Large Scale Methods for Kernels, Causal Inference and Survival Modelling



Robert Hu

Mansfield College

University of Oxford

A thesis submitted for the degree of

Doctor of Philosophy

Michaelmas 2022

Statement of Originality

I hereby declare that except where specific reference is made to the work of others, the intellectual contents of this dissertation are original and have not been submitted in whole or in part for consideration for any other degree or qualification. My personal contributions are detailed in the authorship forms at the end of each chapter. This dissertation is my own work except as specified in the text and authorship forms.

Robert Hu
Michaelmas 2022

To my old man, mommy and grandparents

Acknowledgements

I would first and foremost like to thank Professor Dino Sejdinovic for having the heart and compassion to reply to a nervously composed email pleading for studentship during the summer of 2018. I would also like to thank Professor Robin J. Evans and Professor Geoff K. Nicholls for seeing my few redeeming qualities in the subsequent interviews. Now that the acknowledgements for the thesis are finally being written, I would like to express further gratitude to Dino, Robin and Geoff for bearing with me for 4 years.

Hugs and kisses to my brothers in arms during our time in the stats department: Adam, Andrew, Anthony, Bobby, Bohao, Carlo, Desi, Edo, Edwin, Emi, Emile, Faaiz, Fran, Kaspar, Hanwen, Jin, Leon, Rob, Sheh, Shahine, Tyler and Zhongyi. You made the best better and the worst suck a lot less.

I would further like to express special gratitude towards Professor Joan Alexis Glaunès, Professor David Steinsaltz, Dr Maria Christodoulou and Dr Zhu Li for further guiding and helping me grow as a scientist and a machine learning practitioner. Also, Alan, David, Jake, Jef and Veit, thank you so much for the happy times and work together, I wouldn't have made it to this paragraph if it weren't for you bros!

I would also like to thank Professor Ricardo Silva, Professor Tom Rainforth, Professor Xiaowen Dong and Professor Yee Whye Teh for being the transfer, confirmation and viva examiners, and putting up with all the ill-formatted beamer presentations.

A special thanks to Stuart McRobert at the IT department for being an absolute unit by keeping the ziz servers alive despite my constant abuse of the GPUs.

I am grateful to Mansfield College and the staff of the Statistics Department for being very patient and helpful during the time of my DPhil.

I would like to direct a heartfelt thanks to Hennes & Mauritz AB for their support, Arti Zeighami for making this journey possible, Errol Koolmeister for

providing additional computational resources and Marco Trincavelli for guidance and encouragement. Further, I'd like to thank Linus Hartin, Mathias Sjöberg and Jaime Ferrando Huertas for being the best colleagues I could wish for.

I am also indebted to my bosom buddies Alex, Goran, Simon, Sven and Viktor who kept the mood up with board games and MMA even when I was feeling the most hopeless during the DPhil.

Lastly I probably wouldn't have made it through if it weren't for my parents and their unconditional love, support and patience when I was throwing tantrums when my papers got rejected. It is with a great sense of gratitude and achievement to conceive this thesis as a step toward becoming the third generation of doctorates in our family, in honour of my late grandfather's wishes.

Looking back, I wanted to do DPhil in Machine Learning and thought it was just another thing to accomplish. Now having been in the deep trenches of data with Dino, Robin and Geoff, I realize the adventure only begins with the conception of this thesis.

Abstract

Machine Learning is widely applied in industrial applications such as product design and recommendations, sales and churn prediction among many others. This thesis proposes scalable methods geared towards industrial application where we address problems related to dimensionality, uncertainty quantification and inference.

First, we inferring features from images of products that are pertinent to high sales. We propose an additional regularization objective inspired by kernel two-sample testing to separate high-sellers from low-sellers in latent space for generative models. Experiments demonstrate that we can achieve linear separation in latent space and consequently infer important features using lasso methods.

Next, we develop a non-parametric test for causal association. We propose the *backdoor-Hilbert Schmidt Independence Criterion* with importance weights that encode the *interventional distribution*. Experiments demonstrate that the proposed test is calibrated and has good power.

We then propose *Kernel Fried Tensor (KFT)*, which is a tensor factorization model augmented with kernelized side information, designed to handle high-dimensional forecasting. To improve the utility KFT, we derive a variational inference scheme to provide uncertainty quantification around our estimates. Experiments demonstrate competitive forecasting results and *calibrated* uncertainty quantification.

Further, we consider *right-censored* data where existing survival models do not have a tractable likelihood and evaluate on non-*proper-scores*. We propose *survival monotonic neural networks* (SuMo-net) to train and evaluate on the *right-censored likelihood*, a proper score. Experiments demonstrate that SuMo-net achieves improved likelihood scores with lower inference times.

As many of our proposed methods rely on kernel matrix vector multiplication (KMVM), they are bottlenecked by the $\mathcal{O}(n^2)$ memory and complexity. We propose *Faster Fast and Free Memory Method* (F^3M) and achieve $\mathcal{O}(n)$ memory and complexity for tall ($n \leq 10^9$) and skinny data ($D \leq 7$).

In summary, the methods proposed in this thesis provide a collection of scalable methods for various use cases often encountered in industrial settings and contributes both through novel methodology and statistical characterization of the tasks, which hopefully inspires further research in the machine learning community.

Contents

1 Introduction and Literature review **1**
- 1.1 Motivation and Contribution . . . 2
- 1.2 Background . . . 5
  - 1.2.1 Matrix and Tensor operations . . . 5
    - 1.2.1.1 Setup and Notation . . . 5
    - 1.2.1.2 Tensor factorization . . . 6
  - 1.2.2 Reproducing Kernel Hilbert Space . . . 8
  - 1.2.3 Causal Inference . . . 11
  - 1.2.4 Survival regression . . . 14
- 1.3 Thesis Outline . . . 16

2 Towards Deep Interpretable Features **21**

3 A Kernel Test for Causal Association via Noise Contrastive Backdoor Adjustment **49**

4 Large Scale Tensor Regression using Kernels and Variational Inference **104**

5 Survival Regression with Proper Scoring Rules and Monotonic Neural Networks **157**

6 Giga-scale Kernel Matrix Vector Multiplication on GPU **175**

7 Conclusion and Discussion **197**
- 7.1 Contributions . . . 197
- 7.2 Limitations and open questions . . . 198
  - 7.2.1 DIF . . . 198
  - 7.2.2 bd-HSIC . . . 199
  - 7.2.3 Kernel Fried Tensor . . . 199
  - 7.2.4 F^3M . . . 200
- 7.3 Closing remarks . . . 201

Bibliography **202**

1

Introduction and Literature review

Contents

1.1 Motivation and Contribution 2
1.2 Background 5
1.2.1 Matrix and Tensor operations 5
1.2.2 Reproducing Kernel Hilbert Space 8
1.2.3 Causal Inference 11
1.2.4 Survival regression 14
1.3 Thesis Outline 16

This dissertation is written as an integrated thesis and consists of a collection of 5 papers with one published in *Northern Lights Deep Learning (NLDL) 2019* and to appear in *Journal of Computational Mathematics and Data Science (JCMDS)*, one currently under review at *Journal of Machine Learning Research (JMLR)*, one published in *Springer Machine Learning Journal (MLJ) 2021*, one published in *Proceedings of Artificial Intelligence and Statistics (AISTATS) 2022* and one published at *Neural Information Processing Systems (NeurIPS) 2022*, respectively. These articles are presented in chapters 2, 3, 4, 5 and 6. Each article is an independent piece of work which contains a literature review, a bibliography and an appendix. While the papers can be read independently, their overarching aim is to develop large scale methods for different kinds of data for different kinds of application areas. Collectively, these papers should provide nuance and intuition on

how modelling and approximation techniques need to be adapted to the structure of the data and the task that is being considered to achieve scale.

This introductory chapter is intended to give an overview of tensor operations, core concepts and definitions of the *Reproducing Kernel Hilbert Space (RKHS)*, causality and survival modeling we will need for a basic understanding of each chapter. This chapter is concluded with a brief synopsis of the five aforementioned chapters/articles.

1.1 Motivation and Contribution

Data-driven digital services are becoming increasingly intertwined with everyday life. These days, transportation, shopping, food and various services are frequently purchased on the phone or computer. For many vendors and service providers, this implies a very large collection of data that are often used to make inferences about customer preferences or improve customer experience. To achieve inference and customer experience improvement at scale, machine learning is often used to learn their behaviour from large quantities of data and then deployed to the end user often as a service or product development.

More concretely, improving customer experience through data occurs in several ways such as data-driven product design for improved products, marketing attribution for more relevant promotions, sales forecasting for improved inventory and item availability, product recommendation to reduce search times, and churn prediction for improved user experience among many others.

Generative modelling has recently gained traction in product design [Nobari et al., 2021]. While generative models are good at generating realistic samples, we consider the problem of extracting features that reflect customer preference in Chapter 2. To adequately extract these features, the problem translates into imposing constraints in the latent space of a generative model to isolate more preferred products from

less preferred products and further how to isolate and visualize relevant features.

Marketing attribution considers the problem of correctly attributing the effect of marketing on sales or any observable metric. This effect is often confounded by factors beyond the marketing itself, making the attribution problem a *causal inference* problem, which we consider in Chapter 3. We specifically consider the case of *non-parametric* causal inference with high-dimensional confounders, which is a setting of broad interest beyond marketing attribution.

In many prediction and forecasting tasks, flexible methods such as neural networks or boosting trees are unable to accommodate both high-dimensional and large scale applications, which only leaves less flexible methods such as linear regression or factorization machines. In Chapter 4, we propose *KernelFriedTensor (KFT)* as a tensor factorization model specifically tailored for both high-dimensional and large scale prediction and forecasting. As the tensor model accommodates high-dimensional data, the problem then turns to formulating a tensor model that allows for incorporating side-information and achieving model flexibility at scale, which is of general applicability to any dataset with high dimensional categorical features.

Churn prediction considers the task of predicting when customers will "unsubscribe" or stop using a service. The problem can both be cast as a classification problem or survival regression problem, where an immediate advantage of survival regression over the classification interpretation, is that it provides a survival distribution which can vary with time, whereas a classification only provides a point prediction. We focus on the latter in Chapter 5, and consider flexible survival regression at scale. However, as many existing methods do not have tractable likelihood and rely on numerical integration to output the prediction across time, the question shifts to endowing survival models with tractable likelihood at scale in a principled manner.

As many of our proposed methods are reliant on kernel methods, we dedicate Chapter

6 to improve the fundamental operation *kernel matrix vector multiplication* to scale beyond the conventional $\mathcal{O}(n^2)$ complexity and memory. While there currently exists methods for kernel interpolation in Aussal and Bakry [2019], Greengard et al. [2020], Carrier et al. [1988], Ambikasaran et al. [2015] among many others, none of them scale to $n = 10^9$ on GPU, which is necessary for training models fast at scale. The challenge then becomes implementing, improving upon the approximations and parallelizing the existing FFM method on GPU, which has limited memory capacity.

We summarize the main contributions as follows:

1. A framework termed *Deep Interpretable Features (DIF)* for extracting features in binary classification.
2. A non-parametric test termed *backdoor-HSIC (bd-HSIC)*, which is used to test causal association for graphs satisfying the backdoor-criteria.
3. A tensor framework termed *Kernel Fried Tensor*, which enables learning high-dimensional data with side information in both a frequentist and Bayesian manner.
4. A survival regression framework termed *Survival Monotonic Neural-Network (SuMo-net)*, which proposes *monotonic* neural networks to train on the right-censored likelihood directly, enabling evaluation on a tractable likelihood.
5. A method termed *Faster Fast and Free Memory Method (F^3M)*, capable of scaling kernel matrix vector multiplication for tall ($10^8 \sim 10^9$) and skinny ($D \leq 7$) data with $\mathcal{O}(n)$ memory footprint and controllable error tolerance on GPU.

It is the author's sincere hope that this thesis will provide a useful reference for various non-parametric and flexible techniques in large scale settings, with applications both within and outside of industrial settings.

1.2 Background

1.2.1 Matrix and Tensor operations

Machine learning and statistics are widely dependent on computational linear algebra in estimating models. The most regularly used operation is matrix multiplication. The Hadamard product is oftentimes used in the kernel literature, which relies on weighted sums of kernel matrices on tasks such as regression and hypothesis testing. In tensor factorization, the mode product is frequently used in factorizing tensors. We define these operations mathematically and introduce notation in the sections below.

1.2.1.1 Setup and Notation

A matrix $\mathbf{A} \in \mathbb{R}^{m \times n}$ can always be factorized into two lower dimensional matrices $\mathbf{A} = \mathbf{P} \cdot \mathbf{Q}^\top, \quad \mathbf{P} \in \mathbb{R}^{m \times p}, \mathbf{Q} \in \mathbb{R}^{n \times p}$. In many applications, an approximate factorization is desired, then $\mathbf{P}, \mathbf{Q}$ are optimized to reconstruct $\mathbf{A}$ as accurately as possible. Usually, the accuracy is defined by the squared error, i.e. we optimise

$$\|\mathbf{A} - \mathbf{P} \cdot \mathbf{Q}^\top\|_F^2,$$

where $\|\cdot\|_F$ denotes the Frobenius norm.

