname{vec}(K_{\mathbf{yy}}) \\
&= \operatorname{vec}\left(r_{\mathbf{yy}} k_{x\mathbf{x}}\right)^{\top} \left(\left(K_{\mathbf{xx}} + \lambda I\right)^{-1} \otimes R_{\mathbf{yy}}^{-1}\right) \operatorname{vec}(K_{\mathbf{yy}}) \\
&= \operatorname{vec}\left(r_{\mathbf{yy}} k_{x\mathbf{x}}\right)^{\top} \operatorname{vec}\left(R_{\mathbf{yy}}^{-1} K_{\mathbf{yy}} \left(K_{\mathbf{xx}} + \lambda I\right)^{-1}\right) \\
&= \operatorname{tr}\left(r_{\mathbf{yy}} k_{x\mathbf{x}} \left(K_{\mathbf{xx}} + \lambda I\right)^{-1} K_{\mathbf{yy}} R_{\mathbf{yy}}^{-1}\right) \\
&= k_{x\mathbf{x}} \left(K_{\mathbf{xx}} + \lambda I\right)^{-1} K_{\mathbf{yy}} R_{\mathbf{yy}}^{-1} r_{\mathbf{y}y}.
\end{aligned}$$

And the covariance is,

$$\begin{aligned}
\kappa\left((x,y),(x',y')\right) &= k(x,x')r(y,y') - \operatorname{vec}\left(r_{\mathbf{yy}} k_{x\mathbf{x}}\right)^{\top} \left(K_{\mathbf{xx}} \otimes R_{\mathbf{yy}} + \lambda I \otimes R_{\mathbf{yy}}\right)^{-1} \operatorname{vec}\left(r_{\mathbf{y}y'} k_{x'\mathbf{x}}\right) \\
&= k(x,x')r(y,y') - \operatorname{vec}\left(r_{\mathbf{yy}} k_{x\mathbf{x}}\right)^{\top} \left(\left(K_{\mathbf{xx}} + \lambda I\right)^{-1} \otimes R_{\mathbf{yy}}^{-1}\right) \operatorname{vec}\left(r_{\mathbf{y}y'} k_{x'\mathbf{x}}\right) \\
&= k(x,x')r(y,y') - \operatorname{vec}\left(r_{\mathbf{yy}} k_{x\mathbf{x}}\right)^{\top} \operatorname{vec}\left(R_{\mathbf{yy}}^{-1} r_{\mathbf{y}y'} k_{x'\mathbf{x}} \left(K_{\mathbf{xx}} + \lambda I\right)^{-1}\right) \\
&= k(x,x')r(y,y') - \operatorname{tr}\left(r_{\mathbf{yy}} k_{x\mathbf{x}} \left(K_{\mathbf{xx}} + \lambda I\right)^{-1} k_{\mathbf{x}x'} r_{y'\mathbf{y}} R_{\mathbf{yy}}^{-1}\right) \\
&= k(x,x')r(y,y') - k_{x\mathbf{x}} \left(K_{\mathbf{xx}} + \lambda I\right)^{-1} k_{\mathbf{x}x'} r_{y'\mathbf{y}} R_{\mathbf{yy}}^{-1} r_{\mathbf{y}y}.
\end{aligned}$$

To compute the log likelihood, note that it contains the following two terms:

$$\begin{aligned}
\operatorname{vec}(K_{\mathbf{yy}})^{\top} \left(K_{\mathbf{xx}} \otimes R_{\mathbf{yy}} + \lambda I \otimes R_{\mathbf{yy}}\right)^{-1} \operatorname{vec}(K_{\mathbf{yy}}) &= \operatorname{vec}(K_{\mathbf{yy}})^{\top} \left(\left(K_{\mathbf{xx}} + \lambda I\right)^{-1} \otimes R_{\mathbf{yy}}^{-1}\right) \operatorname{vec}(K_{\mathbf{yy}}) \\
&= \operatorname{vec}(K_{\mathbf{yy}})^{\top} \operatorname{vec}\left(R_{\mathbf{yy}}^{-1} K_{\mathbf{yy}} \left(K_{\mathbf{xx}} + \lambda I\right)^{-1}\right) \\
&= \operatorname{tr}\left(K_{\mathbf{yy}} \left(K_{\mathbf{xx}} + \lambda I\right)^{-1} K_{\mathbf{yy}} R_{\mathbf{yy}}^{-1}\right)
\end{aligned}$$

and

$$\begin{aligned}
-\frac{1}{2}\Big(\log\left|\left(K_{\mathbf{xx}} + \lambda I\right) \otimes R_{\mathbf{yy}}\right|\Big) &= -\frac{1}{2}\log\left(\left|\left(K_{\mathbf{xx}} + \lambda I\right)\right|^N |R|^N\right) \\
&= -\frac{N}{2}\Big(\log\left|K_{\mathbf{xx}} + \lambda I\right| + \log|R|\Big)
\end{aligned}$$

where we used the fact that determinant of Kronecker product of two $N \times N$ matrices A, B is: $|A \otimes B| = |A|^N |B|^N$.

Therefore the log likelihood can be expressed as

$$-\frac{N}{2}\Big(\log\left|K_{\mathbf{xx}} + \lambda I\right| + \log|R|\Big) - \frac{1}{2}\operatorname{tr}\left(\left(K_{\mathbf{xx}} + \lambda I\right)^{-1} K_{\mathbf{yy}} R_{\mathbf{yy}}^{-1} K_{\mathbf{yy}}\right) \tag{4.37}$$

□

4.B.4 Causal BayesCME derivations

The following proposition extend BayesCME to the causal setting.

Proposition C.1 (Causal BayesCME)**.** *Denote μ_{gp}^{do} as the GP modelling $\mu_{Y|do(X)}$. Then using the Ω notations introduced in proposition 1, the posterior GP of μ_{gp}^{do} given observations $\{\mathbf{x}, \mathbf{z}, \mathbf{y}\}$ has the following mean and covariance:*

$$m_{\mu}^{do}((x,y)) = \Phi_{\Omega_x}(x)^{\top}\Big(K_{\Omega_x} + \lambda I\Big)^{-1} K_{\mathbf{yy}} R_{\mathbf{yy}}^{-1} r_{\mathbf{y}y} \tag{4.38}$$

$$\kappa_{\mu}^{do}((x,y),(x',y')) = \Phi_{\Omega_x}(x)^{\top}\Phi_{\Omega_x}(x') r_{y,y'} - \Phi_{\Omega_x}(x)^{\top}(K_{\Omega_x} + \lambda I)^{-1}\Phi_{\Omega_x}(x') r_{y\mathbf{y}} R_{\mathbf{yy}}^{-1} r_{\mathbf{y}y'} \tag{4.39}$$

In addition, the following marginal likelihood can be used for hyperparameter optimisation,

$$-\frac{N}{2}\Big(\log|K_{\Omega_x} + \lambda I| + \log|R|\Big) - \frac{1}{2}\operatorname{tr}\Big((K_{\Omega_x} + \lambda I)^{-1} K_{\mathbf{yy}} R_{\mathbf{yy}}^{-1} K_{\mathbf{yy}}\Big) \tag{4.40}$$

Proof of Proposition C.1. In the following we will assume Z is the backdoor adjustment variable. Front-door and general cases follow analogously. Denote $\mu_{gp}((x,z),y)$ as the BayesCME model for $\mu_{Y|X=x,Z=z}(y)$. As we have

$$\mu_{Y|do(X)=x} = \int\int \phi_y(y) p(y|x,z) p(z) dz dy \tag{4.41}$$

$$= \int \mu_{Y|X=x,Z=z} p(z) dz \tag{4.42}$$

$$= \mathbb{E}_Z[\mu_{Y|X=x,Z}] \tag{4.43}$$

It is thus natural to define μ_{gp}^{do} as the induced GP when we replace $\mu_{Y|X=x,Z=z}$ with $\mu_{gp}((x,z),\cdot)$,

$$\mu_{gp}^{do}(x,\cdot) = \mathbb{E}_Z[\mu_{gp}((x,Z),\cdot)] \tag{4.44}$$

Now we can compute the mean of μ_{gp}^{do},

$$m_{\mu}^{do}(x,y) = \mathbb{E}_{\mu_{gp}}\mathbb{E}_Z[\mu_{gp}(x,Z,y)] \tag{4.45}$$

$$= \mathbb{E}_Z\Big((k_{x\mathbf{x}} \odot k_z(Z,\mathbf{z}))(K_{\mathbf{xx}} \odot K_{\mathbf{zz}} + \lambda I)^{-1} K_{\mathbf{yy}} R_{\mathbf{yy}}^{-1} r_{\mathbf{y}y}\Big) \tag{4.46}$$

$$= \Big((k_{x\mathbf{x}} \odot \mu_z^{\top}\Phi_{\mathbf{z}})(K_{\mathbf{xx}} \odot K_{\mathbf{zz}} + \lambda I)^{-1} K_{\mathbf{yy}} R_{\mathbf{yy}}^{-1} r_{\mathbf{y}y}\Big) \tag{4.47}$$

$$= \Phi_{\Omega_x}(x)^{\top}(K_{\Omega_x} + \lambda I)^{-1} K_{\mathbf{yy}} R_{\mathbf{yy}}^{-1} r_{\mathbf{y}y} \tag{4.48}$$

Similarly for covariance, we have,

$$\kappa_{\mu}^{do}((x,y),(x',y')) = \mathbb{E}_{Z,Z'}[cov\Big(\mu_{gp}((x,Z),y),\mu_{gp}((x',Z'),y')\Big)] \tag{4.49}$$

and the rest is just algebra,

$$= \Phi_{\Omega_x}(x)^\top \Phi_{\Omega_x}(x') r_{y,y'} - \Phi_{\Omega_x}(x)^\top (K_{\Omega_x} + \lambda I)^{-1} \Phi_{\Omega_x}(x') r_{y\mathbf{y}} R_{\mathbf{yy}}^{-1} r_{\mathbf{y}y'} \tag{4.50}$$

□

4.B.5 BayesIME derivation

Now we have derived the Causal BayesCME, it is time to compute $\langle f, \mu_{gp}^{do}(x,\cdot)\rangle$ where $f \in \mathcal{H}_{k_y}$. This requires us to be able to compute $\langle f, r_y(\cdot,y)\rangle$ which corresponds to the following:

$$\langle f, r_y(\cdot,y)\rangle_{\mathcal{H}_{k_y}} = \Big\langle f, \int k_y(\cdot,u)k_y(u,y)\nu(du)\Big\rangle \tag{4.51}$$

$$= \int f(u)k_y(u,y)\nu(du) \tag{4.52}$$

when f is a KRR learnt from $\mathcal{D}_2$, i.e $f(y) = k_{y\tilde{\mathbf{y}}}(K_{\tilde{\mathbf{y}}\tilde{\mathbf{y}}} + \lambda_f I)^{-1}\mathbf{t}$, we have

$$= \mathbf{t}^\top (K_{\tilde{\mathbf{y}}\tilde{\mathbf{y}}} + \lambda_f I)^{-1} \int k_{\tilde{\mathbf{y}}u} k_y(u,y)\nu(du) \tag{4.53}$$

$$= \mathbf{t}^\top (K_{\tilde{\mathbf{y}}\tilde{\mathbf{y}}} + \lambda_f I)^{-1} r_{\tilde{\mathbf{y}}y} \tag{4.54}$$

Now we are ready to derive BayesIME.