For two matrices $\mathbf{A}, \mathbf{B} \in \mathbb{R}^{m \times n}$, their *Hadamard product* is taken as

$$\mathbf{A} \circ \mathbf{B} := \begin{bmatrix} a_{1,1}b_{1,1} & \dots & a_{1,m}b_{1,m} \\ \vdots & \ddots & \\ a_{n,1}b_{n,1} & & a_{n,m}b_{n,m} \end{bmatrix}. \tag{1.1}$$

We define the *sum operator* for a matrix $\mathbf{A} \in \mathbb{R}^{n \times m}$ as

$$(\mathbf{A})_{++} := \sum_{i=1}^{n} \sum_{j=1}^{m} a_{i,j}. \tag{1.2}$$

For two matrices $\mathbf{A} \in \mathbb{R}^{m \times n}$, $\mathbf{B} \in \mathbb{R}^{p \times q}$, their *Kronecker product* is taken as

$$\mathbf{A} \otimes \mathbf{B} := \begin{bmatrix} a_{1,1}\mathbf{B} & \dots & a_{1,m}\mathbf{B} \\ \vdots & \ddots & \\ a_{n,1}\mathbf{B} & & a_{n,m}\mathbf{B} \end{bmatrix} \in \mathbb{R}^{mp \times qn}. \tag{1.3}$$

Building on the Kronecker product, we are now ready to define the *Khatri-Rao product*. Consider matrices $\mathbf{A} = \left[\begin{array}{c|c} \mathbf{A}_{11} & \mathbf{A}_{12} \\ \hline \mathbf{A}_{21} & \mathbf{A}_{22} \end{array}\right] = \left[\begin{array}{cc|c} 1 & 2 & 3 \\ 4 & 5 & 6 \\ \hline 7 & 8 & 9 \end{array}\right]$ and $\mathbf{B} = \left[\begin{array}{c|c} \mathbf{B}_{11} & \mathbf{B}_{12} \\ \hline \mathbf{B}_{21} & \mathbf{B}_{22} \end{array}\right] = \left[\begin{array}{c|cc} 1 & 4 & 7 \\ \hline 2 & 5 & 8 \\ 3 & 6 & 9 \end{array}\right]$. The Khatri-Rao product is then given by

$$\mathbf{A} * \mathbf{B} = \left[\begin{array}{c|c} \mathbf{A}_{11} \otimes \mathbf{B}_{11} & \mathbf{A}_{12} \otimes \mathbf{B}_{12} \\ \hline \mathbf{A}_{21} \otimes \mathbf{B}_{21} & \mathbf{A}_{22} \otimes \mathbf{B}_{22} \end{array}\right] = \left[\begin{array}{cc|cc} 1 & 2 & 12 & 21 \\ 4 & 5 & 24 & 42 \\ \hline 14 & 16 & 45 & 72 \\ 21 & 24 & 54 & 81 \end{array}\right] \tag{1.4}$$

It should be noted that Khatri-Rao product extends to arbitrary partitions $\mathbf{A}_{ij}$ and $\mathbf{B}_{ij}$. A commonly used variation of the Khatri-Rao product is the *transposed Khatri-Rao product*, which for matrices $\mathbf{C} = \left[\begin{array}{c} \mathbf{C}_1 \\ \hline \mathbf{C}_2 \\ \hline \mathbf{C}_3 \end{array}\right]$ and $\mathbf{D} = \left[\begin{array}{c} \mathbf{D}_1 \\ \hline \mathbf{D}_2 \\ \hline \mathbf{D}_3 \end{array}\right]$ is defined as

$$\mathbf{C} \bullet \mathbf{D} := \left[\begin{array}{c} \mathbf{C}_1 \otimes \mathbf{D}_1 \\ \hline \mathbf{C}_2 \otimes \mathbf{D}_2 \\ \hline \mathbf{C}_3 \otimes \mathbf{D}_3 \end{array}\right]. \tag{1.5}$$

Intuitively, the difference between the transposed Khatri-Rao product and the regular Khatri-Rao product is that the transposed Kronecker product specifically only occurs row-wise. The Khatri-Rao and tranposed Khatri-Rao product are particularly useful for expressing advanced tensor sums.

1.2.1.2 Tensor factorization

Often in machine learning, it is useful to represent data in higher dimensional objects than vectors and matrices – i.e. their generalization, tensors. A tensor $\mathbf{V} \in \mathbb{R}^{n_1 \times \ldots \times n_p}$, $p \in \mathbb{N}_{>0}$ admits a *mode product*, where a matrix $\mathbf{U} \in \mathbb{R}^{J \times n_i}$ applied to the kth mode of the tensor, where $k = 1, \ldots, p$. On an elementwise scale, this product is defined as

$$(\mathbf{V} \times_i \mathbf{U})_{i_1 \ldots j \ldots i_p} := \sum_{i_k=1}^{n_k} v_{i_1 \ldots i_k \ldots i_p} \, u_{j i_k}. \tag{1.6}$$

The resulting tensor $\mathbf{V} \times_i \mathbf{U}$ has dimensions $n_1 \times \ldots \times J \times \ldots \times n_p$. A tensor can also be factorized into lower dimensional components, however, in contrast to its matrix counterpart the factorization can vary significantly due to more than two dimensions being factorized. We briefly introduce three common strategies below.

Canonical Polyadic decomposition Canonical Polyadic (CP) decomposition was first introduced in Hitchcock [1927], where one considers a p dimensional tensor $\mathbf{V} \in \mathbb{R}^{n_1 \times \ldots \times n_p}$ that is decomposed as

$$\mathbf{V} \approx \sum_{i=1}^{r} \lambda_i \mathbf{a}_{1,i} \otimes \mathbf{a}_{2,i} \otimes \cdots \otimes \mathbf{a}_{p,i} \tag{1.7}$$

where $\lambda_i \in \mathbb{R}$, $\mathbf{a}_{k,i} \in \mathbb{R}^{n_k}, k = 1, \ldots, p$ and r denotes the rank of the tensor decomposition. This rank is chosen, where a higher r usually implies a more flexible decomposition. CP decomposition can be seen as a generalization of *Singular Value Decomposition*. See Figure 1.1 for a graphical illustration.

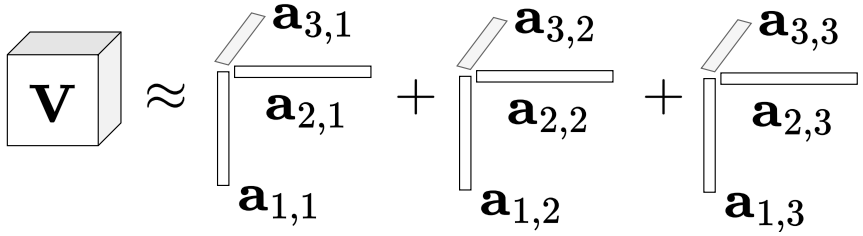


Figure 1.1: 3D example of CP decomposition.

Tucker decomposition Tucker decomposition Tucker [1966] also considers the tensor $\mathbf{V}$. However, it instead decomposes the tensor as

$$\mathbf{V} \approx \prod_{k=1}^{p} \mathbf{G} \times_k \mathbf{U}_k \tag{1.8}$$

where $\mathbf{G} \in \mathbb{R}^{r_1 \times \ldots \times r_p}$ and $\mathbf{U}_k \in \mathbb{R}^{n_k \times r_k}$. Here $\mathbf{G}$ is a p-dimensional core tensor, where each dimension has rank $r_k, \quad k = 1, \ldots, p$. Each rank r_k is chosen accordingly, where a higher rank implies a more flexible decomposition. If one chooses $r_k = 1, \quad \forall k$, CP decomposition is recovered. See Figure 1.2 for a graphical illustration.

Tensor-Train decomposition A more recent factorization strategy is the Tensor-Train decomposition introduced in Oseledets [2011]. Here, $\mathbf{V}$ is decomposed into p latent tensors $\mathbf{U}_k \in \mathbb{R}^{r_k \times n_k \times r_{k-1}}, k = 1, \ldots, p$, with $\mathbf{U}_1 \in \mathbb{R}^{r_1 \times n_1 \times 1}$ and $\mathbf{U}_p \in \mathbb{R}^{1 \times n_p \times r_{p-1}}$. Here r_k is the latent dimensionality for each component in the

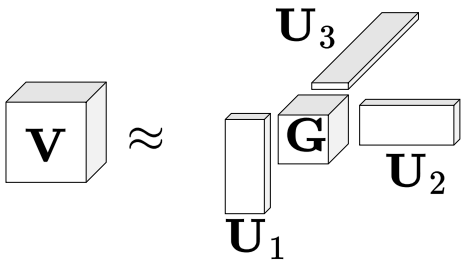


Figure 1.2: 3D example of Tucker decomposition.

decomposition, with $r_1 = r_p = 1$. Let $\times_{-1}\mathbf{U}_k$ be the operation of applying the mode product to the last dimension of $\mathbf{U}_k$. We seek $\mathbf{U}_1 \dots \mathbf{U}_p$ so that

$$\mathbf{V} \approx \prod_{k=1}^{p} \times_{-1}\mathbf{U}_k, \quad \text{where each element is} \quad v_{i_1 \dots i_p} \approx \sum_{r_0 \dots r_p} \prod_{k=1}^{p} u_{r_k i_k r_{k-1}}. \tag{1.9}$$

Similarly to other factorization methods, larger r_k's often leads to a more flexible model. See Figure 1.3 for a graphical illustration of the decomposition.

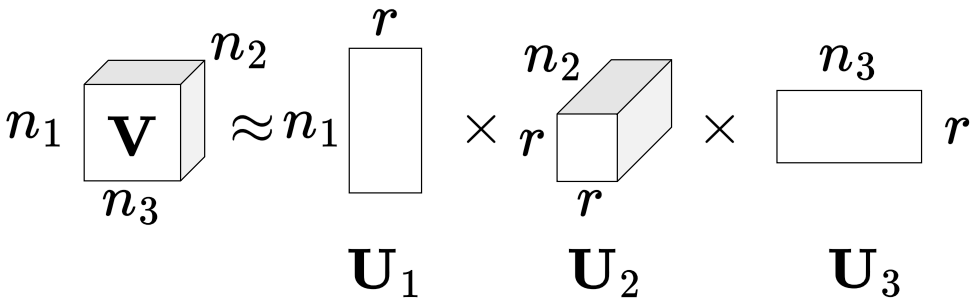


Figure 1.3: 3D example of Tensor-Train decomposition.

1.2.2 Reproducing Kernel Hilbert Space

The Reproducing Kernel Hilbert Space (RKHS) was introduced in Aronszajn [1950]. Since then it has become the foundation for kernel methods in Machine Learning, with the kernel perceptron first introduced in Aizerman [1964]. Kernel methods were popularized in machine learning with the invention of kernel support vector machines Boser et al. [1992], which at the time of its introduction had competitive performance compared to neural networks LeCun et al. [1989] on the task of classifying handwritten digits. Since then, kernel methods have become ubiquitous in the machine learning literature, with applications in hypothesis testing [Gretton

et al., 2005], regression [Saunders et al., 1998], probabilistic modelling [Rasmussen and Williams, 2005] and deep learning [Jacot et al., 2018] among many other areas. We briefly introduce some fundamental concepts of the RKHS and kernel methods below.

The theorems follow the expositions from textbooks Steinwart and Christmann [2008], Berlinet and Thomas-Agnan [2011] and further details and a broader overview can be found therein. First, we define what an *evaluation functional* is.

Definition 1. *(Evaluation functional). Let $\mathcal{H}$ be a Hilbert space of functions $f : \mathcal{X} \to \mathbb{R}$, defined on a non-empty set $\mathcal{X}$. For a fixed $x \in \mathcal{X}$, map $\delta_x : \mathcal{H} \to \mathbb{R}$, $\delta_x : f \mapsto f(x)$ is called the (Dirac) evaluation functional at x.*

Evaluation functionals are fundamental in defining an RKHS. The evaluation functional δ_x can intuitively be understood as an operator that "instantiates" a functional f at x. The following theorem demonstrates the properties of linear operators.

Theorem 1. *Let $(\mathcal{F}, \|\cdot\|_{\mathcal{F}})$ and $(\mathcal{G}, \|\cdot\|_{\mathcal{G}})$ be normed linear spaces. If L is a linear operator, then the following three conditions are equivalent:*

1. *L is a bounded operator.*
2. *L is continuous on $\mathcal{F}$.*
3. *L is continuous at one point of $\mathcal{F}$.*

As the evaluation functional δ_x is a linear operator, it follows from Theorem 1 that it also is a bounded and continuous operator. Using this equivalence we can define an RKHS:

Definition 2. *(Reproducing kernel Hilbert space) A Hilbert space $\mathcal{H}$ of functions $f : \mathcal{X} \to \mathbb{R}$, defined on a non-empty set $\mathcal{X}$ is said to be a Reproducing Kernel Hilbert Space (RKHS) if δ_x is continuous $\forall x \in \mathcal{X}$.*

While the above definition technically defines an RKHS, it does not give an intuitive meaning to the reproducing property. To see how an RKHS is "reproducing", we first define a *reproducing kernel*.

Definition 3. *(Reproducing kernel) 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 $\mathcal{H}$ if it satisfies*

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

It should further be noted that for any $x, y \in \mathcal{X}$

$$k(x, y) = \langle k(x, \cdot), k(y, \cdot)\rangle_{\mathcal{H}}. \tag{1.10}$$

It is straightforward to see how such a kernel would be uniquely associated with an RKHS if it existed. Its existence is proved in the following theorem.

Theorem 2. *(Existence of the reproducing kernel) $\mathcal{H}$ is a reproducing kernel Hilbert space (i.e., its evaluation functionals δ_x are continuous linear operators), if and only if $\mathcal{H}$ has a reproducing kernel.*

This theorem tells us that every RKHS is guaranteed to have a reproducing kernel and vice versa. The key insight here is that a reproducing kernel can define an entire RKHS. An important property of the reproducing kernel is that it is positive definite:

Definition 4. *(Positive definite functions) A symmetric function $h : \mathcal{X} \times \mathcal{X} \to \mathbb{R}$ is positive definite if $\forall n \geq 1, \forall (a_1, \ldots a_n) \in \mathbb{R}^n, \forall (x_1, \ldots, x_n) \in \mathcal{X}^n$,*

$$\sum_{i=1}^{n} \sum_{j=1}^{n} a_i a_j h(x_i, x_j) \geq 0.$$

It turns out that all reproducing kernels are positive definite. We start by defining a *kernel*.

Definition 5. *(Kernel) Let $\mathcal{X}$ be a non-empty set. The function $k : \mathcal{X} \times \mathcal{X} \to \mathbb{R}$ is said to be a kernel if there exists a real Hilbert space $\mathcal{H}$ and a map $\phi : \mathcal{X} \to \mathcal{H}$ such that $\forall x, y \in \mathcal{H}$,*

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

Here we refer to ϕ as a *feature map* and $\mathcal{H}$ the feature space. Note that a feature map is not uniquely associated with a kernel. The following lemma then states that all reproducing kernels are positive definite:

Lemma 1. *Let $\mathcal{F}$ be any Hilbert space (not necessarily an $RKHS$), $\mathcal{X}$ a nonempty set and $\phi : \mathcal{X} \to \mathcal{F}$. Then $h(x, y) := \langle \phi(x), \phi(y) \rangle_{\mathcal{F}}$ is a positive definite function.*

From the lemma, it follows immediately that a reproducing kernel is positive definite by setting $\phi : x \mapsto k(x, \cdot)$ for a reproducing kernel k. It turns out that a positive definite kernel defines an RKHS.

Theorem 3. *(Moore-Aronszajn) Let $k : \mathcal{X} \times \mathcal{X} \to \mathbb{R}$ be positive definite. Then there is a unique RKHS $\mathcal{H} \subset \mathbb{R}^{\mathcal{X}}$ with reproducing kernel k.*

Theorem 3 completes the circle in how to define an RKHS, where we first defined it from a linear evaluation functional perspective and then associated a reproducing kernel with an RKHS, and finally showed that reproducing kernels are positive definite and that they define an RKHS as well.

1.2.3 Causal Inference

Causal Inference is the process of isolating the independent actual effect of a certain treatment that is an element of a larger system. The process of isolating such an effect is of great importance in subjects such as epidemiology [Robins et al., 2000], economics [Imbens and Angrist, 1994], brain imaging [Ramsey et al., 2009], and retail Aguilar-Palacios et al. [2021] among many others. While there is an ongoing debate on the definition of causality from a philosophical perspective, we abide by the statistical interpretation presented through frameworks of *Causal Graphs* [Pearl, 2009b]. Other framework include, but are not limited to *potential outcomes* [Rubin, 2005], *g-formula* [Robins, 1986] and *decision-theoretic* [Dawid, 2000].

We first recall core definitions from Lauritzen [1996] and Pearl [2009a], starting with a *directed graph*.

Definition 6. *(Directed Graph) A directed graph $\mathcal{G}$ is a pair (V, D), where*

- *V is a finite set of vertices; and*
- *$D \subseteq V \times V$ is a collection of edges, which are ordered pairs of vertices. Loops (i.e. edges of the form (v, v)) are not allowed.*

If vertices $(v, w) \in D$ we write $v \to w$, and say that v is a *parent* of w, and conversely w a *child* of v. Using Pearl's framework, statistical relationships (the edges) between covariates (the vertices) are modelled by a directed graph, where the direction of the edges would represent the direction of the causation. A directed graph can contain a *directed cycle*, which is defined below.

Definition 7. *(Directed cycle) A graph contains a directed cycle if there is a directed path from v to w together with an edge $w \to v$. A directed graph is acyclic if it contains no directed cycles. We call such graphs* directed acyclic graphs *(DAGs).*

Causal inference with acyclic graphs is conceptually much simpler than with cyclic graphs, as the meaning of a cycle is a matter of significant disagreement, see Bongers et al. [2021] and references therein.

The causal effect of a directed edge can be quantified if the underlying probability distribution has a density and factorizes such that it is consistent with its graphical representation.

Definition 8. *(Factorization Property). Let $\mathcal{G}$ be a directed acyclic graph with vertices V. We say that a probability density function $p(x_V)$ factorizes with respect to $\mathcal{G}$ if*

$$p(x_V) = \prod_{v \in V} p\left(x_v \mid x_{\mathrm{pa}_{\mathcal{G}}(v)}\right), \quad x_V \in \mathcal{X}_V.$$

here $\mathrm{pa}_{\mathcal{G}}(v)$ refers to the set of parent nodes of v.

We will generally assume that a DAG accurately depicts a probability distribution.

Definition 9. *(Causal Markov Assumption) [Scheines, 1997] Let $\mathcal{G}$ be a directed acyclic graph with vertices V, $p(x_v), v \in V$ be a probability distribution over the vertices and* $\mathrm{nd}(v)$ *denote the non-descendants of the node v. Let $\mathcal{G}$ and p satisfy the* Causal Markov Assumption *if*

$$\forall v \in V: \quad X_v \perp X_{\mathrm{nd}(v)\setminus \mathrm{pa}(v)} | X_{\mathrm{pa}(v)} \tag{1.11}$$

This essentially means that conditional on the set of all its direct causes, a node is independent of all variables which are not direct causes or direct effects of that node. While a graph may represent a correct probability distribution, the converse where a probability distribution corresponds to the correct graph requires an additional assumption.

Definition 10. *(Causal Faithfulness Assumption) [Scheines, 1997] If all independence relations in the probability distribution are a consequence of the Causal Markov condition, then the probability distribution is* faithful

Intuitively the causal faithfulness assumption means that conditional independence relations represent in a probability distribution can be represented exactly by the graph. The final assumption we need is to assume that we don't have any unmeasured confounders:

Definition 11. *(Causal Sufficiency) [Scheines, 1997] If the set of measured variables M include all of the common causes of pairs in M, then* causally sufficiency *is satisfied.*

To satisfy the requirement, it intuitively implies that we have no unmeasured confounders, meaning we have measured everything there is to know about the system we are trying to model. Hence this assumption almost always turns out to be problematic, because we have very little means to validate causal sufficiency in practice.

In causal modelling, we can *intervene* on the graph by setting $P(Y = y \mid do(T = t))$, where $do(T = t)$ means fixing the system such that the random variable T is a

constant value t. Intervention is however not a probabilistic operation, and hence the resulting *interventional distribution* depends on the causal structure of the graph. Note that intervening is generally different from observing $P(Y = y \mid T = t)$, since intervening may involve marginalizing out confounders.

Definition 12. *(Intervention) [Peters et al., 2017] Let $\mathcal{G}$ be a directed acyclic graph representing a causal system and let p be a probability distribution over the variables X_V. An intervention on a variable X_t (for $t \in V$) does two things:*

- *graphically we represent this by removing edges pointing into t (i.e. of the form $v \to t$);*
- *probabilistically, we replace our usual factorization*

$$p\left(x_V\right) = \prod_{v \in V} p\left(x_v \mid x_{\mathrm{pa}(v)}\right)$$

with

$$p\left(x_{V \setminus \{t\}} \mid do\left(x_t\right)\right) = \frac{p\left(x_V\right)}{p\left(x_t \mid x_{\mathrm{pa}(t)}\right)} = \prod_{v \in V \setminus \{t\}} p\left(x_v \mid x_{\mathrm{pa}(v)}\right).$$

Anecdotally, one can view interventions as "fixating" a treatment. Consider an experiment where experimental pills are given to members of a cohort depending on the age of the member. An intervention would correspond to looking at the distribution of the outcome had we given everyone in the cohort pills regardless of age.

1.2.4 Survival regression

Survival analysis is a branch of statistics for analyzing the expected duration of time until one event occurs, such as death in biological organisms [Ziehm and Thornton, 2013] or failure in mechanical systems [Susto et al., 2015]. In particular, survival regression is the topic of relating the time that passes before some event occurs, to one or more covariates that may be associated with that quantity of time. Classical models for survival regression include the Cox proportional hazards model [Cox, 1972] and the accelerated failure time model [MacKenzie, 1982]. Recently, many machine learning methods for survival data have been proposed, which allow more flexible survival functions than the Cox model such as [Kvamme et al., 2019], [Lee

et al., 2018] and [Katzman et al., 2018] among many. We give a brief introduction to some fundamental concepts in survival regression below.

When conducting survival regression, we associate covariates $x_i \in \mathbb{R}^d$ with event times $T_i \in \mathbb{R}$, where event time refers to the time between the start of the study until a certain event of interest occurs, such as recovering from a disease or defaulting on a loan. The regression model is a survival distribution f_θ, where the parameters θ are trained to maximize the *survival log-likelihood*, defined below.

Definition 13. *(Survival log-likelihood) The survival log-likelihood is given by*

$$\mathcal{S}_L(f_\theta, t, x) = \log f_\theta(t|x). \tag{1.12}$$

The survival log-likelihood assumes that all the event times are observed, however, this assumption is often not very practical as participants in the experiment might drop out of the study. In such cases, we say that the event-times are *right-censored* and consider the *censoring-time* $C_i \in \mathbb{R}$, which denotes the time between the start of the study until a participant becomes unobserved. Instead of observing T_i, we rather observe $Z_i = \min(T_i, C_i)$ and $d_i = 1_{T_i<C_i}$. Consequently, we must instead consider the *right-censored log-likelihood* when conducting survival regression.

Definition 14. *(Right-censored survival log-likelihood)*

$$\mathcal{S}_L(S(z|x), (z, d)) = d \log f(z|x) + (1 - d) \log S(z|x). \tag{1.13}$$

Here $S(z|x) := 1 - F_\theta(z|x)$, were $F_\theta(z|x)$ denotes the cumulative survival distribution. It should be noted that many survival models [Kvamme et al., 2019] do not optimize likelihood directly and optimize a surrogate objective such as *partial likelihood*. When optimizing survival distributions on different objective (or *scores*), it is important to use a *proper scoring rule*:

Definition 15. *(Proper scoring rule) A scoring rule $\mathcal{S}$ is called proper if the true distribution achieves the optimal expected score, i.e. if*

$$\mathbb{E}_{y\sim S}\mathcal{S}(S, y) \geq \mathbb{E}_{y\sim S}\mathcal{S}(\hat{S}, y) \quad \textit{for all distributions } \hat{S}.$$

A proper scoring rule essentially guarantees that optimizing it ensures finding the true survival distribution, which can be critical in healthcare applications involving survival modeling.

1.3 Thesis Outline

This integrated thesis consists of five articles (chapters) proposing methodologies for handling large scale data in different contexts. While these articles were written independently, they collectively make up a toolbox to model data in a variety of learning and inference tasks. In this section, we briefly introduce the articles and summarize their contributions. All code used in the papers is publicly available. Following each chapter, there will also be a signed statement of authorship detailing my contributions, per thesis guidelines.

Towards Deep Interpretable Features While Machine Learning is rapidly growing in both academia and the industry, it is also experiencing growing pains in regards to the lack of transparency and interpretable models in areas such as medicine and engineering [Gilpin et al., 2018, Rudin, 2019]. In chapter 2, we consider on the problem of binary image classification and focus on extracting and visualizing interpretable features that affect the classification boundary. To achieve this, we introduce the *Deep Interpretable Features (DIF)* framework, which consists of a latent variable model as a medium to extract interpretable features which are then fed to a downstream task of binary classification. DIF is built on desiderata, which propose geometric separation in latent space to be consistent with visual distinction in images. This geometric separation is achieved by combining the ideas from kernel two-sample testing with that of generative modelling. More concretely, we propose a novel regularization objective for generative models that can geometrically separate semantically different objects in latent space. Experiments on a wide variety of datasets demonstrate that DIF can retain the classification performance of a black-box model while having interpretable features.

The article was initially published as a conference paper, referenced as:

- **Hu, R.** and Sejdinovic, D. (2019) Robust Deep Interpretable Features for Binary Image Classification. *Proceedings of the Northern Lights Deep Learning (NLDL 2021)*

An extended version, currently under review, is presented in this thesis and referenced as:

- **Hu, R.** and Sejdinovic, D. (2023) Towards Deep Interpretable Features. To appear in *Journal of Computational Mathematics and Data Science (2023)*
- Codebase: `https://github.com/MrHuff/DIF`

A Kernel Test for Causal Association via Noise Contrastive Backdoor Adjustment Causal inference in a modern setting often involves large datasets with a high number of confounders with different properties. More often than not, these settings such as epidemiology [Rothman and Greenland, 2005], brain imaging [Castro et al., 2020] and retail [Moriyama and Kuwano, 2021] among many, do not have access to randomized controlled trials. To conduct causal inference in such contexts, it is often a necessary step to begin with establishing the existence of a causal association before quantifying it exactly. In chapter 3 we introduce *backdoor Hilbert-Schmidt Independence Criterion* (bd-HSIC) as a non-parametric test for causal association, building on *G-computation* [Robins, 1986] and HSIC [Gretton et al., 2005]. We establish the convergence properties of the bd-HSIC estimator and demonstrate that it has good power and the correct size across a multitude of settings, including a few where the popular *Post-double selection* (PDS) [Belloni et al., 2014] procedure fails.

The paper is currently under review at JMLR and is referenced as

- **Hu, R.**, Sejdinovic D. and Evans R.J. (2021). A Kernel Test for Causal Association via Noise Contrastive Backdoor Adjustment. *arXiv:2111.13226*
- Codebase: `https://github.com/MrHuff/kgformula`

Large Scale Tensor Regression using Kernels and Variational Inference In chapter 4, we introduce a novel tensor factorization model that incorporates kernelized side information for prediction and forecasting using high-dimensional tensor data. In particular, we propose *Kernel Fried Tensor* (KFT) to improve model flexibility and further extend our model to a Bayesian setting using variational inference. For the variational inference setting, explicit tensor expressions for the *Evidence Lower BOund* (ELBO) are derived and implemented to improve convergence and computational efficiency. Experimental results showcase superior performance over existing methods such as *LightGBM* [Ke et al., 2017] and *Field-aware Factorization Machines* [Juan et al., 2016]. Bayesian experiments on both prediction and forecasting tasks further suggest that the variational inference scheme yields calibrated uncertainty quantification.