Proposition 4. *Given dataset $D_1 = \{(x_i,y_i,z_i)\}_{i=1}^N$ and $D_2 = \{(\tilde{y}_j,t_j)\}_{j=1}^M$, if f is a KRR learnt from $\mathcal{D}_2$ and $\mu_{Y|do(X)}$ modelled as a V-GP using $\mathcal{D}_1$, then $g = \langle f, \mu_{Y|do(X)}\rangle \sim \mathcal{GP}(m_2,\kappa_2)$ where,*

$$m_2(x) = \Phi_{\Omega_x}(x)^\top (K_{\Omega_x} + \lambda I)^{-1} K_{\mathbf{yy}} R_{\mathbf{yy}}^{-1} R_{\mathbf{y}\tilde{\mathbf{y}}} A \tag{4.55}$$

$$\kappa_2(x,x') = B\Phi_{\Omega_x}(x)^\top \Phi_{\Omega_x}(x) - C\Phi_{\Omega_x}(x)^\top (K_{\Omega_x} + \lambda I)^{-1} \Phi_{\Omega_x}(x') \tag{4.56}$$

where $A = (K_{\tilde{\mathbf{y}}\tilde{\mathbf{y}}} + \lambda_f I)^{-1}\mathbf{t}$, $B = A^\top R_{\tilde{\mathbf{y}}\tilde{\mathbf{y}}} A$ and $C = A^\top R_{\tilde{\mathbf{y}}\mathbf{y}} R_{\mathbf{yy}}^{-1} R_{\mathbf{y}\tilde{\mathbf{y}}} A$

Proof of Proposition 4. Using the μ_{gp}^{do} notation from Proposition C.1, we can write the inner product as $\langle \mu_{gp}^{do}(x,\cdot), f\rangle$, where the mean is,

$$m_2(x) = \mathbb{E}[\mu_{gp}^{do}(x,\cdot)]^\top f \tag{4.57}$$

$$= \Phi_{\Omega_x}(x)^\top (K_{\Omega_x} + \lambda I)^{-1} K_{\mathbf{yy}} R_{\mathbf{yy}}^{-1} R(\mathbf{y},\cdot)^\top f \tag{4.58}$$

$$= \Phi_{\Omega_x}(x)^\top (K_{\Omega_x} + \lambda I)^{-1} K_{\mathbf{yy}} R_{\mathbf{yy}}^{-1} R_{\mathbf{y}\tilde{\mathbf{y}}} (K_{\tilde{\mathbf{y}}\tilde{\mathbf{y}}} + \lambda_f I)^{-1}\mathbf{t} \tag{4.59}$$

where we used the fact f is a KRR learnt from $\mathcal{D}_2$. The covariance can then be computed by realising $cov(f^\top \mu^{do}_{gp}(x,\cdot), f^\top \mu^{do}_{gp}(x',\cdot)) = f^\top cov(\mu^{do}_{gp}(x,\cdot), \mu^{do}_{gp}(x',\cdot))f$. □

4.B.6 BayesIMP Derivations

BayesIMP can be understood as a model characterising the RKHS inner product of Gaussian Processes. In the following, we will first introduce some general theory of inner product of GPs, and introduce a finite dimensional scheme later on. Finally, we will show how BayesIMP can be derived right away from this general framework.

Before that, we will showcase the following identity for computing variance of inner products of independent multivariate Gaussians,

Proposition C.2. *Let $\mu_X := \mathbb{E}[X]$ and $\Sigma_X := Var(X)$ be the mean and variance of a multivariate Gaussian rv, similarly μ_Y, Σ_Y for Gaussian rv Y. If X and Y are independent, then the variance of their inner product is given by the following expression,*

$$Var(X^\top Y) = \mu_X^\top \Sigma_Y \mu_X + \mu_Y^\top \Sigma_X \mu_Y + tr\Big(\Sigma_Y \Sigma_X\Big) \tag{4.60}$$

Moreover, the covariance between $X^\top Y_1$, $X^\top Y_2$ follows a similar form,

$$cov(X^\top Y_1, X^\top Y_2) = \mu_X^\top \Sigma_{Y_1 Y_2} \mu_X + \mu_{Y_1}^\top \Sigma_X \mu_{Y_2} + \mathrm{tr}(\Sigma_X \Sigma_{Y_1 Y_2}) \tag{4.61}$$

Proof.

$$\begin{aligned}
\mathrm{Var}\left[X^\top Y\right] &= \mathbb{E}\left[\left(X^\top Y\right)^2\right] - \mathbb{E}\left[X^\top Y\right]^2 \\
&= \mathbb{E}\left[X^\top Y Y^\top X\right] - \left(\mathbb{E}[X]^\top \mathbb{E}[Y]\right)^2 \\
&= \mathbb{E}\left[\mathrm{tr}\left(X X^\top Y Y^\top\right)\right] - \left(\mu_X^\top \mu_Y\right)^2 \\
&= \mathrm{tr}\left(\mathbb{E}\left[X X^\top\right] \mathbb{E}\left[Y Y^\top\right]\right) - \left(\mu_X^\top \mu_Y\right)^2 \\
&= \mathrm{tr}\left(\left(\mu_X \mu_X^\top + \Sigma_X\right)\left(\mu_Y \mu_Y^\top + \Sigma_Y\right)\right) - \left(\mu_X^\top \mu_Y\right)^2 \\
&= \mathrm{tr}\left(\mu_X \mu_X^\top \mu_Y \mu_Y^\top\right) + \mathrm{tr}\left(\mu_X \mu_X^\top \Sigma_Y\right) + \mathrm{tr}\left(\Sigma_X \mu_Y \mu_Y^\top\right) + \mathrm{tr}\left(\Sigma_X \Sigma_Y\right) - \left(\mu_X^\top \mu_Y\right)^2 \\
&= \left(\mu_X^\top \mu_Y\right)^2 + \mathrm{tr}\left(\mu_X^\top \Sigma_Y \mu_X\right) + \mathrm{tr}\left(\mu_Y^\top \Sigma_X \mu_Y\right) + \mathrm{tr}\left(\Sigma_X \Sigma_Y\right) - \left(\mu_X^\top \mu_Y\right)^2 \\
&= \mu_X^\top \Sigma_Y \mu_X + \mu_Y^\top \Sigma_X \mu_Y + \mathrm{tr}\left(\Sigma_X \Sigma_Y\right)
\end{aligned} \tag{4.62}$$

Generalising to the case for covariance is straight forward. □

RKHS inner product of Gaussian Processes

Let $f_1 \sim GP(m_1, \kappa_1)$ and $f_2 \sim GP(m_2, \kappa_2)$. We assume that f and g are both supported within the RKHS $\mathcal{H}_k$. Can we characterise the distribution of $\langle f_1, f_2 \rangle_{\mathcal{H}_k}$?

This situation would arise if f_1 and f_2 arise as GP posteriors in a regression model corresponding to the priors $f_1 \sim GP(0, r_1)$, $f_2 \sim GP(0, r_2)$ where r_1, r_2 satisfy the nuclear dominance property. In particular, we could choose

$$\begin{aligned} r_1(u,v) &= \int k(u,z)\, k(z,v)\, \nu_1(dz), \\ r_2(u,v) &= \int k(u,z)\, k(z,v)\, \nu_2(dz). \end{aligned}$$

Posterior means in that case can be expanded as

$$m_1 = \sum \alpha_i r_1(\cdot, x_i), \qquad m_2 = \sum \beta_j r_2(\cdot, y_j).$$

We assume that f_1 and f_2 are independent, i.e. they correspond to posteriors computed on independent data. Then

$$\begin{aligned} \mathbb{E}\langle f_1, f_2 \rangle_{\mathcal{H}_k} &= \langle m_1, m_2 \rangle_{\mathcal{H}_k} \\ &= \left\langle \sum \alpha_i r_1(\cdot, x_i), \sum \beta_j r_2(\cdot, y_j) \right\rangle_{\mathcal{H}_k} \\ &= \alpha^\top Q \beta, \end{aligned}$$

where

$$\begin{aligned} Q_{ij} = q(x_i, y_j) &:= \langle r_1(\cdot, x_i), r_2(\cdot, y_j) \rangle_{\mathcal{H}_k} \\ &= \left\langle \int k(\cdot, z)\, k(z, x_i)\, \nu_1(dz), \int k(\cdot, z')\, k(z', y_j)\, \nu_2(dz') \right\rangle_{\mathcal{H}_k} \\ &= \int\int \langle k(\cdot, z), k(\cdot, z') \rangle_{\mathcal{H}_k}\, k(z, x_i)\, k(z', y_j)\, \nu_1(dz)\, \nu_2(dz') \\ &= \int\int k(z, z')\, k(z, x_i)\, k(z', y_j)\, \nu_1(dz)\, \nu_2(dz'). \end{aligned}$$

The variance would be given, in analogy to the finite dimensional case, by

$$\operatorname{var}\langle f_1, f_2 \rangle_{\mathcal{H}_k} = \langle m_1, \Sigma_2 m_1 \rangle_{\mathcal{H}_k} + \langle m_2, \Sigma_1 m_2 \rangle_{\mathcal{H}_k} + \operatorname{tr}(\Sigma_1 \Sigma_2),$$

with $\Sigma_1 f = \int \kappa_1(\cdot, u) f(u) du$ and similarly for Σ_2. Thus

$$\begin{aligned} \langle m_1, \Sigma_2 m_1 \rangle_{\mathcal{H}_k} &= \left\langle \sum \alpha_i r_1(\cdot, x_i), \sum \alpha_j \int \kappa_2(\cdot, u)\, r_1(u, x_j)\, du \right\rangle_{\mathcal{H}_k} \\ &= \sum\sum \alpha_i \alpha_j \int \langle r_1(\cdot, x_i), \kappa_2(\cdot, u) \rangle_{\mathcal{H}_k}\, r_1(u, x_j)\, du. \end{aligned}$$

Now, given that kernel κ_2 depends on r_2 in a simple way, it should be possible to write down the full expression similarly as for Q_{ij} above. In particular

$$\kappa_2(\cdot, u) = r_2(\cdot, u) - r_2(\cdot, \mathbf{y}) \left(R_{2,\mathbf{yy}} + \sigma_2^2 I\right)^{-1} r_2(\mathbf{y}, u).$$

Hence

$$\langle r_1(\cdot, x_i), \kappa_2(\cdot, u)\rangle_{\mathcal{H}_k} = q(x_i, u) - q(x_i, \mathbf{y}) \left(R_{2,\mathbf{yy}} + \sigma_2^2 I\right)^{-1} r_2(\mathbf{y}, u).$$

However, this further requires approximating integrals of the type

$$\int q(x_i, u)\, r_1(u, x_j)\, du =$$

$\int\int\int\int k(z, z')\, k(z, x_i)\, k(z', u)\, k(u, z'')\, k(z'', x_j)\, \nu_1(dz)\, \nu_2(dz')\, \nu_1(dz'')\, du,$

etc. Thus, while possible in principle, this approach to compute the variance is cumbersome.

A finite dimensional approximation

To approximate the variance, hence, it is simpler to consider finite-dimensional approximations to f_1 and f_2. Namely, collate $\{x_i\}$ and $\{y_j\}$ into a single set of points ξ (note that we could here take an arbitrary set of points), and consider finite-dimensional GPs given by

$$\tilde{f}_1 = \sum a_j k(\cdot, \xi_j), \quad \tilde{f}_2 = \sum b_j k(\cdot, \xi_j),$$

where we selects distribution of a and b such that evaluations of $\tilde{f}_1$ and $\tilde{f}_2$ on ξ, $K_{\xi\xi}a$ and $K_{\xi\xi}b$ respectively, have the same distributions as evaluations of f_1 and f_2 on ξ. In particular, we take

$$a \sim \mathcal{N}\left(K_{\xi\xi}^{-1} m_1(\xi), K_{\xi\xi}^{-1} \mathcal{K}_{1,\xi\xi} K_{\xi\xi}^{-1}\right), \quad b \sim \mathcal{N}\left(K_{\xi\xi}^{-1} m_2(\xi), K_{\xi\xi}^{-1} \mathcal{K}_{2,\xi\xi} K_{\xi\xi}^{-1}\right),$$

where we denoted by $m_1(\xi)$ a vector such that $[m_1(\xi)]_i = m_1(\xi_i)$ and by $\mathcal{K}_{1,\xi\xi}$ a matrix such that $[\mathcal{K}_{1,\xi\xi}]_{ij} = \kappa_1(\xi_i, \xi_j)$.