The article is referenced as:

- **Hu, R.**, Nicholls, G.K. and Sejdinovic, D. (2020) Large scale tensor regression using kernels and variational inference. *Mach Learn (2021).*
- Codebase: `https://github.com/MrHuff/KernelFriedTensor`

Survival Regression with Proper Scoring Rules and Monotonic Neural Networks In chapter 5, we consider survival regression for right-censored data. There are many previously proposed models such as Cox-Time [Kvamme et al., 2019], DeepHit [Lee et al., 2018] and DeepSurv [Katzman et al., 2018] among many. These models use popular scoring rules such as time-dependent concordance [Antolini et al., 2005], CRPS [Avati et al., 2020], integrated binomial log-likelihood and integrated Brier score, however as we show in the article, it turns out that none of these scores are *proper scoring rules*, meaning that the true survival distribution may be scored worse than incorrect distributions, leading to inaccurate estimation. Despite this flaw, many feed-forward neural network-based survival models still can not train and evaluate on right-censored likelihood directly, i.e. resorting to approximate methods. We propose *Survival Monotonic Neural Network* (SuMo-net), a simple feed-forward neural network with a monotonic restriction that is

capable of directly training and evaluating on the right-censored log-likelihood. Experiments demonstrate that Sumo-net both has computational benefits and achieves state-of-the-art log-likelihood scores on several different datasets.

The article is referenced as:

- Rindt, D.*, **Hu, R.***, Steinsaltz, D. and Sejdinovic, D. (2020) Survival Regression with Proper Scoring Rules and Monotonic Neural Networks. *Proceedings of the 25th International Conference on Artificial Intelligence and Statistics (AISTATS 2022)*[1]
- Codebase: `https://github.com/MrHuff/Sumo-Net`

Giga-scale Kernel Matrix Vector Multiplication on GPU Kernel Matrix Vector Multiplication (KMVM) is a ubiquitous operation frequently occurring in machine learning, image processing and electrostatics. In recent times, the scale of problems has grown to a size where the time and memory complexity of $\mathcal{O}(n^2)$ for direct KMVM no longer remains practically feasible. The pioneering algorithm *Fast Multipole Method* (FMM) [Carrier et al., 1988] amends the time complexity to $\mathcal{O}(n)$ with various reductions in memory complexity, for input data with dimension $D \leq 2$. Subsequent contributions such as [Börm et al., 2019, Greengard et al., 2020] mostly extend FMM to different types of kernels for different physical problems. Recent contributions such as *KeOps* [Charlier et al., 2021] utilizes *map-reduce* schemes to reduce the memory footprint of KMVM to $\mathcal{O}(n)$ on GPU devices and *Fast and Free Memory Method* (FMM) [Aussal and Bakry, 2019], proposes a kernel agnostic method that is experimentally demonstrated to scale linearly in both time and memory, however only on CPU. In chapter 6 we introduce F^{3}M, an extension of FFM to GPU with various computational and memory improvements. Experiments on KMVM and Kernel Ridge Regression problems demonstrate that F^{3}M significantly outperforms FFM and KeOps on GPU with small and controllable errors. It is referenced as:

[1]The paper received an honourable mention best paper award

- **Hu, R.**, Chau, S.L., Sejdinovic, D. and Glaunès, J.A. (2021) Giga-scale Kernel Matrix Vector Multiplication on GPU. To appear in the *Thirty-sixth Conference on Neural Information Processing Systems (Neurips 2022)*
- Codebase: `https://github.com/MrHuff/F3M`

2

Towards Deep Interpretable Features

Journal of Computational Mathematics and Data Science 6 (2023) 100067



Contents lists available at ScienceDirect

Journal of Computational Mathematics and Data Science

journal homepage: www.elsevier.com/locate/jcmds



Towards Deep Interpretable Features



Robert Hu *, Dino Sejdinovic

Department of Statistics, University of Oxford, 29 St Giles', Oxford OX1, United Kingdom

ARTICLE INFO

MSC:
00-01
99-00

Keywords:
Deep generative models
Kernel methods
Interpretability

ABSTRACT

The problem of interpretability for binary image classification is considered through the lens of kernel two-sample tests and generative modeling. A feature extraction framework coined Deep Interpretable Features is developed, which is used in combination with IntroVAE, a generative model capable of high-resolution image synthesis. Experimental results on a variety of datasets, including COVID-19 chest x-rays demonstrate the benefits of combining deep generative models with the ideas from kernel-based hypothesis testing in moving towards more robust interpretable deep generative models.

1. Introduction and related work

While Machine Learning has enjoyed tremendous growth in both academia and industry, it has also caused concern in regard to its transparency and interpretability [1,2] in high-stakes decision making in areas such as medicine and engineering. In this paper, we consider interpretability for binary image classification and focus on extracting interpretable features that affect the classification boundary. In this context, convolutional neural networks (CNN) [3] are often used as they exhibit superior performance. However, due to their complexity, they are often considered black-box models [4,5]. In contrast, simple models such as logistic classification applied directly to the pixel representation would give an interpretation in terms of pixel-wise effects, but these are not meaningful for the overall classification task. A compromise between CNNs and linear models (see Fig. 1) would be to consider lower-dimensional latent representations for each image, where each element in the latent representations corresponds to an extracted feature that is interpretable. Such latent representations can be recovered by using deep generative models, in particular when the latent representations are linearly separated on class. To impose such a structure on the latent space, one can add a regularization objective to the loss function of the generative model. In that scenario, we can extract these features using a linear model and visualize them using the generative component of our model. For the remainder of this paper, we will consider features to be computable quantities given any input and our goal is to discover such quantities that are class-discerning. Our motivation for building such an interpretable framework is to extend the existing post-hoc methods [6,7] in the deep learning setting by providing a more thorough analysis of latent features and exploring deep generative models as fundamentally interpretable models [8,9]. As the aforementioned works mostly rely on prototypes for interpretability, we focus on interpreting continuous latent features. This requires that the learned representations are informative, which can be achieved through *contrastive learning*, exemplified in [10]. While [10] provides ample discussion on how to obtain informative latent representations for downstream tasks, it is not obvious how to use these representations in the context of interpretability. We consider a supervised approach when finding latent features similar to [11], as unsupervised disentanglement methods are demonstrated to be incapable of recovering these features [12]. A supervised method has the added benefit of imposing structure on the latent space, potentially improving the robustness of the generative model to adversarial attacks [13].

* Corresponding author.
E-mail address: robert.hu@stats.ox.ac.uk (R. Hu).

https://doi.org/10.1016/j.jcmds.2022.100067
Received 21 February 2022; Received in revised form 23 October 2022; Accepted 20 November 2022

2772-4158/© 2022 The Author(s). Published by Elsevier B.V. This is an open access article under the CC BY license (http://creativecommons.org/licenses/by/4.0/).

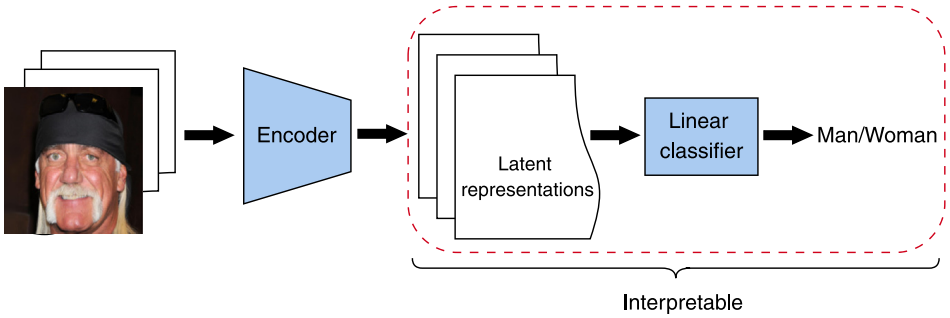


Fig. 1. Logistic classifier applied to latent representations.

We pose the following questions to guide us:

1. How can we find linearly separated latent representations?
2. What implications on performance do such latent representations have?
3. How can we interpret a model using linearly separated latent representations?

The rest of the paper is organized as follows. We start by providing desiderata for a generative model that can be used for our purposes, leading to the choice of Variational AutoEncoders (VAE) and its variants. We then present our main contribution: the framework termed Deep Interpretable Features (DIF) and its key components. The following sections will describe our experimental setup and evaluation criteria, present the empirical results and ablation studies, followed by concluding remarks.

2. Generative modeling desiderata

What type of interpretability are we considering?. Interpretability is generally defined through the end-user of the model and the context it is being used in. We consider the setting where the end-user needs rules or high-level features of images to understand what distinguishes an image in class A from class B, which can be described in layman's terms. In this setting, it is necessary for the end-user to be able to interact and traverse features between examples (see Fig. 6 for an example). This can then be viewed as an additional layer of interpretability beyond comparing examples of different labels that are close to the decision boundary, as a simpler way to achieve an intuition for separation.

We present desiderata based on our application needs and provide examples and motivation for feasible and unfeasible generative model choices.

Desiderata. *Consider generative models* $G : \mathbf{z} \to \mathbf{x}$ *and the latent representations* $\mathbf{z}$ *of images* $\mathbf{x}$ *in the context of interpretability:*

1. $\mathbf{z}$ *is a low dimensional representation of* $\mathbf{x}$.
2. *Semantically dissimilar images are encoded into semantically dissimilar latent representations* $\mathbf{z}$.
3. *Semantically dissimilar latent representations are decoded back into semantically dissimilar images through* G.
4. *All latent representations have a visually meaningful map* $G(\mathbf{z})$.

If desideratum 1 does not hold, $G(\mathbf{z})$ fails to generalize pixels into features. If desideratum 2 does not hold, distinguishing images on a feature-wise basis will fail. If desideratum 3 or 4 fails, then features found in latent space would not bear any visual meaning when decoded back. We briefly review the generative model literature in the context of our desiderata below.

VAE-based models [14] are generally the most feasible, as they have an encoder component that can be trained to directly map images to latent representations. In particular, they encourage desideratum 3 by training on the reconstruction error objective $\mathcal{L}_{\text{recon}}(\mathbf{x}, G(\mathbf{z})) = \|G(\mathbf{z}) - \mathbf{x}\|^2$. A shortcoming of VAEs is the quality of the synthesized images, which is discussed in [15,16]. The disentanglement literature [17,18] of VAEs has proposed additional regularization objectives to encourage desideratum 2. [11] proposes *characteristic capturing*-VAE (CCVAE) to embed label characteristics of images into the latent space by explicitly mapping the to the latent space with an auxiliary objective, suggesting that VAE-based methods offer a high degree of flexibility to satisfy the desiderata.

Generative Adversarial Network (GAN) [19] based models work well due to the adversarial training procedure which encourages desideratum 4. However, they become computationally infeasible; as GANs do not have an encoder, one needs to reverse optimize [20] a GAN generator to obtain the latent representation of each image.

Normalizing flows [21] are not feasible since they do not yield low dimensional latent representations due to their bijection property.

We conclude that VAE-based models are the most suitable choice wherein the problem lies in finding a suitable variation capable of achieving our aims.

3. Deep Interpretable Features (DIF)

We propose our framework Deep Interpretable Features (DIF), which at its core gives any generative model the three following properties:

1. Encourages the generative model desiderata through a training objective
2. Finds interpretable features

We give an overview of our proposed method below.

3.1. Overview

The core idea is to combine kernel two-sample testing with generative modeling. The observation is that if one calculates the two-sample test-statistic using latent representations with their labels, one can then optimize the test-statistic with respect to the representations to maximize the discrepancy between representations of different labels. We then simply parametrize the kernel-two sample test-statistic with learned latent variables from the generative model and the associated label. We then optimize the sum of losses, informally as

$$\mathcal{L}_{\text{Total}} := \mathcal{L}_{\text{Generative model}}(\underbrace{\mathbf{x}}_{\text{image}}) - \Lambda_{\text{Test-statistic}}(\underbrace{\mathbf{z}}_{\text{latent representation}}, \underbrace{\mathbf{y}}_{\text{label}}) \tag{1}$$

using gradient descent. The negative sign in front of the test-statistic comes from that we want to maximize the value of the two-sample test-statistic as it implies maximizing the dissimilarity of latent representations with different labels. One could in principle choose any two-sample test-statistic, however we choose to focus on the Mean Embedding test (ME-test) [22] which enables different types of separation depending on kernel choice, which we outline below.

3.2. The mean embedding test

Two sample testing. Non-parametric hypothesis testing considers whether two independently and identically drawn samples $\{\mathbf{x}_i\}_{i=1}^{n_1} \sim P$ and $\{\mathbf{y}_i\}_{i=1}^{n_2} \sim Q$ from probability distributions P, Q, where $\mathbf{x}_i, \mathbf{y}_i \in \mathbb{R}^d$. The null hypothesis $H_0 : P = Q$ is tested against the general alternative $H_1 : P \neq Q$.

ME-test. We focus on the Mean Embedding test (ME test) [22] which introduces a Hotelling's T-squared statistic computed using a kernel mean embedding, expressed as $\hat{\lambda}_n = n\overline{\mathbf{h}}_n^\top \mathbf{S}_n^{-1}\overline{\mathbf{h}}_n$. Here, $\overline{\mathbf{h}}_n = \frac{1}{n}\sum_{i=1}^n \mathbf{h}_i$, with $\mathbf{h}_i = (k(\mathbf{x}_i, \mathbf{v}_j) - k(\mathbf{y}_i, \mathbf{v}_j))_{j=1}^J \in \mathbb{R}^J$. Further we have that $\mathbf{S}_n = \frac{1}{n-1}\sum_{i=1}^n \left(\mathbf{h}_i - \overline{\mathbf{h}}_n\right)\left(\mathbf{h}_i - \overline{\mathbf{h}}_n\right)^\top$, for n samples. The statistic further depends on a positive definite kernel $k : \mathcal{X} \times \mathcal{X} \to \mathbb{R}$ (with $\mathcal{X} \subseteq \mathbb{R}^d$), and a set of J prototypes $\mathcal{V} = \{\mathbf{v}_j\}_{j=1}^J \subset \mathbb{R}^d$.

The test-statistic intuitively measures the normalized square distance between samples from P and Q in kernel space. This can be interpreted as the distributional difference between samples from P and Q. A small value of $\hat{\lambda}_n$ would imply that P and Q are similar, while a larger value suggests that P and Q are different.

Prototypes. A prototype can be thought of as a typical example of a class. For images, one could for example take the average image of all images from P as a prototype.

Practical considerations. When computing $\hat{\lambda}_n$ we modify the statistic with a regularization parameter $\gamma_n > 0$, giving $\hat{\lambda}_n = n\overline{\mathbf{h}}_n^\top \left(\mathbf{S}_n + \gamma_n \mathbf{I}\right)^{-1}\overline{\mathbf{h}}_n$, for numerical stability. We use a slightly modified version of the ME-test statistic proposed in [22] since our application considers $n_1 \neq n_2$. For different sample sizes, we use the pooled covariance version of the ME-test statistic defined as

$$\begin{aligned}
\hat{\lambda}_{n_1,n_2} &= \frac{n_1 n_2 \overline{\mathbf{h}}_{n_1,n_2}^\top \mathbf{S}_{n_1,n_2}^{-1}\overline{\mathbf{h}}_{n_1,n_2}}{n_1+n_2}, \\
\mathbf{S}_{n_1,n_2} &= \frac{(n_1-1)\mathbf{S}_{n_1} + (n_2-1)\mathbf{S}_{n_2}}{n_1+n_2-2}, \\
\overline{\mathbf{h}}_{n_1,n_2} &= \left(\frac{1}{n_1}\sum_{i=1}^{n_1} k(\mathbf{x}_i, \mathbf{v}_j) - \frac{1}{n_2}\sum_{k=1}^{n_2} k(\mathbf{y}_k, \mathbf{v}_j)\right)_{j=1}^J \in \mathbb{R}^J.
\end{aligned} \tag{2}$$

We are now ready to present our first technique of linearly separating the latent space.

3.3. Generative modeling objective: Mean Embedding-objective

To encourage desideratum 2 we introduce the ME-objective for generative models. We will denote a mapping parametrized by θ as $T_\theta(\cdot)$, labeled objects we wish to separate as $\zeta_i^{\mathbb{X}}, \zeta_j^{\mathbb{Y}}$ and batches of latent representations $\mathbf{b}_{\mathbb{X}} = \{\mathbf{z}_i^{\mathbb{X}}\}_{i=1}^{b_1}$, $\mathbf{b}_{\mathbb{Y}} = \{\mathbf{z}_j^{\mathbb{Y}}\}_{j=1}^{b_2}$ with $\mathbf{z}_i^{\mathbb{X}} = C\tanh\left(\frac{T_\theta(\zeta_i^{\mathbb{X}})}{C}\right)$, $\mathbf{z}_j^{\mathbb{Y}} = C\tanh\left(\frac{T_\theta(\zeta_j^{\mathbb{Y}})}{C}\right)$ and labeled prototypes $\mathbf{v}_{\mathbb{X}} \subset \mathbf{b}_{\mathbb{X}}$, $\mathbf{v}_{\mathbb{Y}} \subset \mathbf{b}_{\mathbb{Y}}$. The ME-objective $\Lambda(\mathbf{b}_{\mathbb{X}}, \mathbf{b}_{\mathbb{Y}}, \mathcal{V})$ is calculated as

$$\begin{aligned}
\Lambda(\mathbf{b}_{\mathbb{X}}, \mathbf{b}_{\mathbb{Y}}, \mathcal{V}) &= \frac{b_1 b_2 \overline{\mathbf{h}}(\mathbf{b}_{\mathbb{X}}, \mathbf{b}_{\mathbb{Y}}, \mathcal{V})^\top (\mathbf{S}_{b_1,b_2} + \gamma_{b_1,b_2}\mathbf{I})^{-1}\overline{\mathbf{h}}(\mathbf{b}_{\mathbb{X}}, \mathbf{b}_{\mathbb{Y}}, \mathcal{V})}{b_1+b_2}, \\
\overline{\mathbf{h}}(\mathbf{b}_{\mathbb{X}}, \mathbf{b}_{\mathbb{Y}}, \mathcal{V}) &= \left(\frac{1}{b_1}\sum_{i=1}^{b_1} k(\mathbf{b}_{\mathbb{X}}, \mathbf{v}_j) - \frac{1}{b_2}\sum_{j=1}^{b_2} k(\mathbf{b}_{\mathbb{Y}}, \mathbf{v}_j)\right)_{j=1}^J \in \mathbb{R}^J \\
\mathbf{v}_j \in \mathcal{V} &= \left\{\mathbf{v}_{\mathbb{X}}; \mathbf{v}_{\mathbb{Y}}\right\}.
\end{aligned} \tag{3}$$

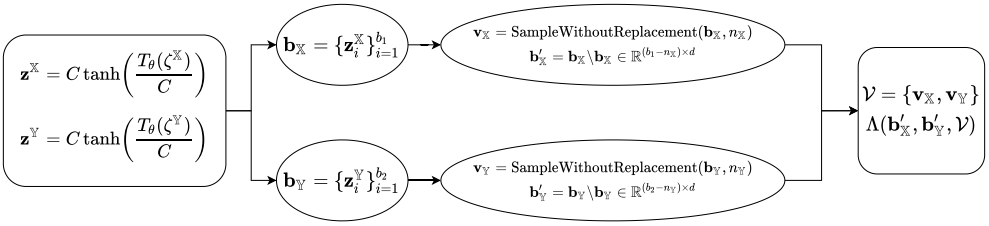


Fig. 2. Illustration of how the ME-objective is calculated. First one builds batches for each label, and then samples fixed prototypes from each batch which is concatenated into prototypes $\mathcal{V}$, which enables computation of Λ.

The overall idea is to use T_θ to map $\zeta^{\mathbb{X}}$ and $\zeta^{\mathbb{Y}}$ away from each other such that $\mathbf{z}^{\mathbb{X}}$ and $\mathbf{z}^{\mathbb{Y}}$ are separated in latent space as shown in Fig. 2.

Training with the ME-objective. We calculate the ME-test statistic on batches of latent representations with randomly subsampled prototypes for each label and train T_θ to "pull" latent representations away from each other by maximizing the ME-test statistic as an objective. By maximizing $\Lambda(\mathbf{b}_{\mathbb{X}}, \mathbf{b}_{\mathbb{Y}}, \mathcal{V})$ with respect to $\mathbf{b}_{\mathbb{X}}, \mathbf{b}_{\mathbb{Y}}$ we are maximizing the distributional difference between $\mathbf{z}^{\mathbb{X}}$ and $\mathbf{z}^{\mathbb{Y}}$ resulting in T_θ mapping apart $\zeta^{\mathbb{X}}$ and $\zeta^{\mathbb{Y}}$. T_θ is updated by first calculating $\Lambda(\mathbf{b}_{\mathbb{X}}, \mathbf{b}_{\mathbb{Y}}, \mathcal{V})$ using subsampled prototypes $\mathbf{v}_{\mathbb{X}}, \mathbf{v}_{\mathbb{Y}}$ from $\mathbf{b}_{\mathbb{X}}, \mathbf{b}_{\mathbb{Y}}$ and then taking the gradient of $\Lambda(\mathbf{b}_{\mathbb{X}}, \mathbf{b}_{\mathbb{Y}}, \mathcal{V})$ with respect to θ, illustrated in Fig. 2.

The role of subsampling prototypes. While it is possible to use the mean of $\mathbf{b}_{\mathbb{X}}, \mathbf{b}_{\mathbb{Y}}$ as prototypes for each class, we found through early runs that this often led to a numerical collapse of the ME-objective. Randomly sampling the prototypes instead can be interpreted as a type of jittering, which helped prevent a numerical collapse.

Combining the ME-objective with a generative loss. We train T_θ on all objectives related to the generative model, denoted $\mathcal{L}_G$, to ensure that $\mathbf{z}^{\mathbb{X}}$ and $\mathbf{z}^{\mathbb{Y}}$ are separated under the condition that they still retain a meaningful representation to the generator G. The loss for T_θ is then

$$\mathcal{L}_{T_\theta}(\mathbf{b}_{\mathbb{X}}, \mathbf{b}_{\mathbb{Y}}, \mathcal{X}) = \mathcal{L}_G(\mathcal{X}) + \eta_\Lambda \Lambda(\mathbf{b}'_{\mathbb{X}}, \mathbf{b}'_{\mathbb{Y}}, \mathcal{V}) \tag{4}$$

where $\mathcal{X}$ are any input required of the generative model objective and $\eta_\Lambda < 0$ is a hyperparameter controlling the effect of the ME-objective. In a VAE context, T_θ can be the encoder or a mapping between the encoder and decoder.

To prevent unbounded $\Lambda(\mathbf{b}_{\mathbb{X}}, \mathbf{b}_{\mathbb{Y}}, \mathcal{V})$, the domain of $\mathbf{z}$ is bounded to $[-C, C]^d, C \in \mathbb{R}$ by applying a $C\tanh(\frac{\cdot}{C})$ transform to $T_\theta(\zeta)$.

3.3.1. Choice of generative model: IntroVAE

We apply the ME-objective to the VAE variation IntroVAE [15], which is a VAE with an additional modified GAN objective. IntroVAE has been demonstrated to offer high-quality image synthesis on larger images coupled with a minimalistic VAE architecture. This was favored over [16,23] which either offered too complicated setups or convoluted latent representations. IntroVAE extends the training procedure of VAEs by training the encoder and generator on

$$\begin{aligned} \mathcal{L}_E(\mathbf{x}, \mathbf{z}) &= \text{ELBO}(\boldsymbol{\theta}, \boldsymbol{\phi}; \mathbf{x}) + [m - D_{KL}\left(q_{\boldsymbol{\phi}}(\mathbf{z} \mid \mathbf{x}) \| p(\mathbf{z})\right)]^+ \\ \mathcal{L}_G(\mathbf{x}, \mathbf{z}) &= D_{KL}\left(q_{\boldsymbol{\phi}}(\mathbf{z} \mid \mathbf{x}) \| p(\mathbf{z})\right) + \mathbb{E}_{q_{\boldsymbol{\phi}}(\mathbf{z}|\mathbf{x})}\left[\log p_{\boldsymbol{\theta}}(\mathbf{x} \mid \mathbf{z})\right] \end{aligned} \tag{5}$$

where $[m-x]^+ = \text{ReLU}(0, m-x)$ serves as a saddle point objective for the encoder p_θ and decoder $q_{\boldsymbol{\phi}}$ using the KL-divergence between approximative posterior and prior of $\mathbf{z}$ $D_{KL}\left(q_{\boldsymbol{\phi}}(\mathbf{z} \mid \mathbf{x}) \| p(\mathbf{z})\right)$. DIF is then applied by adding the ME-objective to the encoder loss:

$$\mathcal{L}(\mathbf{x}, \mathbf{b}_{\mathbb{X}}, \mathbf{b}_{\mathbb{Y}}) = \mathcal{L}_E(\mathbf{x}, \mathbf{z}) + \eta_\Lambda \Lambda(\mathbf{b}'_{\mathbb{X}}, \mathbf{b}'_{\mathbb{Y}}, \mathcal{V}). \tag{6}$$

It should be noted that we take T_θ to be the encoder and consequently ζ's are taken to be images. When training DIF, we use both losses specified in Eqs. (4) and (6). We give a holistic overview of the training procedure in Fig. 3.

3.4. Isolating features

To isolate features, we take our binary classifier $f(\mathbf{z})$ to be a L^1 regularized logistic regression classifier. We can then exploit the shrinkage properties of the L^1 regularization to isolate the most important features. We then select features by their magnitude $|w_i|$, where w_i is the i:th element in the linear weights $\mathbf{w} = [w_1, \ldots, w_n]$.

We linearly separate the latent representations by choosing a linear kernel $k(\mathbf{x}, \mathbf{y}) = \nu \mathbf{x}^\top \mathbf{y}$ with hyperparameter $\nu > 0$ for our ME-objective. With $\mathbf{z}$ bounded by a $\tanh(\cdot)$ transform, it follows that $\Lambda(\mathbf{b}_{\mathbb{X}}, \mathbf{b}_{\mathbb{Y}}, \mathcal{V})$ with a linear kernel also is bounded [22].

3.5. Prototype generation

As previously mentioned, prototypes can be seen as a typical example of an object belonging to a class. Prototypes can both be used for training and evaluation of latent representations, where we so far have only discussed training.

From an evaluation perspective, one can use the latent representations of the trained model and then optimize the ME-test with respect to randomly initialized prototypes. These optimized prototypes can then be visualized using the generative model. One could then qualitatively evaluate the prototypes by examining if they semantically represent the class, i.e. if the prototypes distinctly look like dogs or cats.