Then, clearly

$$\begin{aligned} \left\langle \tilde{f}_1, \tilde{f}_2 \right\rangle_{\mathcal{H}_k} &= a^\top K_{\xi\xi} b \\ &= \left(K_{\xi\xi}^{1/2} a\right)^\top \left(K_{\xi\xi}^{1/2} b\right), \end{aligned}$$

and now we are left with the problem of computing the mean and the variance of inner product between two independent Gaussian vectors, as given in Proposition *C*.2. We have

$$\begin{aligned}\mathbb{E}\left\langle \tilde{f}_1, \tilde{f}_2 \right\rangle_{\mathcal{H}_k} &= \left(K_{\xi\xi}^{1/2} K_{\xi\xi}^{-1} m_1(\xi)\right)^\top \left(K_{\xi\xi}^{1/2} K_{\xi\xi}^{-1} m_2(\xi)\right) \\ &= m_1(\xi)^\top K_{\xi\xi}^{-1} K_{\xi\xi} K_{\xi\xi}^{-1} m_2(\xi) \\ &= m_1(\xi)^\top K_{\xi\xi}^{-1} m_2(\xi),\end{aligned}$$

and

$$\begin{aligned}\operatorname{var}\left\langle \tilde{f}_1, \tilde{f}_2 \right\rangle_{\mathcal{H}_k} &= \left(K_{\xi\xi}^{1/2} K_{\xi\xi}^{-1} m_1(\xi)\right)^\top K_{\xi\xi}^{-1/2} \mathcal{K}_{2,\xi\xi} K_{\xi\xi}^{-1/2} \left(K_{\xi\xi}^{1/2} K_{\xi\xi}^{-1} m_1(\xi)\right) \\ &+ \left(K_{\xi\xi}^{1/2} K_{\xi\xi}^{-1} m_2(\xi)\right)^\top K_{\xi\xi}^{-1/2} \mathcal{K}_{1,\xi\xi} K_{\xi\xi}^{-1/2} \left(K_{\xi\xi}^{1/2} K_{\xi\xi}^{-1} m_2(\xi)\right) \\ &+ \operatorname{tr}\left(K_{\xi\xi}^{-1/2} \mathcal{K}_{1,\xi\xi} K_{\xi\xi}^{-1/2} K_{\xi\xi}^{-1/2} \mathcal{K}_{2,\xi\xi} K_{\xi\xi}^{-1/2}\right) \\ &= m_1(\xi)^\top K_{\xi\xi}^{-1} \mathcal{K}_{2,\xi\xi} K_{\xi\xi}^{-1} m_1(\xi) \\ &+ m_2(\xi)^\top K_{\xi\xi}^{-1} \mathcal{K}_{1,\xi\xi} K_{\xi\xi}^{-1} m_2(\xi) \\ &+ \operatorname{tr}\left(\mathcal{K}_{1,\xi\xi} K_{\xi\xi}^{-1} \mathcal{K}_{2,\xi\xi} K_{\xi\xi}^{-1}\right).\end{aligned}$$

Coming back to BayesIMP

Now coming back to the derivation of BayesIMP. We will first provide two finite approximation of f and $\mu_{gp}^{do}(x, \cdot)$ in the following two propositions. Recall these finite approximations are set up such that they match the distributions of evaluations of f and μ_{gp}^{do} at $\hat{\mathbf{y}} = [\mathbf{y}^\top \tilde{\mathbf{y}}^\top]^\top$. The latter thus act as landmark points for the finite dimensional approximations.

Proposition C.3 (Finite dimensional approximation of f)**.** *Let* $\hat{\mathbf{y}} = [\mathbf{y}^\top \tilde{\mathbf{y}}^\top]^\top$ *be the concatenation of* $\mathbf{y}$ *and* $\tilde{\mathbf{y}}$*. We can approximate* f *with ,*

$$\tilde{f}|\mathbf{t} \sim N(m_{\tilde{f}}, \Sigma_{\tilde{f}}) \tag{4.63}$$

where,

$$m_{\tilde{f}} = \Phi_{\hat{\mathbf{y}}} K_{\hat{\mathbf{y}}\hat{\mathbf{y}}}^{-1} R_{\hat{\mathbf{y}}\tilde{\mathbf{y}}} (R_{\tilde{\mathbf{y}}\tilde{\mathbf{y}}} + \lambda_f I)^{-1} \mathbf{t} \tag{4.64}$$

$$\Sigma_{\tilde{f}} = \Phi_{\hat{\mathbf{y}}} K_{\hat{\mathbf{y}}\hat{\mathbf{y}}}^{-1} \bar{R}_{\hat{\mathbf{y}}\hat{\mathbf{y}}} K_{\hat{\mathbf{y}}\hat{\mathbf{y}}}^{-1} \Phi_{\hat{\mathbf{y}}}^\top \tag{4.65}$$

and $\bar{R}_{\hat{\mathbf{y}}\hat{\mathbf{y}}} = R_{\hat{\mathbf{y}}\hat{\mathbf{y}}} - R_{\hat{\mathbf{y}}\tilde{\mathbf{y}}} (R_{\tilde{\mathbf{y}}\tilde{\mathbf{y}}} + \lambda_f I)^{-1} R_{\tilde{\mathbf{y}}\hat{\mathbf{y}}}$.

Similarly for $\mu_{gp}^{do}(x,\cdot)$, we have the following

Proposition C.4 (Finite dimensional approximation of $\mu_{gp}^{do}(x,\cdot)$)**.** *Let* $\hat{\mathbf{y}} = [\mathbf{y}^\top\ \tilde{\mathbf{y}}^\top]^\top$ *be the concatenation of* $\mathbf{y}$ *and* $\tilde{\mathbf{y}}$*. We can approximate* $\mu_{gp}^{do}(x,\cdot)$ *with ,*

$$\tilde{\mu}_{gp}^{do}(x,\cdot)|\,\mathrm{vec}(K_{\mathbf{yy}}) \sim N(m_{\tilde{\mu}}, \Sigma_{\tilde{\mu}}) \tag{4.66}$$

where,

$$m_{\tilde{\mu}} = \Phi_{\hat{\mathbf{y}}} K_{\hat{\mathbf{y}}\hat{\mathbf{y}}}^{-1} R_{\hat{\mathbf{y}}\mathbf{y}} R_{\mathbf{yy}}^{-1} K_{\mathbf{yy}} (K_{\Omega_x} + \lambda I)^{-1} \Phi_{\Omega_x}(x) \tag{4.67}$$

$$\Sigma_{\tilde{\mu}} = \Phi_{\hat{\mathbf{y}}} K_{\hat{\mathbf{y}}\hat{\mathbf{y}}}^{-1} K_{\hat{\mathbf{y}}\hat{\mathbf{y}}}^{\mu} K_{\hat{\mathbf{y}}\hat{\mathbf{y}}}^{-1} \Phi_{\hat{\mathbf{y}}}^\top \tag{4.68}$$

where $K_{\hat{\mathbf{y}}\hat{\mathbf{y}}}^{\mu} = \Phi_{\Omega_x}(x)^\top \Phi_{\Omega_x}(x) R_{\hat{\mathbf{y}}\hat{\mathbf{y}}} - \Big(\Phi_{\Omega_x}(x)^\top (K_{\Omega_x} + \lambda I)^{-1} \Phi_{\Omega_x}(x)\Big) R_{\hat{\mathbf{y}}\mathbf{y}} R_{\mathbf{yy}}^{-1} R_{\mathbf{y}\hat{\mathbf{y}}}$

Now we have everything we need to derive the main algorithm in our paper, the BayesIMP. Note that we did not introduce the μ_{gp}^{do} notation in the main text to avoid confusion as we did not have space to properly define μ_{gp}^{do}.

Proposition 5 (**BayesIMP**)**.** *Let* f *and* $\mu_{Y|do(X)}$ *be GPs learnt as above. Denote* $\tilde{f}$ *and* $\tilde{\mu}_{Y|do(X)}$ *as the finite dimensional approximation of* f *and* $\mu_{Y|do(X)}$ *respectively. Then* $\tilde{g} = \langle \tilde{f}, \tilde{\mu}_{Y|do(X)} \rangle$ *has the following mean and covariance:*

$$m_3(x) = E_x K_{\mathbf{y}\hat{\mathbf{y}}} K_{\hat{\mathbf{y}}\hat{\mathbf{y}}}^{-1} R_{\hat{\mathbf{y}}\tilde{\mathbf{y}}} (R_{\tilde{\mathbf{y}}\tilde{\mathbf{y}}} + \lambda_f I)^{-1} \mathbf{t} \tag{4.69}$$

$$\kappa_3(x,x') = \underbrace{E_x \Theta_1^\top \tilde{R}_{\hat{\mathbf{y}}\hat{\mathbf{y}}} \Theta_1 E_{x'}^\top}_{\textit{Uncertainty from } \mathcal{D}_1} + \underbrace{\Theta_2^{(a)} F_{xx'} - \Theta_2^{(b)} G_{xx'}}_{\textit{Uncertainty from } \mathcal{D}_2} + \underbrace{\Theta_3^{(a)} F_{xx'} - \Theta_3^{(b)} G_{xx'}}_{\textit{Uncertainty from Interaction}} \tag{4.70}$$

where $E_x = \Phi_{\Omega_x}(x)^\top (K_{\Omega_x} + \lambda I)^{-1}, F_{xx'} = \Phi_{\Omega_x}(x)^\top \Phi_{\Omega_x}(x'), G_{xx'} = \Phi_{\Omega_x}(x)^\top (K_{\Omega_x} + \lambda I)^{-1} \Phi_{\Omega_x}(x')$*, and* $\Theta_1 = K_{\hat{\mathbf{y}}\hat{\mathbf{y}}}^{-1} R_{\hat{\mathbf{y}}\mathbf{y}} R_{\mathbf{yy}}^{-1} K_{\mathbf{yy}}$*,* $\Theta_2^{(a)} = \Theta_4^\top R_{\hat{\mathbf{y}}\hat{\mathbf{y}}} \Theta_4, \Theta_2^{(b)} = \Theta_4^\top R_{\hat{\mathbf{y}}\mathbf{y}} R_{\mathbf{yy}}^{-1} R_{\mathbf{y}\hat{\mathbf{y}}} \Theta_4$ *and* $\Theta_3^{(a)} = tr(K_{\hat{\mathbf{y}}\hat{\mathbf{y}}}^{-1} R_{\hat{\mathbf{y}}\hat{\mathbf{y}}} K_{\hat{\mathbf{y}}\hat{\mathbf{y}}}^{-1} \bar{R}_{\hat{\mathbf{y}}\hat{\mathbf{y}}}), \Theta_3^{(b)} = tr(R_{\hat{\mathbf{y}}\mathbf{y}} R_{\mathbf{yy}}^{-1} R_{\mathbf{y}\hat{\mathbf{y}}} K_{\hat{\mathbf{y}}\hat{\mathbf{y}}}^{-1} \bar{R}_{\hat{\mathbf{y}}\hat{\mathbf{y}}} K_{\hat{\mathbf{y}}\hat{\mathbf{y}}}^{-1})$ *and* $\Theta_4 = K_{\hat{\mathbf{y}}\hat{\mathbf{y}}}^{-1} R_{\hat{\mathbf{y}}\tilde{\mathbf{y}}} (K_{\tilde{\mathbf{y}}\tilde{\mathbf{y}}} + \lambda_f)^{-1} \mathbf{t}$*.* $\bar{R}_{\hat{\mathbf{y}}\hat{\mathbf{y}}}$ *is the posterior covariance of* f *evaluated at* $\hat{\mathbf{y}}$

Proof of Proposition 5. Since $\tilde{g} = \langle \tilde{f}, \tilde{\mu}_{gp}^{do} \rangle$ is an inner product between two finite dimensional GPs, we know the variance (as given by Proposition $C.2$) is characterised by,

$$var(g) = m_{\tilde{\mu}}^\top \Sigma_{\tilde{f}} m_{\tilde{\mu}} + m_{\tilde{f}}^\top \Sigma_{\tilde{\mu}} m_{\tilde{f}} + \mathrm{tr}(\Sigma_{\tilde{f}} \Sigma_{\tilde{\mu}}) \tag{4.71}$$