DIF training procedure with IntroVAE

$$\mathrm{KL}_E^{\text{real}} = D_{KL}\left(q_\phi\left(\mathbf{z} \mid \mathbf{x}\right) \| p(\mathbf{z})\right)$$
$$\mathrm{KL}_E^{\text{rec}} = \left[m - D_{KL}\left(q_\phi\left(\mathbf{z} \mid \mathbf{x}_r\right) \| p(\mathbf{z})\right)\right]^+$$
$$\mathrm{KL}_E^{\text{fake}} = \left[m - D_{KL}\left(q_\phi\left(\mathbf{z} \mid \mathbf{x}_f\right) \| p(\mathbf{z})\right)\right]^+$$
$$\mathrm{KL}_G^{\text{rec}} = D_{KL}\left(q_\phi\left(\mathbf{z} \mid \mathbf{x}_r\right) \| p(\mathbf{z})\right)$$
$$\mathrm{KL}_G^{\text{fake}} = D_{KL}\left(q_\phi\left(\mathbf{z} \mid \mathbf{x}_f\right) \| p(\mathbf{z})\right)$$

Data sample	$\mathbf{x}$	$\mathbb{R}^{n\times n\times c}$
Fake sample	$\mathbf{x}_f$	$\mathbb{R}^{n\times n\times c}$
Recon. sample	$\mathbf{x}_r$	$\mathbb{R}^{n\times n\times c}$
Latent repr.	$\mathbf{z}$	$\mathbb{R}^d$
Mean of q_ϕ	μ	$\mathbb{R}$
Std. of q_ϕ	σ	$\mathbb{R}_{\geq 0}$

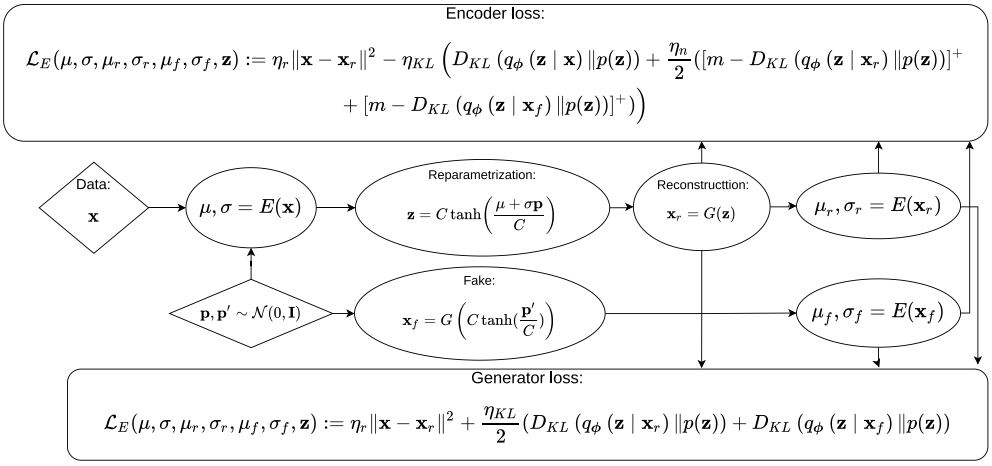


Fig. 3. This figure illustrates how data $\mathbf{x}$ flows through our generative model (IntroVAE) and which loss is used for the encoder and generator respectively. IntroVAE has the additional aspect of adversarial learning, where a fake sample is additionally generated and passed into the loss of both the encoder and generator.

Algorithm 1 Procedure for finding prototypes $\mathcal{V}$

Require: Training set $\{\mathbf{z}_i^{\mathbb{X}}\}_{i=1}^{n_1}, \{\mathbf{z}_j^{\mathbb{Y}}\}_{j=1}^{n_2}$ and test set $\{\mathbf{z'}_i^{\mathbb{X}}\}_{i=1}^{n_1^{\text{te}}}, \{\mathbf{z'}_j^{\mathbb{Y}}\}_{j=1}^{n_2^{\text{te}}}$. # of prototypes J. $\mathcal{N}(\mu_{\mathbb{X}}, \Sigma_{\mathbb{X}}), \mathcal{N}(\mu_{\mathbb{Y}}, \Sigma_{\mathbb{Y}})$ fitted on $\{\mathbf{z}_i^{\mathbb{X}}\}_{i=1}^{n_1}, \{\mathbf{z}_j^{\mathbb{Y}}\}_{j=1}^{n_2}$ respectively

$\mathcal{V} \leftarrow [\mathbf{v}_{\mathbb{X}}; \mathbf{v}_{\mathbb{Y}}]$ where $\mathbf{v}_{\mathbb{X}} \sim \mathcal{N}(\mu_{\mathbb{X}}, \Sigma_{\mathbb{X}}), \mathbf{v}_{\mathbb{Y}} \sim \mathcal{N}(\mu_{\mathbb{Y}}, \Sigma_{\mathbb{Y}})$
$l \leftarrow$ MedianHeuristic($\{\mathbf{z}_i^{\mathbb{X}}\}_{i=1}^{n_1}, \{\mathbf{z}_j^{\mathbb{Y}}\}_{j=1}^{n_2}$) ▷ Initialize kernel parameter with median heuristic [24]
while not converged **do**
  for p in $\{l, \mathcal{V}\}$ **do** ▷ Alternate between l and $\mathcal{V}$
    for iterations **do**
      Calculate $\hat{\lambda}_{n_1,n_2}$
      $p = p + \epsilon\frac{\partial\hat{\lambda}_{n_1,n_2}}{\partial p}$ ▷ Update prototypes, with learning rate ϵ
    end for
  end for
end while
Calculate ME-statistic on test data $\hat{\lambda}^{\text{te}}_{n_1,n_2}$
Calculate p-value $p = \chi_J(\hat{\lambda}^{\text{te}}_{n_1,n_2})$
return Prototypes $\mathcal{V}$, p-value p

Procedure. We find the prototypes according to Algorithm 1 (see [24]). Intuitively, the algorithm can be understood as using a training set of latent representations $\{\mathbf{z}_i^{\mathbb{X}}\}_{i=1}^{n_1}, \{\mathbf{z}_j^{\mathbb{Y}}\}_{j=1}^{n_2}$ to first find J plausible prototypes, where J is chosen. We then proceed to test the validity of these prototypes by calculating the ME-test statistic on a test set of latent representations $\{\mathbf{z'}_i^{\mathbb{X}}\}_{i=1}^{n_1^{\text{te}}}, \{\mathbf{z'}_j^{\mathbb{Y}}\}_{j=1}^{n_2^{\text{te}}}$. Once the prototypes are validated by the ME-test on the test set, we map them back to image space using our generative model $\mathbf{x}_{\text{prototypes}} = G(\mathcal{V})$.

Table 1
All classification results in test data AUC, averaged over 5 runs. Sparser models do better with DIF representation, which implies that DIF indeed imposes a linearly separable latent space. DIF generally seems to improve image quality by the lower FID scores reported for fake images. The improved image generation quality is potentially caused by the structure imposed by DIF in latent space.

Dataset-model	Fake-FID (n = 32)	Prototype-FID (J = 16)	Log-likelihood	ELBO	$\lambda_{L^1} = 0$	$\lambda_{L^1} = 0.001$	$\lambda_{L^1} = 0.01$	$\lambda_{L^1} = 0.1$	$\lambda_{L^1} = 0.1$ %Non sparse	CNN
CelebHQ-DIF	**103.64 ± 4.177**	**125.704 ± 3.26**	**−2239.154 ± 0.593**	**−2422.622 ± 1.508**	**0.983 ± 0.002**	0.981 ± 0.001	0.98 ± 0.001	0.97 ± 0.004	9.4%±4.9%	0.967 ± 0.006
CelebHQ-Vanilla	107.184 ± 3.612	132.478 ± 2.53	−2301.921 ± 0.903	−2512.501 ± 3.796	0.948 ± 0.004	0.945 ± 0.002	0.92 ± 0.003	0.744 ± 0.015	2.7%±0.2%	0.967 ± 0.006
CelebHQ-logistic	114.514 ± 6.227	147.25 ± 3.166	−2299.671 ± 0.834	−2471.561 ± 1.473	0.967 ± 0.002	0.97 ± 0.001	0.94 ± 0.003	0.769 ± 0.004	2.9%±0.0%	0.967 ± 0.006
Fashion-DIF	**200.625 ± 4.125**	192.460 ± 7.5	−1192.343 ± 0.138	−1217.098 ± 0.264	0.822 ± 0.007	0.82 ± 0.002	**0.823 ± 0.003**	0.806 ± 0.001	3.6%±0.8%	0.775 ± 0.008
Fashion-Vanilla	227.644 ± 9.447	199.731 ± 10.552	**−1084.383 ± 0.156**	**−1107.932 ± 0.566**	0.763 ± 0.002	0.759 ± 0.002	0.737 ± 0.003	0.581 ± 0.002	0.9%±0.1%	0.775 ± 0.008
Fashion-logistic	209.541 ± 7.145	**189.057 ± 2.31**	−1179.895 ± 0.155	−1202.83 ± 0.439	0.791 ± 0.003	0.801 ± 0.002	0.756 ± 0.002	0.56 ± 0.009	1.4%±0.0%	0.775 ± 0.008
MNIST-DIF	**87.112 ± 2.535**	**101.479 ± 3.926**	−71.487 ± 0.03	−74.486 ± 0.047	0.992 ± 0.001	0.992 ± 0.001	0.989 ± 0.002	0.939 ± 0.001	29.2%±3.6%	**0.994 ± 0.007**
MNIST-Vanilla	92.574 ± 0.89	103.547 ± 1.463	−72.077 ± 0.038	−75.936 ± 0.078	0.989 ± 0.002	0.988 ± 0.002	0.989 ± 0.002	0.917 ± 0.003	37.5%±0.0%	**0.994 ± 0.007**
MNIST-logistic	88.894 ± 8.272	110.101 ± 1.236	**−64.5 ± 0.01**	**−69.809 ± 0.068**	0.989 ± 0.001	0.987 ± 0.003	0.989 ± 0.002	0.909 ± 0.002	43.8%±0.0%	**0.994 ± 0.007**
Covid-DIF	**214.266 ± 1.426**	**227.86 ± 1.462**	−929.991 ± 1.596	−1122.549 ± 10.712	**0.995 ± 0.005**	0.989 ± 0.003	0.979 ± 0.01	0.952 ± 0.01	10.0%±12.9%	0.983 ± 0.009
Covid-Vanilla	253.288 ± 2.564	268.854 ± 4.008	−761.591 ± 0.151	−891.08 ± 2.046	0.977 ± 0.004	0.968 ± 0.005	0.968 ± 0.021	0.888 ± 0.025	2.3%±0.2%	0.983 ± 0.009
Covid-logistic	247.358 ± 7.338	229.389 ± 5.821	**−652.715 ± 0.131**	**−759.155 ± 2.906**	0.874 ± 0.027	0.852 ± 0.012	0.916 ± 0.013	0.937 ± 0.0	1.1%±0.2%	0.983 ± 0.009

4. Experiments

4.1. Data

We conduct our experiments on CelebHQ (separating females with 18 943 samples and males with 11 057 samples, 256 × 256 × 3), COVID-19[1] chest X-rays (separating healthy with 1602 samples and COVID-19 infected with 434 samples, 256 × 256 × 1) and MNIST (separating 3s with 7141 samples and 8s with 6825 samples, upsampled to 64 × 64 × 1).

All datasets are divided into training (90%) and testing (10%) sets. We use the same architecture for all generative models and their respective components.

4.2. Evaluation

We evaluate our experiments in the following way on test data:

1. **Generative model performance**: We estimate the log-likelihood and calculate the ELBO on test data together with FID scores [25] for fake samples.
2. **Finding Prototypes**: We look at generated prototypes and measure them quantitatively with p-value and FID score.
3. **Latent traversals**: We traverse from an image in P to an image in Q and judge if the transition occurs smoothly without artifacts. This evaluation is done to ensure the generative model does not overfit.
4. **Predictive performance** We compare our L^1 regularized classifiers to a CNN classifier (using the identical architecture as the encoder) fitted directly on images and measure performance in Area Under the Curve (AUC).
5. **Isolating features**: We judge the extracted features qualitatively on how distinct the features are and how the latent transitions look. We analyze the 3 largest features w_i by magnitude $|w_i|$ for a fix λ_{L^1}. Strong classification performance for a large λ_{L^1} indicates the validity of the isolated features. We also report the percentage of non sparse features, which is calculated as $\frac{\sum_i^d \mathbf{1}_{w_i \geq 0.1 \max |w|}}{d}$.

4.3. Benchmarks

Our benchmarks include vanilla IntroVAE, as well as a method inspired by [26], detailed below.

A logistic separation objective. As an ablation study to demonstrate the importance of using an ME-based objective, we also consider an objective akin to logistic regression applied to the latent space. Namely, instead of using ME, we take Λ to be a logistic regression objective (i.e. binary cross-entropy loss) $\mathcal{L}(w^\top C \tanh\left(\frac{T_\theta(\zeta)}{C}\right), y_{P/Q})$ for a linear decision boundary $w^\top C \tanh\left(\frac{T_\theta(\zeta)}{C}\right)$, where we keep the weight vector fixed $w = \mathbf{1}_{1\times d}$ but optimize the encoder parameters θ. Here, $y_{P/Q} \in \{0, 1\}$ is the class indicator for ζ. We want T_θ to map $\{\zeta_i^{\mathbb{X}}\}_{i=1}^{n_1}$ and $\{\zeta_j^{\mathbb{Y}}\}_{j=1}^{n_2}$ to their respective sides of a constant hyperplane w and thus minimize this objective. In contrast to the ME-objective, this method imposes the desired structure directly through a fixed hyperplane rather than "pulling" latent representations apart through a kernel.

5. Results

We present the quantitative comparisons in Table 1. As classification performance tends to drop when $\lambda_{L^1} = 0.1$ for vanilla IntroVAE, we investigate sparsity levels at this value.

Prototypes. We display prototypes for all datasets in Fig. 4. The prototypes are classified into P and Q using the same linear classifier trained for feature isolation. We find that DIF seems to generate prototypes with more visually distinctive features. Further, DIF achieves better FID scores in Table 1.

[1] **Disclaimer**: The results presented for the COVID-19 x-ray data are for illustration purposes and further work is needed to draw any medically relevant conclusions based on them.

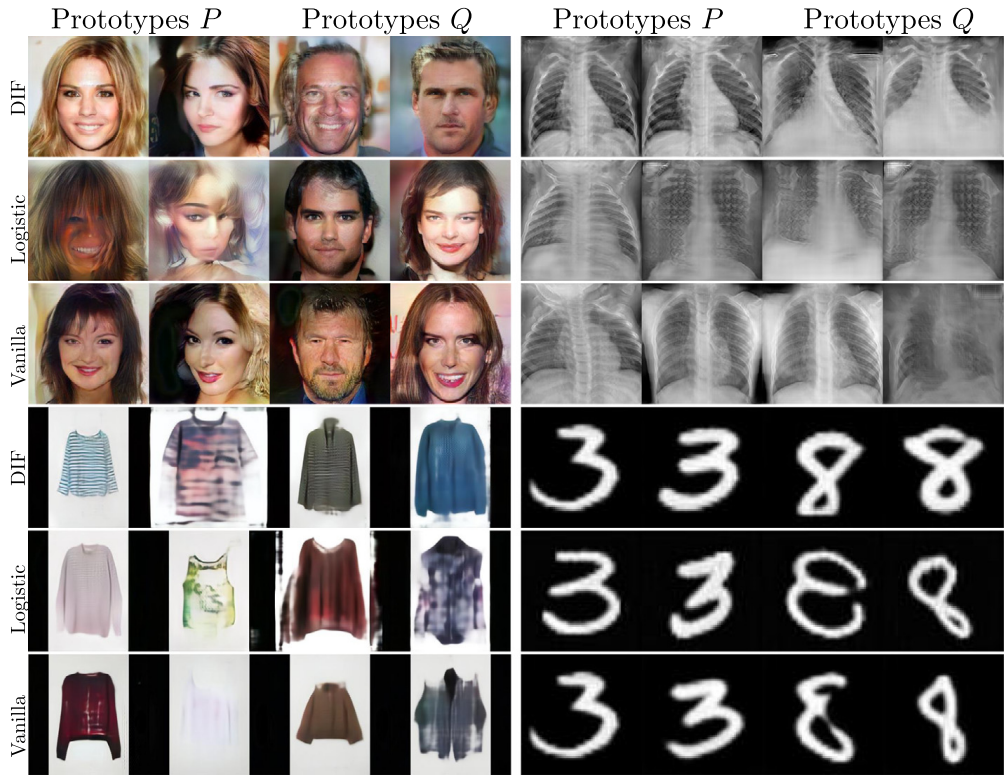


Fig. 4. Prototypes for CelebHQ, fashion dataset, COVID-19 and MNIST. P denotes female, low price, healthy and 3's, and Q denotes male, high price, COVID-19 infected and 8's respectively.

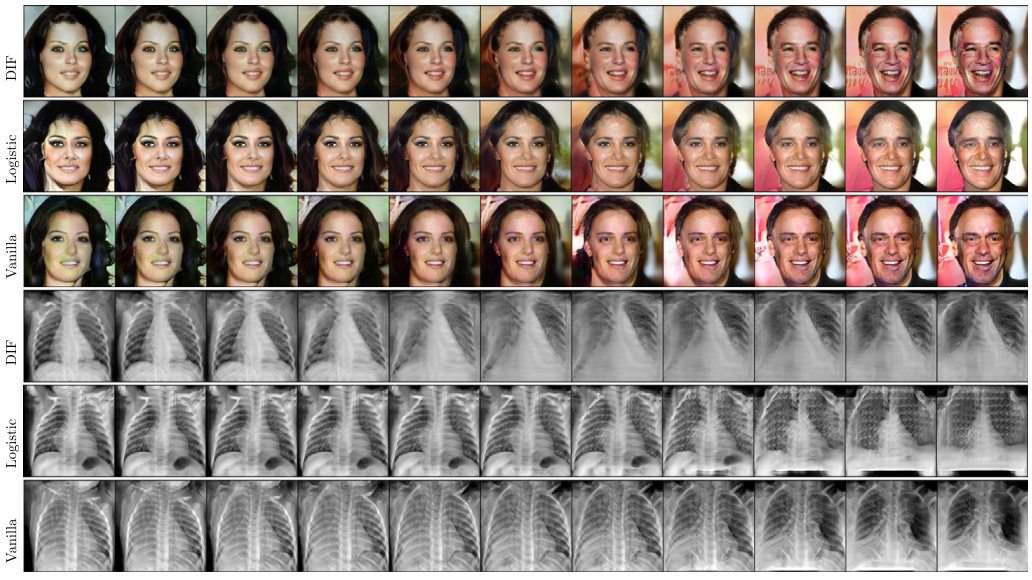


Fig. 5. Image traversals for samples from P to samples from Q.

Image traversals. Smooth artifact-free image traversals between latent representations in Fig. 5 indicate that no model is overfitting.

Isolating features. We present isolated features using L^1 regularized logistic classification in Fig. 6. DIF finds features that control facial hair, the width of numbers respectively, and non-organ matter (non-heart gray matter) in the lungs. These changes occur in an isolated fashion compared to the other methods where one element seemingly affect multiple visual features. In particular, we find that the non-organ matter is consistent with medical results reported in [27,28]. We present the test accuracy of a L^1-regularized classification trained on the latent representations for different regularization values $\lambda_{L^1} \in \{0.0, 0.001, 0.01, 0.1\}$ in Table 1. We find that lasso regression surprisingly outperforms a CNN for CelebHQ and the COVID-19 dataset for both DIF and the logistic benchmark, where DIF yields the most gains in classification performance. Models trained with DIF also retain performance for sparser classifiers.

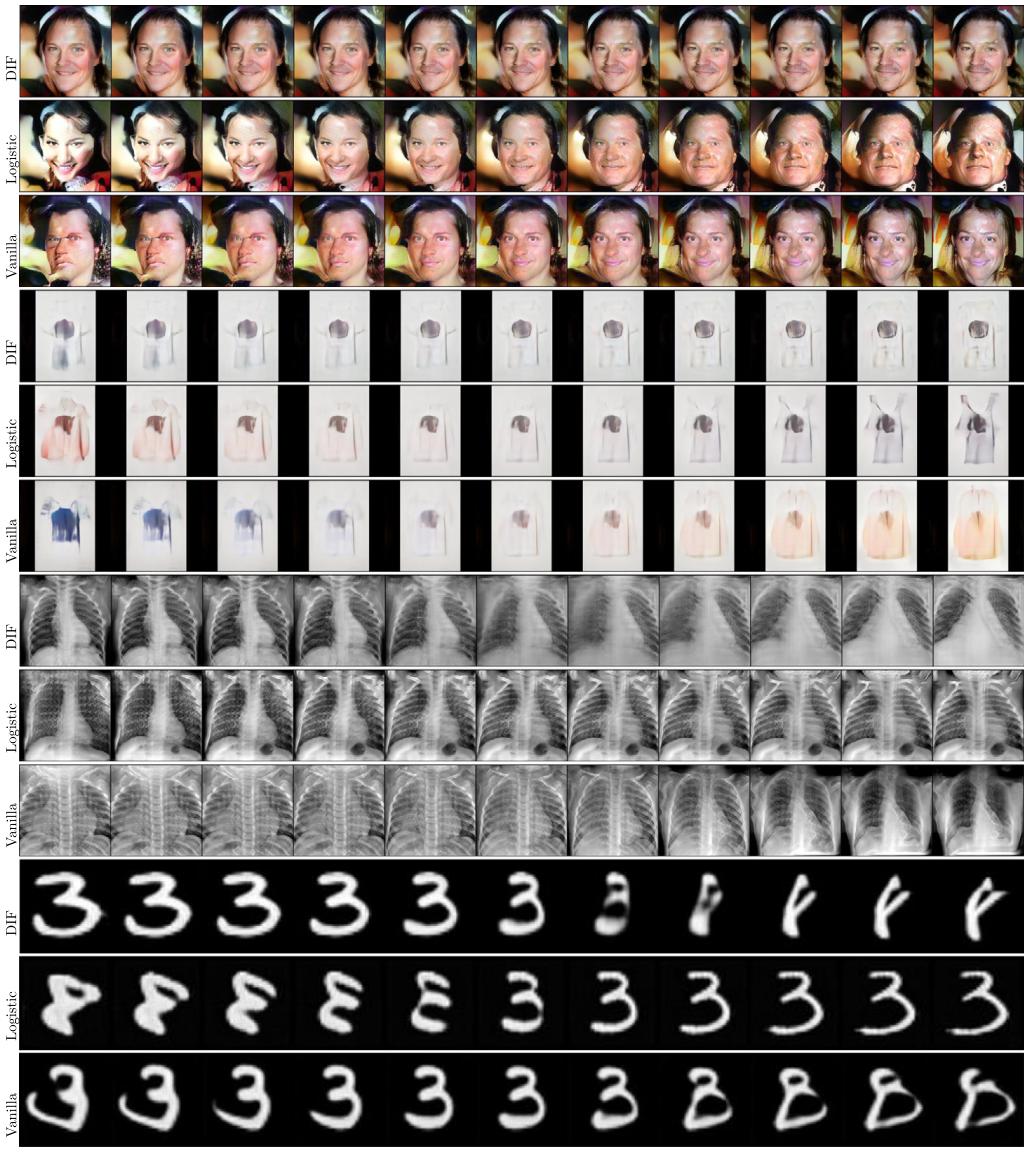


Fig. 6. Comparison of isolated features for all four data sets. We traverse each feature between (-C, C), the interval of our tanh transform.

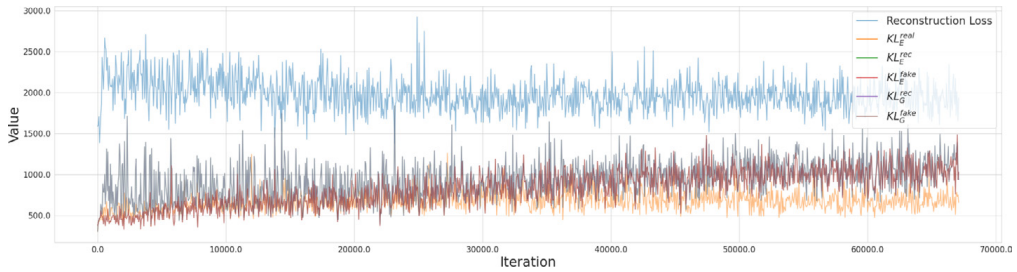


Fig. 7. Training IntroVAE with DIF for 100 epochs on the CelebHQ dataset.

As the percentage of non-sparse features are higher for DIF, this suggests that DIF finds more meaningful and robust features that discriminate between P and Q.

Training stability of IntroVAE. We noted during the experiments that IntroVAE was unstable during training. While [29] has proposed replacing the hinge loss in IntroVAE with an exponentiated ELBO loss for better stability, we found that pre-training IntroVAE on the VAE objective also improved stability when for later training on the hinge loss. We train DIF on the CelebHQ dataset for 100 epochs and plot the reconstruction loss together with the KL-losses for the encoder and decoder in Fig. 7.

For further results on training stability on different datasets, we refer to Appendix.

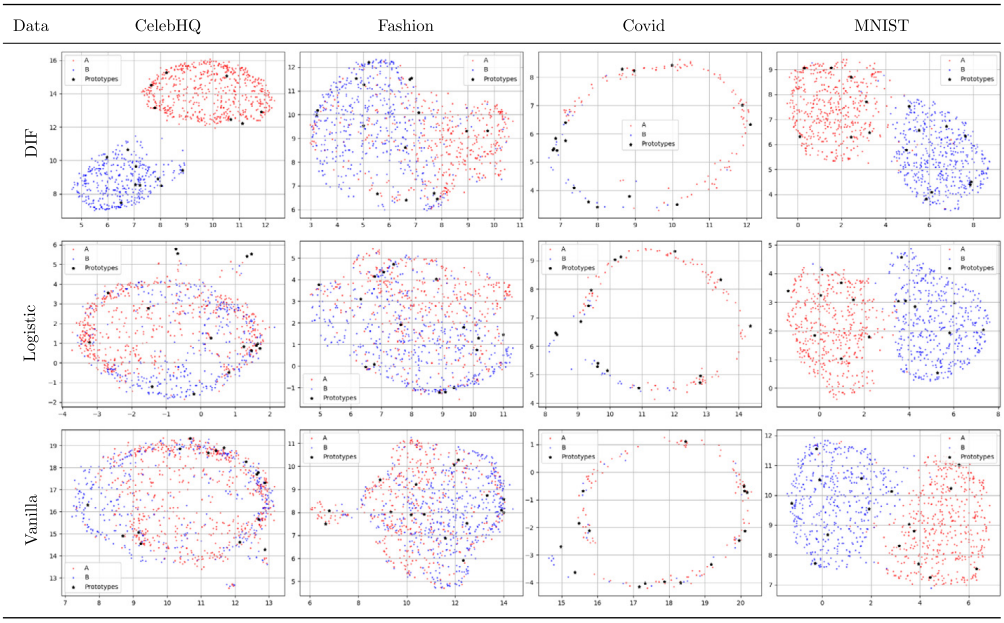


Fig. 8. UMAP visualization $\mathbf{z}$ for test data. Red points belong to P, blue points to Q and black points denote prototypes. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

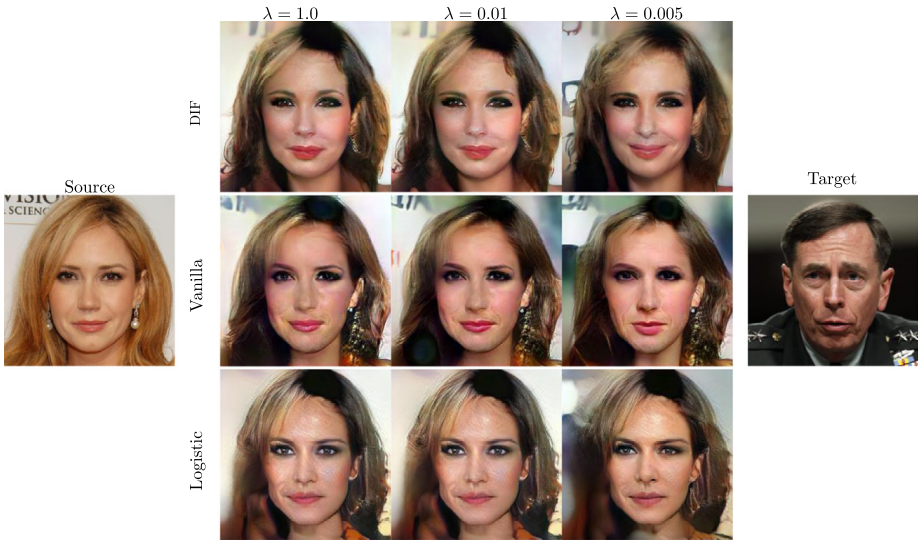


Fig. 9. λ denotes the strength of the adversarial attack. A smaller lambda implies a stronger attack. Both DIF and the Logistic benchmarks exhibit robustness towards adversarial attacks, while the Vanilla version starts to exhibit masculine features at $\lambda = 0.005$.