Expanding out each terms we get Proposition 5:

$$m_{\tilde{\mu}}^\top \Sigma_{\tilde{f}} m_{\tilde{\mu}} = E_x \Theta_1^\top \tilde{R}_{\hat{\mathbf{y}}\hat{\mathbf{y}}} \Theta_1 E_{x'}^\top \tag{4.72}$$

$$m_{\tilde{f}}^\top \Sigma_{\tilde{\mu}} m_{\tilde{f}} = \Theta_2^{(a)} F_{xx'} - \Theta_2^{(b)} G_{xx'} \tag{4.73}$$

$$\tag{4.74}$$

while the first two terms resembles the uncertainty obtained from IMP and BayesIME, the trace term is new and we will expand it out here,

$$\begin{aligned}
\mathrm{tr}(\Sigma_{\tilde{f}}\Sigma_{\tilde{\mu}}) &= \mathrm{tr}\left(\Phi_{\hat{\mathbf{y}}} K^{-1}_{\hat{\mathbf{y}}\hat{\mathbf{y}}} K^{\mu}_{\hat{\mathbf{y}}\hat{\mathbf{y}}} K^{-1}_{\hat{\mathbf{y}}\hat{\mathbf{y}}} \Phi^{\top}_{\hat{\mathbf{y}}} \Phi_{\hat{\mathbf{y}}} K^{-1}_{\hat{\mathbf{y}}\hat{\mathbf{y}}} \bar{R}_{\hat{\mathbf{y}}\hat{\mathbf{y}}} K^{-1}_{\hat{\mathbf{y}}\hat{\mathbf{y}}} \Phi^{\top}_{\hat{\mathbf{y}}}\right) && (4.75)\\
&= \mathrm{tr}\left(K^{-1}_{\hat{\mathbf{y}}\hat{\mathbf{y}}} K^{\mu}_{\hat{\mathbf{y}}\hat{\mathbf{y}}} K^{-1}_{\hat{\mathbf{y}}\hat{\mathbf{y}}} \bar{R}_{\hat{\mathbf{y}}\hat{\mathbf{y}}}\right) && (4.76)\\
&= \mathrm{tr}\left(K^{-1}_{\hat{\mathbf{y}}\hat{\mathbf{y}}} \left(F_{xx'} R_{\hat{\mathbf{y}}\hat{\mathbf{y}}} - G_{xx'} R_{\hat{\mathbf{y}}\mathbf{y}} R^{-1}_{\mathbf{y}\mathbf{y}} R_{\mathbf{y}\hat{\mathbf{y}}}\right) K^{-1}_{\hat{\mathbf{y}}\hat{\mathbf{y}}} \bar{R}_{\hat{\mathbf{y}}\hat{\mathbf{y}}}\right) && (4.77)\\
&= \Theta^{(a)}_3 F_{xx'} - \Theta^{(b)}_3 G_{xx'} && (4.78)
\end{aligned}$$

□

4.C Details on Experimental setup

4.C.1 Details on Ablation Study

4.C.1.1 Data Generating Process

We use the following causal graphs, $X \to Y$ and $Y \to T$, to demonstrate a simple scenario for our data fusion setting. As linking functions, we used for $\mathcal{D}_1$, $Y = xcos(\pi x) + \epsilon_1$ and for $\mathcal{D}_2$, $T = 0.5 * y * cos(y) + \epsilon_2$. where $\epsilon_i \sim \mathcal{N}(0, \sigma_i)$. Here below we plotted the data for illustration purposes.

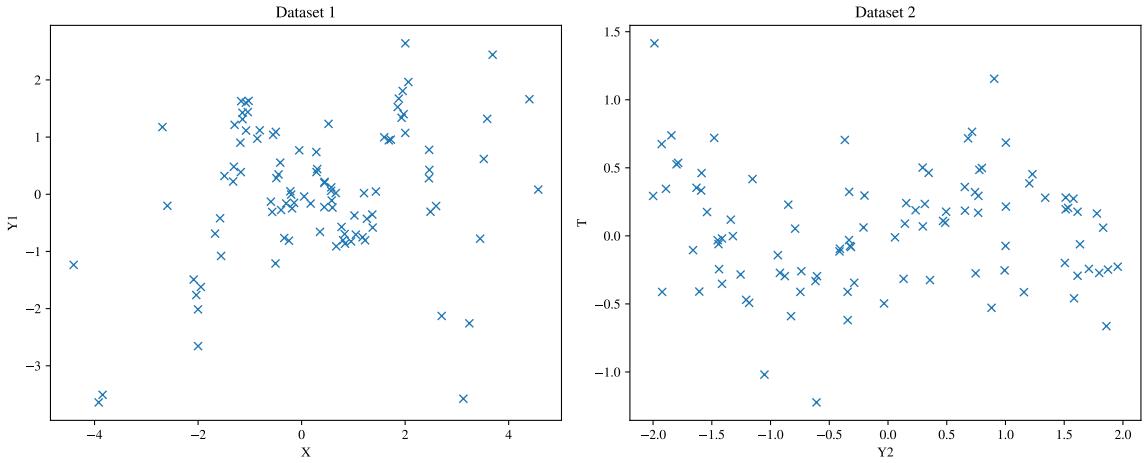

Figure 4.C.1: (Left) Illustration of $\mathcal{D}_1$ (Right) Illustration of $\mathcal{D}_2$

4.C.1.2 Explanation on the extrapolation effect

In the main text we referred to the case where IMP is better than BayesIME as **extrapolation effect**. We note from the figure above that in $\mathcal{D}_1$ we have x around -4 being mapped onto y values around -3. Note however, that in $\mathcal{D}_2$, we do not observe any values $\tilde{Y}$ below -2. Hence, because IMP uses a GP model for $\mathcal{D}_2$ we are able to account for this mismatch in support and hence attribute more uncertainty to this region, i.e. we see the spike in uncertainty in Fig.4.4.1 for IMP.

4.C.1.3 Calibration Plots

To investigate the accuracy of the uncertainty quantification in the proposed methods, we perform a (frequentist) calibration analysis of the credible intervals stemming from

each method. Fig. 4.C.2 gives the calibration plots of the Sampling methods (sampling-based method of [Aglietti et al., 2020b]) as well as the three proposed methods. On the x-axis is the portion of the posterior mass, corresponding to the width of the credible interval. We will interpret that as a nominal coverage probability of the true function values. On the y-axis is the true coverage probability estimated using the percentage of the times true function values do lie within the corresponding credible intervals. A perfectly calibrated method should have nominal coverage probability equal to the true coverage probability, i.e. being closer to the diagonal line is better.

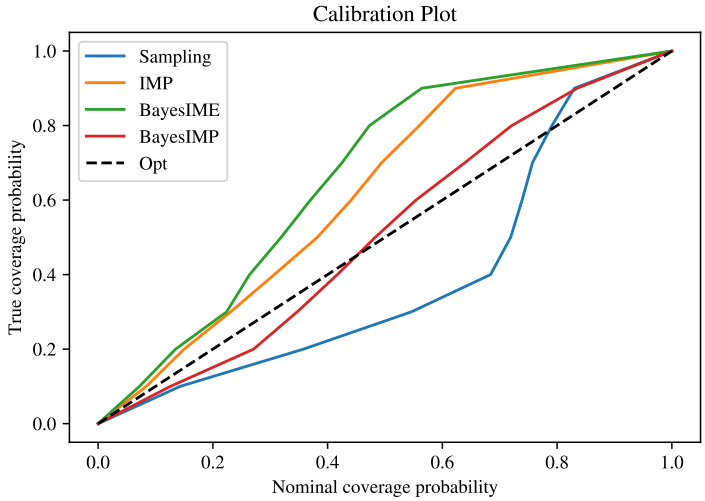

Figure 4.C.2: Calibration plots of Sampling method as well as our 3 proposed methods. We clearly see that BayesIMP is the best calibrated method amongst all other methods.

4.C.2 Details on Synthetic Data experiments

4.C.2.1 Data Generating Process for simple synthetic dataset

For the first simple synthetic dataset (See Fig.4.4.2 (Top)) we used the following data generating graph is defined as.

- $X \to U : U = 2 * X + \epsilon$
- $Z \to X : X = 3 * \cos(Z) + \epsilon$
- $\{Z, U\} \to Y : Y = U + \exp(-Z) + \epsilon$
- $Y \to T : T = \cos(Y) - \exp(-y/20) + \epsilon$

where $\epsilon \sim \mathcal{N}(0, \sigma^2)$ and $Z \sim \mathcal{U}[-4, 4]$, where for $\mathcal{D}_2$ we have that $\tilde{Y} \sim \mathcal{U}[-10, 10]$. In addition, with probability $\pi = 1/2$ we shift U by $+1$ horizontally and -3 vertically to thus create the multimodality in the data. In order to generate from the interventional distribution, we simply remove the edge from $Z \to X$ and fix the value of x.

4.C.2.2 Data Generating Process for harder synthetic dataset from [Aglietti et al., 2020b]

For the first simple synthetic dataset (Fig.4.4.2(Bottom)) we used the same data generating format as in [Aglietti et al., 2020b].

- $U_1 = \epsilon_1$
- $U_2 = \epsilon_2$
- $F = \epsilon_3$
- $A = F^2 + U_1 + \epsilon_A$
- $B = U_2 + \epsilon_B$
- $C = \exp(-B) + \epsilon_C$
- $D = \exp(-C)/10 + \epsilon_D$
- $E = \cos(A) + C/10\epsilon_E$
- $Y_1 = \cos(D) + \sin(E) + U_1 + U_2$
- $Y_2 = \cos(D) + \sin(E) + U_1 + U_2 + 2\pi$
- $T = 6 * \sin(3 * Y) + \epsilon$

where the noise is fixed to be $\mathcal{N}(0, 1)$ and where we switch with $\pi = 1/2$ from mode Y_1 and Y_2, where $\tilde{Y} \sim \mathcal{U}[-2, 9]$ for $\mathcal{D}_2$.

4.C.3 Details on Healthcare Data experiments

4.C.3.1 Data Generating Process

For the healthcare dataset, $\mathcal{D}_1$, (Fig.4.1.1) we used the same data generating format as in [Aglietti et al., 2020b] with the difference that we make *statin* continuous and increased the age range.

- $age = \mathcal{U}[15, 75]$
- $bmi = \mathcal{N}(27 - 0.01 * age, 0.7)$
- $aspirin = \sigma(-8.0 + 0.1 * age + 0.03 * bmi)$
- $statin = -13 + 0.1 * age + 0.2 * bmi$
- $cancer = \sigma(2.2 - 0.05 * age + 0.01 * bmi - 0.04 * statin + 0.02 * aspirin)$
- $PSA = \mathcal{N}(6.8 + 0.04 * age - 0.15 * bmi - 0.6 * statin + 0.55 * aspirin + cancer, 0.4)$

As for the second dataset, $\mathcal{D}_2$ we firstly fit a GP on the data collected from [Stamey et al., 1989]. Once we have the posterior GP, we can then use it as a generator for the $\mathcal{D}_2$ as it takes as input PSA. This generator hence acts as a link between $\mathcal{D}_1$ and $\mathcal{D}_2$. This way we are able to create a simulator that allows us to obtain samples from $\mathbb{E}[Cancer\ volume|do(Statin)]$ for our causal BO setup.

4.C.4 Bayesian Optimisation experiments with IMP and BayesIME

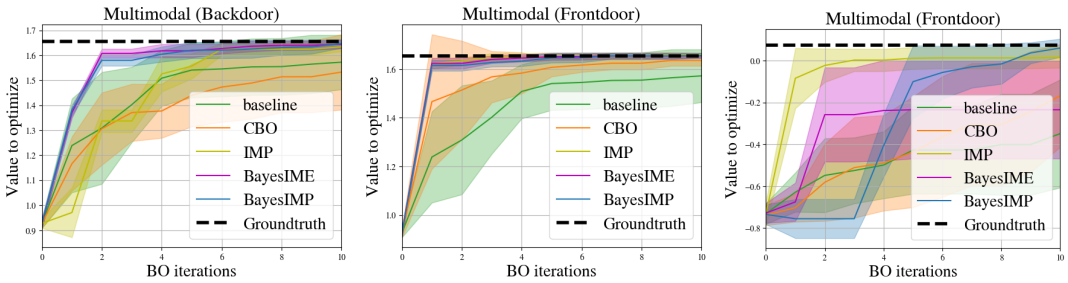

Figure 4.C.3: (Left) Simple graph using backdoor adjustment (Middle) Simple graph using front-door adjustment (Right) Harder graph using front-door adjustment. BayesIMP strikes the right balance between IMP and BayesIME and all three perform better than CBO and the GP baseline.

The main text compares BayesIMP to CBO and the baseline GP with no learnt prior in the Bayesian Optimisation experiments. Here, we include IMP and BayesIME (i.e. simplified versions of BayesIMP that account for only one source of uncertainty each) in those comparisons. We see from Fig.4.C.3 that BayesIMP is comparable to IMP and BayesIME in most cases. While BayesIMP is not the best performing method in every scenario, it does hit a good middle ground between the first two proposed methods. For Fig.4.C.3 (Left, Middle) we used $N = 100$ and $M = 50$. In the left figure, BayesIME and BayesIMP are very similar, whereas IMP is considerably worst. In the middle figure, all methods seems to perform well without much difference. In the right figure, we have $N = 500$ and $M = 50$ and this is a case where IMP is best, while BayesIME appears to get stuck in a local optimum (recall that BayesIME does not take into account uncertainty in $\mathcal{D}_2$ where there is little data). We note that all three methods converge faster than the current SOTA CBO.

Statement of Authorship for joint/multi-authored papers for PGR thesis

To appear at the end of each thesis chapter submitted as an article/paper

The statement shall describe the candidate's and co-authors' independent research contributions in the thesis publications. For each publication there should exist a complete statement that is to be filled out and signed by the candidate and supervisor **(only required where there isn't already a statement of contribution within the paper itself).**

Title of Paper	BayesIMP: Uncertainty Quantification for Causal Data Fusion
Publication Status	xPublished □ Accepted for Publication □Submitted for Publication □Unpublished and unsubmitted work written in a manuscript style
Publication Details	*Siu Lun Chau, Jean-Francois Ton, Javier González, Yee Teh, Dino Sejdinovic* Advances in Neural Information Processing Systems 34 pre-proceedings (NeurIPS 2021)

Student Confirmation

Student Name:	Jean-Francois Ton		
Contribution to the Paper	The initial idea was incepted by me and DS into rewriting causal rules into kernel mean embeddings. This however was unfortunately already proposed by Singh et al 2020 after further research. Together with SLC, we then proposed to combine SLC's current ideas of GP on mean embeddings to the causal setting [1]. I was responsible for most of the experiments while SLC was responsible for the mathematical derivations. IMP (Alg.1) is an extension to [1] to the causal setting which SLC proposed and was discussed together. BayesIME (Alg.2) was proposed by me and formalized by SLC. BayesIMP (Alg.3) was discussed together and formalized by DS and SLC. All ideas were extensively discussed between me and SLC. JG have very valuable insights into the design of the project as we were extending his work. DS and YW gave valuable insights into the work and also helped with the writing. [1] Deconditional Downscaling with Gaussian Processes Chau, Bouabid et al 2022		
Signature		Date	30/03/2022

Supervisor Confirmation

By signing the Statement of Authorship, you are certifying that the candidate made a substantial contribution to the publication, and that the description described above is accurate.

Supervisor name and title:	Professor Dino Sejdinovic		
Supervisor comments	All above seems accurate and fair.		
Signature		Date	30/03/2022

5

Conclusion, Limitations and Future Outlook

5.1 Conclusion

In this thesis, we have tackled several limitations in the current meta learning and causality literature using the help of kernel mean embeddings. In particular, we focused on meta learning for conditional density estimation (MetaCDE [Ton et al., 2021b]) and causal discovery (MetaCGNN [Ton et al., 2021c]) in chapter 2 and 3 respectively. By leveraging the power of kernel mean embeddings, especially conditional mean embeddings, we were able to capture the underlying data distribution efficiently with only few data samples. Hence this allowed us to subsequently perform both density estimation and causal discovery on new datasets and improve upon state-of-the-art methods.

Furthermore, we also delved into causal inference, where we proposed BayesIMP [Chau et al., 2021c], an algorithm that allows quantifying and propagating uncertainty in the causal data fusion setting. Here again, by noticing the connection between kernel mean embeddings and the Pearl's causal *do*-calculus as well as the connection between kernel mean embeddings and Gaussian processes, we were able to come up with an algorithm that is able to take into account the uncertainty in our causal estimates due to for example lack of data.

All of these proposed methods have at their core the use of kernel mean embeddings, which has been essential for the success of each. Being able to capture relevant

probability distributions without making parametric assumptions about them is what makes kernel mean embeddings powerful to use. We have demonstrated in several applications such as computational chemistry and health care settings that being able to account for arbitrary data distributions is paramount and that ignoring such detail can lead practitioners to draw wrong conclusions.

In the below, we will conclude with some limitations of the methods proposed in this thesis as well as a future outlook of the use of kernel mean embeddings in meta learning and causality.

5.2 Limitations

Having introduced our proposed methods in the previous chapters, we would like to also discuss the corresponding limitations to our work.

MetaCDE is able to efficiently adapt to multimodal distributions using kernel mean embeddings and work in the low data setting due to meta learning. However, given enough domain knowledge of the problem, one could fit a flexible model such as a mixture of Gaussians on the output space and hence still capture the multimodal behaviour in the computational chemistry case. Expert knowledge might also allow us to leverage Bayesian priors and hence estimate the conditional density from the posterior distribution. In the case where expert knowledge and handcrafted features are widely available, MetaCDE might not provide additional value. However, we would argue that this domain knowledge is not always available and MetaCDE provides a general framework to perform density estimation in the absence of such domain specific knowledge. In addition, as is the case with many meta learning algorithms, we also require to have access to large set of small datasets on which we can train MetaCDE. This can be restrictive in the real world, but we believe with the increase of data collection in personalized health care, recommendation systems, etc., where scarce data is captured per individual, but for many individuals, meta learning can play a crucial role in the future.

MetaCGNN also suffers from several limitations in addition to the ones mentioned above in the MetaCDE. In particular, one of the main limitation is the access to labelled data for causal discovery. Having access to this type of dataset can be very expensive in many cases, as it requires expert knowledge in order to consolidate such datasets. However, the idea of meta learning in causal discovery and causality in general, has

recently seen increased interest due to its potential applications in the wider realm of artificial intelligence, e.g. representation learning, health care etc. [Ke et al., 2020, Sharma et al., 2019, Bengio et al., 2019, Ton et al., 2021c, Dasgupta et al., 2019].

Lastly, we have BayesIMP, which mainly suffers from the fact that we require full knowledge of the causal graph in addition to the graphical assumptions that we made (see assumptions in the paper). Assuming to know the full causal graph is known to be a strong assumption, however is used in many papers involving causal inference, in particular causal Bayesian optimization [Aglietti et al., 2020b, 2021]. This is where only further progress on the second pillar of causality (causal discovery) can help. However, we remain hopeful as this area is still heavily researched with new perspectives such as [Scherrer et al., 2021], which involves an active learning approach.

5.3 Future outlook

In this thesis, we have outlined several applications of kernel mean embeddings in both meta learning and the causality setting. The particular use of kernel mean embeddings as task embedding has shown promise and in future may prove useful in settings beyond density estimation or causal discovery. One potential other application could be in the time series setting, where one could use kernel mean embeddings in conjunction with Granger Causality [Granger, 1969]. Capturing the conditional distribution of $X \to Y$ or vice versa with mean embeddings and then subsequently comparing them might allow us to view the problem of Granger Causality from a different perspective. This could, in part, follow a similar methodology to that of Mitrovic et al. [2018].

Furthermore, the idea of conditional mean embeddings have recently also seen applications in representation learning [Tsai et al., 2022]. A popular approach in machine learning to summarize datasets, even when knowing that the data comes from $Y|X$ [Garnelo et al., 2018a,b, Zaheer et al., 2017] is to simply concatenate the data elements and push them through a neural network. This hence requires the neural network to "figure out" that in fact $X \to Y$, which might require a lot of data. If we were to instead use conditional mean embeddings, one could include only conditional information into the summarization and thus construct a more data efficient methods. Many high performing algorithms which have copious amounts of data, use ideas that are similar to conditional mean embeddings in the form of Transformers [Vaswani et al., 2017]. Instead of simply concatenating the data, Transformers build a specific

architecture which allows them to distinguish between X and Y in their inputs [Kim, 2019]. However, Transformers usually require a lot of data points and hence we believe in the case where less data is available, instead of relying on "neural kernel" i.e. kernels whose feature maps are neural networks, standard conditional mean embeddings can be used as in our work on BayesIMP Chau et al. [2021c].

In terms of the implications of our work on BayesIMP for future outlook, we believe that there are several avenues that this work opened up.

Firstly, as already noted in concurrent work by [Singh et al., 2021], we are now able to turn the adjustment formulas such as front and backdoor adjustment into a kernel mean embeddings problem. This means that we do not need to estimate the respective conditional and marginal densities anymore, which are required in the standard adjustment formulas. By using our and [Singh et al., 2021] framework, we are in fact able to replace these density estimation tasks with kernel mean embeddings. What makes this framework even more interesting is that we can replace the need for integration in the standard adjustment formulas with regression tasks. By borrowing ideas from probabilistic integration and the fact that conditional mean embeddings are in fact vector-valued regressions, we eliminate the need of Monte Carlo integration as well as the reliance of good density estimation.

Secondly, quantifying the uncertainty in causal problems is not a well studied problem. In BayesIMP, we introduce a principled way to capture uncertainty not only in the data but also in the regression model itself, by employing Gaussian Processes. We combine both the advantages of kernel mean embeddings and Gaussian Processes and are hence able to model distributions without explicit assumptions on the data while also accounting for the uncertainty. We hope to shine a light on this important problem of causal inference and hope to spark many more works in this direction. In particular, an interesting future avenue would be to not only consider the uncertainty in the model and the data but also take into account the uncertainty in the causal graph. If we were unsure whether the causal graph is in fact the correct one, we would want to account for this. Constructing a pipeline with all the sources of uncertainty in mind, would be an important and impactful future direction, as most causal inference methodologies assume knowledge of the causal graph.

Finally, we hope that this thesis has demonstrated the effectiveness of kernel mean embeddings in various applications and problem setups. We hope that it can serve

as a guide on how to build bespoke models using kernel mean embeddings in future research advances on meta learning, causality and beyond.

Bibliography

M. Abadi, P. Barham, J. Chen, Z. Chen, A. Davis, J. Dean, M. Devin, S. Ghemawat, G. Irving, M. Isard, et al. {TensorFlow}: A system for {Large-Scale} machine learning. In *12th USENIX symposium on operating systems design and implementation (OSDI 16)*, pages 265–283, 2016.

V. Aglietti, T. Damoulas, M. Álvarez, and J. González. Multi-task causal learning with gaussian processes. *Advances in Neural Information Processing Systems*, 33, 2020a.

V. Aglietti, X. Lu, A. Paleyes, and J. González. Causal bayesian optimization. In *International Conference on Artificial Intelligence and Statistics*, pages 3155–3164. PMLR, 2020b.

V. Aglietti, N. Dhir, J. González, and T. Damoulas. Dynamic causal bayesian optimization. *Advances in Neural Information Processing Systems*, 34, 2021.

M. Arbel and A. Gretton. Kernel conditional exponential family. *International Conference on Artificial Intelligence and Statistics (AISTATS) 2018*, 2017.

S. Arora, H. Khandeparkar, M. Khodak, O. Plevrakis, and N. Saunshi. A theoretical analysis of contrastive unsupervised representation learning. *arXiv preprint arXiv:1902.09229*, 2019.

E. Bareinboim and J. Pearl. Causal inference and the data-fusion problem. *Proceedings of the National Academy of Sciences*, 113(27):7345–7352, 2016.

A. L. Beam and I. S. Kohane. Big data and machine learning in health care. *Jama*, 319(13):1317–1318, 2018.

Y. Bengio, T. Deleu, N. Rahaman, R. Ke, S. Lachapelle, O. Bilaniuk, A. Goyal, and C. Pal. A meta-transfer objective for learning to disentangle causal mechanisms. *arXiv preprint arXiv:1901.10912*, 2019.

A. Berlinet and C. Thomas-Agnan. *Reproducing kernel Hilbert spaces in probability and statistics*. Springer Science & Business Media, 2011.

L. Bertinetto, J. F. Henriques, J. Valmadre, P. Torr, and A. Vedaldi. Learning feed-forward one-shot learners. *Advances in neural information processing systems*, 29, 2016.

M. Bińkowski, D. J. Sutherland, M. Arbel, and A. Gretton. Demystifying mmd gans. *ICLR 2018*, 2018.

P. Blöbaum, D. Janzing, T. Washio, S. Shimizu, and B. Schölkopf. Analysis of cause-effect inference by comparing regression errors. *PeerJ Computer Science*, 5:e169, 2019.

F.-X. Briol, C. J. Oates, M. Girolami, M. A. Osborne, D. Sejdinovic, et al. Probabilistic integration: A role in statistical computation? *Statistical Science*, 34(1):1–22, 2019.

L. Buesing, T. Weber, Y. Zwols, S. Racaniere, A. Guez, J.-B. Lespiau, and N. Heess. Woulda, coulda, shoulda: Counterfactually-guided policy search. In *International Conference on Learning Representations*, 2019. URL `https://openreview.net/forum?id=BJG0voC9YQ`.

R. Carbonneau, K. Laframboise, and R. Vahidov. Application of machine learning techniques for supply chain demand forecasting. *European Journal of Operational Research*, 184(3):1140–1154, 2008.

S. L. Chau, S. Bouabid, and D. Sejdinovic. Deconditional downscaling with gaussian processes. *Neurips 2021*, 2021a.

S. L. Chau, J. Gonzalez, and D. Sejdinovic. Rkhs-shap: Shapley values for kernel methods. *arXiv preprint arXiv:2110.09167*, 2021b.

S. L. Chau, J.-F. Ton, J. González, Y. Teh, and D. Sejdinovic. Bayesimp: Uncertainty quantification for causal data fusion. *Advances in Neural Information Processing Systems*, 34, 2021c.

W.-Y. Chen, Y.-C. Liu, Z. Kira, Y.-C. F. Wang, and J.-B. Huang. A closer look at few-shot classification. *arXiv preprint arXiv:1904.04232*, 2019.

X. Chen, Z. Wang, S. Tang, and K. Muandet. Mate: plugging in model awareness to task embedding for meta learning. *Advances in Neural Information Processing Systems*, 33:11865–11877, 2020.

X.-W. Chen and X. Lin. Big data deep learning: challenges and perspectives. *IEEE access*, 2:514–525, 2014.

Y. Chen, M. Welling, and A. Smola. Super-samples from kernel herding. *arXiv preprint arXiv:1203.3472*, 2012.

Z. Chen, K. Zhang, L. Chan, and B. Schölkopf. Causal discovery via reproducing kernel hilbert space embeddings. *Neural computation*, 26(7):1484–1517, 2014.

D. M. Chickering. Optimal structure identification with greedy search. *Journal of machine learning research*, 3(Nov):507–554, 2002.

T. A. Courtney, M. Lebrato, N. R. Bates, A. Collins, S. J. De Putron, R. Garley, R. Johnson, J.-C. Molinero, T. J. Noyes, C. L. Sabine, et al. Environmental controls on modern scleractinian coral and reef-scale calcification. *Science advances*, 3(11): e1701356, 2017.

B. Dai, H. Dai, A. Gretton, L. Song, D. Schuurmans, and N. He. Kernel exponential family estimation via doubly dual embedding. *arXiv preprint arXiv:1811.02228*, 2018.

P. Daniusis, D. Janzing, J. Mooij, J. Zscheischler, B. Steudel, K. Zhang, and B. Schölkopf. Inferring deterministic causal relations. *arXiv preprint arXiv:1203.3475*, 2012.

I. Dasgupta, J. Wang, S. Chiappa, J. Mitrovic, P. Ortega, D. Raposo, E. Hughes, P. Battaglia, M. Botvinick, and Z. Kurth-Nelson. Causal reasoning from meta-reinforcement learning. *arXiv preprint arXiv:1901.08162*, 2019.

M. F. Dixon, I. Halperin, and P. Bilokon. *Machine learning in Finance*, volume 1170. Springer, 2020.

A. Ferro, F. Pina, M. Severo, P. Dias, F. Botelho, and N. Lunet. Use of statins and serum levels of prostate specific antigen. *Acta Urológica Portuguesa*, 32(2):71–77, 2015.

C. Finn. *Stanford CS330: Multi-Task and Meta-Learning, 2019 | Lecture 1 - Introduction Overview*, 2019. URL `https://www.youtube.com/watch?v=0rZtSwNOTQo`.

C. Finn, P. Abbeel, and S. Levine. Model-agnostic meta-learning for fast adaptation of deep networks. In *International conference on machine learning*, pages 1126–1135. PMLR, 2017.

C. Finn, K. Xu, and S. Levine. Probabilistic model-agnostic meta-learning. *Advances in neural information processing systems*, 31, 2018.

S. Flaxman, D. Sejdinovic, J. P. Cunningham, and S. Filippi. Bayesian learning of kernel embeddings. *arXiv preprint arXiv:1603.02160*, 2016.

J. A. Fonollosa. Conditional distribution variability measures for causality detection. In *Cause Effect Pairs in Machine Learning*, pages 339–347. Springer, 2019.

K. Fukumizu, F. R. Bach, and M. I. Jordan. Dimensionality reduction for supervised learning with reproducing kernel hilbert spaces. *Journal of Machine Learning Research*, 5(Jan):73–99, 2004.

K. Fukumizu, L. Song, and A. Gretton. Kernel bayes' rule. *arXiv preprint arXiv:1009.5736*, 2010.

K. Fukumizu, L. Song, and A. Gretton. Kernel bayes' rule: Bayesian inference with positive definite kernels. *The Journal of Machine Learning Research*, 14(1):3753–3783, 2013.

M. Garnelo, D. Rosenbaum, C. J. Maddison, T. Ramalho, D. Saxton, M. Shanahan, Y. W. Teh, D. J. Rezende, and S. Eslami. Conditional neural processes. In *Proceedings of the 35th International Conference on Machine Learning*, volume 80, pages 1704–1713. PMLR, 2018a. URL `http://proceedings.mlr.press/v80/garnelo18a.html`.

M. Garnelo, J. Schwarz, D. Rosenbaum, F. Viola, D. J. Rezende, S. Eslami, and Y. W. Teh. Neural processes. *International Conference on Machine Learning*, 2018b.

H. Ghoddusi, G. G. Creamer, and N. Rafizadeh. Machine learning in energy economics and finance: A review. *Energy Economics*, 81:709–727, 2019.

S. Gidaris and N. Komodakis. Dynamic few-shot visual learning without forgetting. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 4367–4375, 2018.

S. Gidaris and N. Komodakis. Generating classification weights with gnn denoising autoencoders for few-shot learning. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 21–30, 2019.

C. Glymour, K. Zhang, and P. Spirtes. Review of causal discovery methods based on graphical models. *Frontiers in genetics*, 10:524, 2019.

I. Goodfellow. Tweet by ian goodfellow. `https://twitter.com/goodfellow_ian/status/1127723454906503168`, 2019. Accessed: 2019-10-02.

I. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio. Generative adversarial nets. *Advances in neural information processing systems*, 27, 2014.

J. Gordon, W. P. Bruinsma, A. Y. Foong, J. Requeima, Y. Dubois, and R. E. Turner. Convolutional conditional neural processes. *ICLR 2020*, 2020.

O. Goudet, D. Kalainathan, P. Caillou, I. Guyon, D. Lopez-Paz, and M. Sebag. Causal generative neural networks. `https://openreview.net/pdf?id=ry5wc1bCW`.

O. Goudet, D. Kalainathan, P. Caillou, I. Guyon, D. Lopez-Paz, and M. Sebag. Causal generative neural networks. *arXiv preprint arXiv:1711.08936*, 2017.

O. Goudet, D. Kalainathan, P. Caillou, I. Guyon, D. Lopez-Paz, and M. Sebag. Learning functional causal models with generative neural networks. In *Explainable and Interpretable Models in Computer Vision and Machine Learning*, pages 39–80. Springer, 2018.

C. W. Granger. Investigating causal relations by econometric models and cross-spectral methods. *Econometrica: journal of the Econometric Society*, pages 424–438, 1969.

S. Gražulis, A. Daškevič, A. Merkys, D. Chateigner, L. Lutterotti, M. Quiros, N. R. Serebryanaya, P. Moeck, R. T. Downs, and A. Le Bail. Crystallography open database (cod): an open-access collection of crystal structures and platform for world-wide collaboration. *Nucleic acids research*, 40(D1):D420–D427, 2011.

A. Gretton. Introduction to rkhs, and some simple kernel algorithms, October 2019. URL `http://www.gatsby.ucl.ac.uk/~gretton/coursefiles/rkhscourse.html`.

A. Gretton, K. M. Borgwardt, M. J. Rasch, B. Schölkopf, and A. Smola. A kernel two-sample test. *The Journal of Machine Learning Research*, 13(1):723–773, 2012.

S. Grünewälder, G. Lever, L. Baldassarre, S. Patterson, A. Gretton, and M. Pontil. Conditional mean embeddings as regressors-supplementary. *arXiv preprint arXiv:1205.4656*, 2012.

P. D. Grünwald et al. Algorithmic information theory. 2008.

M. Gutmann and A. Hyvärinen. Noise-contrastive estimation: A new estimation principle for unnormalized statistical models. In *Proceedings of the Thirteenth International Conference on Artificial Intelligence and Statistics*, pages 297–304, 2010.

M. U. Gutmann and A. Hyvärinen. Noise-contrastive estimation of unnormalized statistical models, with applications to natural image statistics. *Journal of Machine Learning Research*, 13(Feb):307–361, 2012.

Z. N. Harris, E. Dhungel, M. Mosior, and T.-H. Ahn. Massive metagenomic data analysis using abundance-based machine learning. *Biology direct*, 14(1):1–13, 2019.

P. C. D. Hawkins. Conformation Generation: The State of the Art. *Journal of Chemical Information and Modeling*, 57(8):1747–1756, 2017. doi: 10.1021/acs.jcim.7b00221. URL `https://doi.org/10.1021/acs.jcim.7b00221`. PMID: 28682617.

J. B. Heaton, N. G. Polson, and J. H. Witte. Deep learning for finance: deep portfolios. *Applied Stochastic Models in Business and Industry*, 33(1):3–12, 2017.

S. Hochreiter and J. Schmidhuber. Long short-term memory. *Neural computation*, 9 (8):1735–1780, 1997.

P. Hoyer, D. Janzing, J. M. Mooij, J. Peters, and B. Schölkopf. Nonlinear causal discovery with additive noise models. *Advances in neural information processing systems*, 21, 2008.

P. O. Hoyer, D. Janzing, J. M. Mooij, J. Peters, and B. Schölkopf. Nonlinear causal discovery with additive noise models. In *Advances in neural information processing systems*, pages 689–696, 2009.

K. Hsu, R. Nock, and F. Ramos. Hyperparameter learning for conditional kernel mean embeddings with rademacher complexity bounds. In *Joint European Conference on Machine Learning and Knowledge Discovery in Databases*, pages 227–242. Springer, 2018.

Y. Huang and M. Valtorta. Pearl's calculus of intervention is complete. *arXiv preprint arXiv:1206.6831*, 2012.

E. Hüllermeier and W. Waegeman. Aleatoric and epistemic uncertainty in machine learning: An introduction to concepts and methods. *Machine Learning*, 110(3): 457–506, 2021.

M. A. Jamal and G.-J. Qi. Task agnostic meta-learning for few-shot learning. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 11719–11727, 2019.

D. Janzing and B. Scholkopf. Causal inference using the algorithmic markov condition. *IEEE Transactions on Information Theory*, 56(10):5168–5194, 2010.

J. M. Johnson and T. M. Khoshgoftaar. Survey on deep learning with class imbalance. *Journal of Big Data*, 6(1):1–54, 2019.

D. Kalainathan and O. Goudet. Causal discovery toolbox: Uncover causal relationships in python. *Journal of Machine Learning Research*, 21(37):1–5, 2020. URL `http://jmlr.org/papers/v21/19-187.html`.

M. Kanagawa. Empirical representations of probability distributions via kernel mean embeddings. 2016.

N. R. Ke, O. Bilaniuk, A. Goyal, S. Bauer, H. Larochelle, B. Schölkopf, M. C. Mozer, C. Pal, and Y. Bengio. Learning neural causal models from unknown interventions. *arXiv preprint arXiv:1910.01075*, 2019.

N. R. Ke, J. Wang, J. Mitrovic, M. Szummer, D. J. Rezende, et al. Amortized learning of neural causal representations. *arXiv preprint arXiv:2008.09301*, 2020.

H. Kim, A. Mnih, J. Schwarz, M. Garnelo, A. Eslami, D. Rosenbaum, O. Vinyals, and Y. W. Teh. Attentive neural processes. *ICLR 2019*, 2019.

K. Kim. Attention: the analogue of kernels in deep learning, September 2019. URL `https://hyunjik11.github.io/talks/Attention_the_Analogue_of_Kernels_in_Deep_Learning.pdf`.

K.-E. Kim and H. S. Park. Imitation learning via kernel mean embedding. In *Thirty-Second AAAI Conference on Artificial Intelligence*, 2018.

D. P. Kingma and J. Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

S. Lacoste-Julien, F. Lindsten, and F. Bach. Sequential kernel herding: Frank-wolfe optimization for particle filtering. In *Artificial Intelligence and Statistics*, pages 544–552. PMLR, 2015.

H. C. L. Law, D. Sutherland, D. Sejdinovic, and S. Flaxman. Bayesian approaches to distribution regression. In *International Conference on Artificial Intelligence and Statistics*, pages 1167–1176. PMLR, 2018.

Y. LeCun, Y. Bengio, et al. Convolutional networks for images, speech, and time series. *The handbook of brain theory and neural networks*, 3361(10):1995, 1995.

Y. LeCun, Y. Bengio, and G. Hinton. Deep learning. *nature*, 521(7553):436–444, 2015.

J. Lemeire and E. Dirkx. Causal models as minimal descriptions of multivariate systems, 2006. URL `http://parallel.vub.ac.be/~jan/papers/JanLemeire_CausalModelsAsMinimalDescriptionsOfMultivariateSystems_October2006.pdf`.

G. Lever, J. Shawe-Taylor, R. Stafford, and C. Szepesvári. Compressed conditional mean embeddings for model-based reinforcement learning. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 30, 2016.

C.-L. Li, W.-C. Chang, Y. Cheng, Y. Yang, and B. Póczos. MMMD GAN: Towards deeper understanding of moment matching network. In *Advances in Neural Information Processing Systems*, pages 2203–2213, 2017.

H. Li, W. Dong, X. Mei, C. Ma, F. Huang, and B.-G. Hu. Lgm-net: Learning to generate matching networks for few-shot learning. In *International conference on machine learning*, pages 3825–3834. PMLR, 2019a.

X. Li, Z. Sun, J.-H. Xue, and Z. Ma. A concise review of recent few-shot meta-learning methods. *Neurocomputing*, 456:463–468, 2021.

Z. Li, J.-F. Ton, D. Oglic, and D. Sejdinovic. Towards A Unified Analysis of Random Fourier Features. In *International Conference on Machine Learning (ICML)*, pages PMLR 97:3905–3914, 2019b. URL `http://proceedings.mlr.press/v97/li19k.html`.

F. Liu, W. Xu, J. Lu, G. Zhang, A. Gretton, and D. J. Sutherland. Learning deep kernels for non-parametric two-sample tests. *arXiv preprint arXiv:2002.09116*, 2020.

A. Look and S. Riedelbauch. Learning with little data: Evaluation of deep learning algorithms. 2018.

D. Lopez-Paz, K. Muandet, and B. Recht. The randomized causation coefficient. *The Journal of Machine Learning Research*, 16(1):2901–2907, 2015a.

D. Lopez-Paz, K. Muandet, B. Schölkopf, and I. Tolstikhin. Towards a learning theory of cause-effect inference. In *International Conference on Machine Learning*, pages 1452–1461. PMLR, 2015b.

D. Lopez-Paz, R. Nishihara, S. Chintala, B. Scholkopf, and L. Bottou. Discovering causal signals in images. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 6979–6987, 2017.

M. Lukić and J. Beder. Stochastic processes with sample paths in reproducing kernel hilbert spaces. *Transactions of the American Mathematical Society*, 353(10):3945–3969, 2001.

C. Lyle, P. S. Castro, and M. G. Bellemare. A comparative analysis of expected and distributional reinforcement learning. *arXiv preprint arXiv:1901.11084*, 2019.

Z. Ma and M. Collins. Noise contrastive estimation and negative sampling for conditional models: Consistency and statistical efficiency. *arXiv preprint arXiv:1809.01812*, 2018.

K. V. Mardia. Statistical approaches to three key challenges in protein structural bioinformatics. *Journal of the Royal Statistical Society: Series C (Applied Statistics)*, 62(3):487–514, 2013.

A. Marx and J. Vreeken. Telling cause from effect using mdl-based local and global regression. In *2017 IEEE international conference on data mining (ICDM)*, pages 307–316. IEEE, 2017.

T. Meng, X. Jing, Z. Yan, and W. Pedrycz. A survey on machine learning for data fusion. *Information Fusion*, 57:115–129, 2020.

O. Mian, A. Marx, and J. Vreeken. Discovering fully oriented causal networks. In *Proceedings of the AAAI Conference on Artificial Intelligence (AAAI)*, 2021.

M. Minsky. Steps toward artificial intelligence. *Proceedings of the IRE*, 49(1):8–30, 1961.

R. Miotto, F. Wang, S. Wang, X. Jiang, and J. T. Dudley. Deep learning for healthcare: review, opportunities and challenges. *Briefings in bioinformatics*, 19(6):1236–1246, 2018.

N. Mishra, M. Rohaninejad, X. Chen, and P. Abbeel. A simple neural attentive meta-learner. *arXiv preprint arXiv:1707.03141*, 2017.

J. Mitrovic, D. Sejdinovic, and Y. W. Teh. Causal inference via kernel deviance measures. *Advances in neural information processing systems*, 31, 2018.

A. Mnih and Y. W. Teh. A fast and simple algorithm for training neural probabilistic language models. In *Proceedings of the 29th International Conference on Machine Learning*, pages 1751–1758, 2012.

R. P. Monti, K. Zhang, and A. Hyvarinen. Causal discovery with general non-linear relationships using non-linear ICA. In *35th Conference on Uncertainty in Artificial Intelligence (UAI 2019)*, 2019.

J. M. Mooij, J. Peters, D. Janzing, J. Zscheischler, and B. Schölkopf. Distinguishing cause from effect using observational data: methods and benchmarks. *The Journal of Machine Learning Research*, 17(1):1103–1204, 2016.

K. Muandet, K. Fukumizu, B. Sriperumbudur, B. Schölkopf, et al. Kernel mean embedding of distributions: A review and beyond. *Foundations and Trends® in Machine Learning*, 10(1-2):1–141, 2017.

K. Muandet, M. Kanagawa, S. Saengkyongam, and S. Marukatat. Counterfactual mean embeddings. *Journal of Machine Learning Research 22*, 2021.

T. Munkhdalai and H. Yu. Meta networks. In *International Conference on Machine Learning*, pages 2554–2563. PMLR, 2017.

T. Munkhdalai, X. Yuan, S. Mehri, and A. Trischler. Rapid adaptation with conditionally shifted neurons. In *International Conference on Machine Learning*, pages 3664–3673. PMLR, 2018.

M. M. Najafabadi, F. Villanustre, T. M. Khoshgoftaar, N. Seliya, R. Wald, and E. Muharemagic. Deep learning applications and challenges in big data analytics. *Journal of big data*, 2(1):1–21, 2015.

L. G. Neuberg. Causality: models, reasoning, and inference, by judea pearl, cambridge university press, 2000. *Econometric Theory*, 19(4):675–685, 2003.

A. Paszke, S. Gross, S. Chintala, G. Chanan, E. Yang, Z. DeVito, Z. Lin, A. Desmaison, L. Antiga, and A. Lerer. Automatic differentiation in pytorch. 2017.

J. Pearl. Causal diagrams for empirical research. *Biometrika*, 82(4):669–688, 1995.

J. Pearl. *Causality*. Cambridge university press, 2009.

J. Pearl. The do-calculus revisited. *arXiv preprint arXiv:1210.4852*, 2012.

J. Pearl et al. Models, reasoning and inference. *Cambridge, UK: CambridgeUniversity-Press*, 19:2, 2000.

E. Perez, F. Strub, H. De Vries, V. Dumoulin, and A. Courville. Film: Visual reasoning with a general conditioning layer. In *Thirty-Second AAAI Conference on Artificial Intelligence*, 2018.

J. Peters, D. Janzing, and B. Schölkopf. *Elements of causal inference: foundations and learning algorithms*. The MIT Press, 2017.

H. Qi, M. Brown, and D. G. Lowe. Low-shot learning with imprinted weights. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 5822–5830, 2018.

A. Rahimi and B. Recht. Random features for large-scale kernel machines. In *Advances in neural information processing systems*, pages 1177–1184, 2008.

C. Rasmussen and C. Williams. Gaussian Processes for Machine Learning, 2005a.

C. E. Rasmussen and C. K. I. Williams. *Gaussian processes for machine learning*. MIT Press, 2005b.

J. Requeima, J. Gordon, J. Bronskill, S. Nowozin, and R. E. Turner. Fast and flexible multi-task classification using conditional neural adaptive processes. In *Advances in Neural Information Processing Systems*, pages 7957–7968, 2019.

D. B. Rubin. Causal inference using potential outcomes: Design, modeling, decisions. *Journal of the American Statistical Association*, 100(469):322–331, 2005.

F. Rundo, F. Trenta, A. L. di Stallo, and S. Battiato. Machine learning for quantitative finance applications: A survey. *Applied Sciences*, 9(24):5574, 2019.

A. Santoro, S. Bartunov, M. Botvinick, D. Wierstra, and T. Lillicrap. Meta-learning with memory-augmented neural networks. In *International conference on machine learning*, pages 1842–1850. PMLR, 2016.

N. Scherrer, O. Bilaniuk, Y. Annadani, A. Goyal, P. Schwab, B. Schölkopf, M. C. Mozer, Y. Bengio, S. Bauer, and N. R. Ke. Learning neural causal models with active interventions. *arXiv preprint arXiv:2109.02429*, 2021.

B. Schölkopf, K. Muandet, K. Fukumizu, S. Harmeling, and J. Peters. Computing functions of random variables via reproducing kernel hilbert space representations. *Statistics and Computing*, 25(4):755–766, 2015.

J. Schrittwieser, I. Antonoglou, T. Hubert, K. Simonyan, L. Sifre, S. Schmitt, A. Guez, E. Lockhart, D. Hassabis, T. Graepel, et al. Mastering atari, go, chess and shogi by planning with a learned model. *Nature*, 588(7839):604–609, 2020.

I. Schuster, M. Mollenhauer, S. Klus, and K. Muandet. Kernel conditional density operators. In *International Conference on Artificial Intelligence and Statistics*, pages 993–1004. PMLR, 2020.

D. Sejdinovic. Kernel methods, 2019. URL `https://www.stats.ox.ac.uk/~sejdinov/teaching/atsml19/`.

D. Sejdinovic, H. Strathmann, M. L. Garcia, C. Andrieu, and A. Gretton. Kernel adaptive metropolis-hastings. In *International conference on machine learning*, pages 1665–1673. PMLR, 2014.

E. Sgouritsa, D. Janzing, J. Peters, and B. Schölkopf. Identifying finite mixtures of nonparametric product distributions and causal inference of confounders. *arXiv preprint arXiv:1309.6860*, 2013.

K. Shailaja, B. Seetharamulu, and M. Jabbar. Machine learning in healthcare: A review. In *2018 Second international conference on electronics, communication and aerospace technology (ICECA)*, pages 910–914. IEEE, 2018.

A. Sharma, G. Gupta, R. Prasad, A. Chatterjee, L. Vig, and G. Shroff. Metaci: Meta-learning for causal inference in a heterogeneous population. *arXiv preprint arXiv:1912.03960*, 2019.

J. Shawe-Taylor, N. Cristianini, et al. *Kernel methods for pattern analysis*. Cambridge university press, 2004.

S. Shimizu, P. O. Hoyer, A. Hyvärinen, A. Kerminen, and M. Jordan. A linear non-gaussian acyclic model for causal discovery. *Journal of Machine Learning Research*, 7(10), 2006.

I. Shpitser and J. Pearl. *Identification of joint interventional distributions in recursive semi-Markovian causal models*. eScholarship, University of California, 2006.

D. Silver, T. Hubert, J. Schrittwieser, I. Antonoglou, M. Lai, A. Guez, M. Lanctot, L. Sifre, D. Kumaran, T. Graepel, et al. Mastering chess and shogi by self-play with a general reinforcement learning algorithm. *arXiv preprint arXiv:1712.01815*, 2017.

D. Silver, T. Hubert, J. Schrittwieser, I. Antonoglou, M. Lai, A. Guez, M. Lanctot, L. Sifre, D. Kumaran, T. Graepel, et al. A general reinforcement learning algorithm that masters chess, shogi, and go through self-play. *Science*, 362(6419):1140–1144, 2018.

R. Singh, M. Sahani, and A. Gretton. Kernel instrumental variable regression. In H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox, and R. Garnett, editors, *Advances in Neural Information Processing Systems*, volume 32. Curran Associates, Inc., 2019a.

R. Singh, M. Sahani, and A. Gretton. Kernel instrumental variable regression. *33rd Conference on Neural Information Processing Systems (NeurIPS 2019)*, 2019b.

R. Singh, L. Xu, and A. Gretton. Kernel methods for policy evaluation: Treatment effects, mediation analysis, and off-policy planning. *arXiv preprint arXiv:2010.04855*, 2020.

R. Singh, L. Xu, and A. Gretton. Kernel methods for multistage causal inference: Mediation analysis and dynamic treatment effects. *arXiv preprint arXiv:2111.03950*, 2021.

A. Smola, A. Gretton, L. Song, and B. Schölkopf. A hilbert space embedding for distributions. In *International Conference on Algorithmic Learning Theory*, pages 13–31. Springer, 2007.

L. Song, J. Huang, A. Smola, and K. Fukumizu. Hilbert space embeddings of conditional distributions with applications to dynamical systems. In *Proceedings of the 26th Annual International Conference on Machine Learning*, pages 961–968, 2009.

L. Song, K. Fukumizu, and A. Gretton. Kernel embeddings of conditional distributions: A unified kernel framework for nonparametric inference in graphical models. *IEEE Signal Processing Magazine*, 30(4):98–111, 2013.

P. Spirtes and K. Zhang. Causal discovery and inference: concepts and recent methodological advances. In *Applied informatics*, volume 3, pages 1–28. SpringerOpen, 2016.

P. Spirtes, C. N. Glymour, R. Scheines, and D. Heckerman. *Causation, prediction, and search*. MIT press, 2000.

B. K. Sriperumbudur, K. Fukumizu, and G. R. Lanckriet. Universality, characteristic kernels and RKHS embedding of measures. *Journal of Machine Learning Research*, 12(70):2389–2410, 2011a. URL `http://jmlr.org/papers/v12/sriperumbudur11a.html`.

B. K. Sriperumbudur, K. Fukumizu, and G. R. G. Lanckriet. Universality, characteristic kernels and rkhs embedding of measures. *J. Mach. Learn. Res.*, 12:2389–2410, July 2011b. ISSN 1532-4435.

T. A. Stamey, J. N. Kabalin, J. E. McNeal, I. M. Johnstone, F. Freiha, E. A. Redwine, and N. Yang. Prostate specific antigen in the diagnosis and treatment of adenocarcinoma of the prostate. ii. radical prostatectomy treated patients. *The Journal of urology*, 141(5):1076–1083, 1989.

O. Stegle, D. Janzing, K. Zhang, J. M. Mooij, and B. Schölkopf. Probabilistic latent variable models for distinguishing between cause and effect. In *Advances in neural information processing systems*, pages 1687–1695, 2010.

M. Sugiyama, I. Takeuchi, T. Suzuki, T. Kanamori, H. Hachiya, and D. Okanohara. Least-squares conditional density estimation. *IEICE Transactions on Information and Systems*, 93(3):583–594, 2010.

X. Sun, D. Janzing, and B. Schölkopf. Distinguishing between cause and effect via kernel-based complexity measures for conditional distributions. In *15th European Symposium on Artificial Neural Networks (ESANN 2007)*, pages 441–446. D-Side Publications, 2007a.

X. Sun, D. Janzing, B. Schölkopf, and K. Fukumizu. A kernel-based causal learning algorithm. In *Proceedings of the 24th International Conference on Machine Learning*, ICML '07, page 855–862, New York, NY, USA, 2007b. Association for Computing Machinery. ISBN 9781595937933. doi: 10.1145/1273496.1273604. URL `https://doi.org/10.1145/1273496.1273604`.

Z. Szabó, B. K. Sriperumbudur, B. Póczos, and A. Gretton. Learning theory for distribution regression. *The Journal of Machine Learning Research*, 17(1):5272–5311, 2016.

C. Thompson. Causal graph analysis with the causalgraph procedure. In *Proceedings of SAS Global Forum*, 2019.

P. Tokmakov, Y.-X. Wang, and M. Hebert. Learning compositional representations for few-shot recognition. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 6372–6381, 2019.

J.-F. Ton, L. Chan, Y. Whye Teh, and D. Sejdinovic. Noise contrastive meta-learning for conditional density estimation using kernel mean embeddings. In A. Banerjee and K. Fukumizu, editors, *Proceedings of The 24th International Conference on Artificial Intelligence and Statistics*, volume 130 of *Proceedings of Machine Learning Research*, pages 1099–1107. PMLR, 13–15 Apr 2021a. URL `http://proceedings.mlr.press/v130/ton21a.html`.

J.-F. Ton, C. Lucian, Y. W. Teh, and D. Sejdinovic. Noise contrastive meta-learning for conditional density estimation using kernel mean embeddings. In *International Conference on Artificial Intelligence and Statistics*, pages 1099–1107. PMLR, 2021b.

J.-F. Ton, D. Sejdinovic, and K. Fukumizu. Meta learning for causal direction. *Thirty-Fifth AAAI Conference on Artificial Intelligence (AAAI-21)*, 2021c.

G. F. Trecate, C. K. Williams, and M. Opper. Finite-dimensional approximation of gaussian processes. In *Proceedings of the 1998 conference on Advances in neural information processing systems II*, pages 218–224, 1999.

B. L. Trippe and R. E. Turner. Conditional density estimation with bayesian normalising flows. *arXiv preprint arXiv:1802.04908*, 2018.

Y.-H. H. Tsai, T. Li, M. Q. Ma, H. Zhao, K. Zhang, L.-P. Morency, and R. Salakhutdinov. Conditional contrastive learning with kernel. 2022.

I. Tsamardinos, L. E. Brown, and C. F. Aliferis. The max-min hill-climbing bayesian network structure learning algorithm. *Machine learning*, 65(1):31–78, 2006.

A. Van den Oord, Y. Li, and O. Vinyals. Representation learning with contrastive predictive coding. *arXiv preprint arXiv:1807.03748*, 2018.

J. Vanschoren. Meta-learning: A survey. *arXiv preprint arXiv:1810.03548*, 2018.

A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, Ł. Kaiser, and I. Polosukhin. Attention is all you need. *Advances in neural information processing systems*, 30, 2017.

R. Vilalta and Y. Drissi. A perspective view and survey of meta-learning. *Artificial intelligence review*, 18(2):77–95, 2002.

O. Vinyals, C. Blundell, T. Lillicrap, k. kavukcuoglu, and D. Wierstra. Matching networks for one shot learning. In D. D. Lee, M. Sugiyama, U. V. Luxburg, I. Guyon, and R. Garnett, editors, *Advances in Neural Information Processing Systems 29*, pages 3630–3638. Curran Associates, Inc., 2016. URL `http://papers.nips.cc/paper/6385-matching-networks-for-one-shot-learning.pdf`.

E. Wagstaff, F. B. Fuchs, M. Engelcke, M. A. Osborne, and I. Posner. Universal approximation of functions on sets. *Journal of Machine Learning Research*, 23(151): 1–56, 2022.

L. Weng. Meta-learning: Learning to learn fast, November 2018. URL `https://lilianweng.github.io/posts/2018-11-30-meta-learning/`.

L. Wenliang, D. Sutherland, H. Strathmann, and A. Gretton. Learning deep kernels for exponential family densities. *arXiv preprint arXiv:1811.08357*, 2018.

A. G. Wilson, Z. Hu, R. Salakhutdinov, and E. P. Xing. Deep kernel learning. In *Artificial Intelligence and Statistics*, pages 370–378, 2016.

P. Wu and K. Fukumizu. Causal Mosaic: Cause-effect inference via nonlinear ICA and ensemble method. In *Proceedings of Artificial Intelligence and Statistics*, 2020.

J. Xu, J.-F. Ton, H. Kim, A. R. Kosiorek, and Y. W. Teh. Metafun: Meta-learning with iterative functional updates. *ICML 2019*, 2019.

M. Zaheer, S. Kottur, S. Ravanbakhsh, B. Poczos, R. R. Salakhutdinov, and A. J. Smola. Deep sets. *Advances in neural information processing systems*, 30, 2017.

K. Zhang and A. Hyvarinen. On the identifiability of the post-nonlinear causal model. *UAI 2009*, 2012.

K. Zhang, J. Peters, D. Janzing, and B. Schölkopf. Kernel-based conditional independence test and application in causal discovery. In *Proceedings of the Twenty-Seventh Conference on Uncertainty in Artificial Intelligence*, UAI'11, page 804–813, 2011.

Q. Zhang, S. Filippi, A. Gretton, and D. Sejdinovic. Large-scale kernel methods for independence testing. *Statistics and Computing*, 28(1):113–130, 2018a.

R. Zhang, T. Che, Z. Ghahramani, Y. Bengio, and Y. Song. Metagan: An adversarial approach to few-shot learning. In *Advances in Neural Information Processing Systems*, pages 2365–2374, 2018b.

S. Zhu and Z. Chen. Causal discovery with reinforcement learning. In *International Conference on Learning Representations*, 2020. URL `https://openreview.net/forum?id=S1g2skStPB`.