6. Analysis

Does DIF achieve its aim of separation in latent space?. We see that DIF achieves a very good separation on the CelebHQ dataset in Fig. 8. On MNIST, a separation objective is not needed as the latent representations are naturally separated by looking at the latent space of the Vanilla model.

Is DIF more robust to adversarial attacks?. As DIF correctly imposes a linearly separable structure on the latent space, we hypothesize this improves resilience towards adversarial attacks [30,31] and report this in Fig. 9. We use the adversarial attacks proposed in [13] to test the robustness.

What is the training behavior of DIF?. To gain a full understanding of how DIF impacts the latent representation we investigate how the latent space for test data changes with training epochs on the CelebHQ dataset in Fig. 10. For more results, we refer to Appendix.

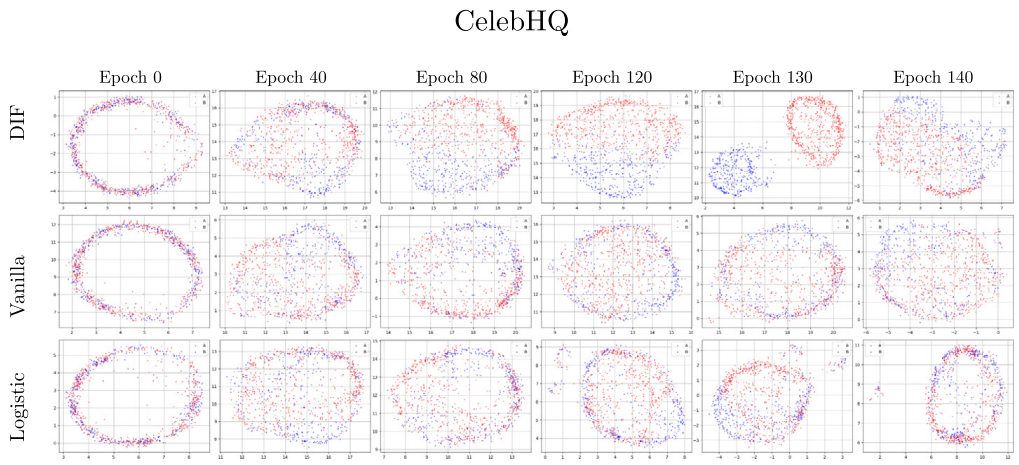


Fig. 10. Evolution of latent space **z** for test data of CelebHQ during training. The change in latent space over training iterations suggests that the ME-objective indeed causes a separation in latent space during training.

7. Conclusion

We presented a novel framework to construct interpretable feature representations applicable to image data with binary labels. The framework combines deep generative modeling with a supervised objective based on kernel-based two-sample testing (ME objective) to encourage the separation between the classes in the latent space. As a result, individual latent dimensions are more interpretable in relation to the classification problem at hand and capture class-discerning image features. Latent traversals for DIF between images of different classes exhibit smooth transitions, indicating a smooth latent space. We note that a linear classifier can then be applied to the resulting latent representation, in some cases resulting in the performance superior to CNNs on previously unseen data. Further, imposing a linear structure on the latent space improves robustness against adversarial attacks. The framework allows for constructing realistic class prototypes, as well as isolating the most discriminative features by simply applying L^1-regularization to the linear classifier on the resulting latent representation. A natural extension would be to extend DIF to multi-label classification, possibly considering a causal setting.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Appendix A. Supplementary data

Supplementary material related to this article can be found online at https://doi.org/10.1016/j.jcmds.2022.100067.

References

[1] Gilpin LH, Bau D, Yuan BZ, Bajwa A, Specter MA, Kagal L. Explaining explanations: An overview of interpretability of machine learning. 2018.
[2] Rudin C. Stop explaining black box machine learning models for high stakes decisions and use interpretable models instead. Nat Mach Intell 2019;1:206–15.
[3] LeCun Y, Bengio Y. Convolutional networks for images, speech, and time series. In: The handbook of brain theory and neural networks. Cambridge, MA, USA: MIT Press; 1998, p. 255–8.
[4] Johansson U, Sönströd C, Norinder U, Boström H. Trade-off between accuracy and interpretability for predictive in silico modeling. Future Med Chem 2011;3:647–63. http://dx.doi.org/10.4155/fmc.11.23.
[5] Gosiewska A, Kozak A, Biecek P. Simpler is better: Lifting interpretability-performance trade-off via automated feature engineering. Decis Support Syst 2021;150:113556. http://dx.doi.org/10.1016/j.dss.2021.113556, URL https://www.sciencedirect.com/science/article/pii/S016792362100066X, Interpretable Data Science For Decision Making.
[6] Ribeiro MT, Singh S, Guestrin C. Why should I trust you?. In: Proceedings of the 22nd ACM SIGKDD international conference on knowledge discovery and data mining - KDD '16. New York, New York, USA: ACM Press; 2016, p. 1135–44. http://dx.doi.org/10.1145/2939672.2939778, URL http://dl.acm.org/citation.cfm?doid=2939672.2939778.
[7] Lundberg SM, Lee S-I. A unified approach to interpreting model predictions. 2017, p. 4768–77.
[8] Chen C, Li O, Tao D, Barnett A, Rudin C, Su JK. This looks like that: Deep learning for interpretable image recognition. 2019, URL https://proceedings.neurips.cc/paper/2019/file/adf7ee2dcf142b0e11888e72b43fcb75-Paper.pdf.
[9] Hase P, Chen C, Li O, Rudin C. Interpretable image recognition with hierarchical prototypes. 2019.
[10] Chen T, Kornblith S, Norouzi M, Hinton G. A simple framework for contrastive learning of visual representations. In: Proceedings of the 37th international conference on machine learning. ICML '20, JMLR.org; 2020.
[11] Joy T, Schmon SM, Torr PHS, Siddharth N, Rainforth T. Capturing label characteristics in VAEs. In: ICLR. 2021.
[12] Locatello F, Bauer S, Lucic M, Raetsch G, Gelly S, Schölkopf B, Bachem O. Challenging common assumptions in the unsupervised learning of disentangled representations. In: Chaudhuri K, Salakhutdinov R, editors. Proceedings of the 36th international conference on machine learning. Proceedings of machine learning research, 97, PMLR; 2019, p. 4114–24, URL https://proceedings.mlr.press/v97/locatello19a.html.

[13] Gondim-Ribeiro G, Tabacof P, Valle E. Adversarial attacks on variational autoencoders. 2018, arXiv:1806.04646.
[14] Kingma DP, Welling M. Auto-encoding variational Bayes. In: Bengio Y, LeCun Y, editors. ICLR. 2014, URL http://dblp.uni-trier.de/db/conf/iclr/iclr2014.html#KingmaW13.
[15] Huang H, Li Z, He R, Sun Z, Tan T. IntroVAE: Introspective variational autoencoders for photographic image synthesis. In: Proceedings of the 32nd international conference on neural information processing systems. NIPS '18, Red Hook, NY, USA: Curran Associates Inc.; 2018, p. 52–63.
[16] Pidhorskyi S, Adjeroh DA, Doretto G. Adversarial latent autoencoders. In: 2020 IEEE/CVF conference on computer vision and pattern recognition (CVPR). 2020, p. 14092–101.
[17] Higgins I, Matthey L, Pal A, Burgess CP, Glorot X, Botvinick MM, Mohamed S, Lerchner A. Beta-VAE: Learning basic visual concepts with a constrained variational framework. In: ICLR. 2017.
[18] Kim H, Mnih A. Disentangling by factorising. In: Dy J, Krause A, editors. Proceedings of the 35th international conference on machine learning. Proceedings of machine learning research, Vol. 80, PMLR; 2018, p. 2649–58, URL https://proceedings.mlr.press/v80/kim18b.html.
[19] Goodfellow I, Pouget-Abadie J, Mirza M, Xu B, Warde-Farley D, Ozair S, Courville A, Bengio Y. Generative adversarial nets. 2014, URL https://proceedings.neurips.cc/paper/2014/file/5ca3e9b122f61f8f06494c97b1afccf3-Paper.pdf.
[20] Lipton ZC, Tripathi S. Precise recovery of latent vectors from generative adversarial networks. 2017, arXiv:1702.04782.
[21] Rezende D, Mohamed S. Variational inference with normalizing flows. In: Bach F, Blei D, editors. Proceedings of the 32nd international conference on machine learning. Proceedings of machine learning research, Vol. 37, Lille, France: PMLR; 2015, p. 1530–8, URL https://proceedings.mlr.press/v37/rezende15.html.
[22] Jitkrittum W, Szabó Z, Chwialkowski K, Gretton A. Interpretable distribution features with maximum testing power. In: Proceedings of the 30th international conference on neural information processing systems. NIPS '16, Red Hook, NY, USA: Curran Associates Inc.; 2016, p. 181–9.
[23] Razavi A, van den Oord A, Vinyals O. Generating diverse high-fidelity images with VQ-VAE-2. In: Wallach H, Larochelle H, Beygelzimer A, d'Alché Buc F, Fox E, Garnett R, editors. Advances in neural information processing systems, Vol. 32. Curran Associates, Inc.; 2019, URL https://proceedings.neurips.cc/paper/2019/file/5f8e2fa1718d1bbcadf1cd9c7a54fb8c-Paper.pdf.
[24] Garreau D, Jitkrittum W, Kanagawa M. Large sample analysis of the median heuristic. 2017, arXiv:1707.07269.
[25] Barratt S, Sharma R. A note on the inception score. 2018, ArXiv abs/1801.01973.
[26] Lopez-Paz D, Oquab M. Revisiting classifier two-sample tests. 2017, URL https://openreview.net/forum?id=SJkXfE5xx.
[27] Cozzi D, Albanesi M, Cavigli E, Moroni C, Bindi A, Luvarà S, Lucarini S, Busoni S, Mazzoni LN, Miele V. Chest X-ray in new coronavirus disease 2019 (COVID-19) infection: findings and correlation with clinical outcome. Radiol Med 2020;125(8):730–7. http://dx.doi.org/10.1007/s11547-020-01232-9.
[28] Kong W, Agarwal P. Chest imaging appearance of COVID-19 infection. Radiol: Cardiothorac Imaging 2020;2(1):e200028. http://dx.doi.org/10.1148/ryct.2020200028, arXiv:https://doi.org/10.1148/ryct.2020200028.
[29] Daniel T, Tamar A. Soft-introvae: Analyzing and improving the introspective variational autoencoder. In: Proceedings of the IEEE/CVF conference on computer vision and pattern recognition (CVPR). 2021, p. 4391–400.
[30] Willetts MJ, Camuto A, Rainforth T, Roberts S, Holmes CC. Improving {vae}s' robustness to adversarial attack. 2021, URL https://openreview.net/forum?id=-Hs_otp2RB.
[31] Camuto A, Willetts M, Roberts SJ, Holmes CC, Rainforth T. Towards a theoretical understanding of the robustness of variational autoencoders. In: AISTATS. 2021.

Supplementary Material for Towards Deep Interpretable Features

1. Log-likelihood estimate

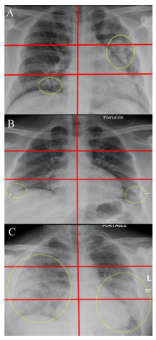
Given the test dataset $\mathcal{D}_{\text{test}} = \{\mathbf{x}_i\}_{i=1}^{m}$, we estimate the log-likelihood for IntroVAE as

$$\frac{1}{m}\sum_{i=1}^{m}\log\left(\frac{1}{S}\sum_{s=1}^{S}\frac{p_\theta\left(\mathbf{z}_i^{(s)} \mid \mathbf{x}_i\right)p(\mathbf{z}_i^{(s)})}{q_\phi\left(\mathbf{z}_i^{(s)} \mid \mathbf{x}\right)}\right), \quad \text{where } \mathbf{z}_i^{(s)} \sim q_\phi\left(\cdot \mid \mathbf{x}_i\right) \tag{1}$$

This is an importance sampling based estimator where S is the number of samples. We choose $S = 50$ for our experiments.

2. COVID-19 medical papers [1, 2, 3] summary

We summarize the findings in [1, 2, 3]. The papers observe that COVID-19 infected lungs exhibits different types of opacities in chest x-rays. These opacities are consistent with the isolated features and prototypes that DIF finds.

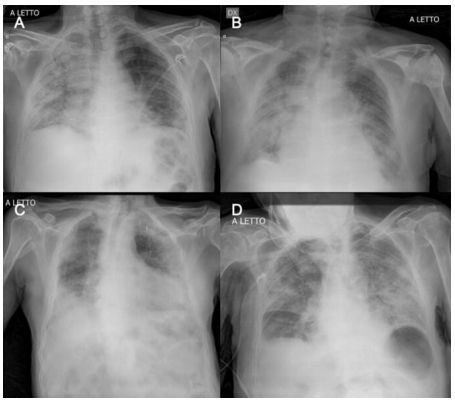


(a) "Examples of the chest severity score. A, Chest radiograph of a 26-year-old male with no past medical history other than obesity (BMI = 38) who was admitted for COVID-19 requiring oxygen supplementation via nasal cannula and initially tested negative for COVID-19 via nasopharyngeal swab, but later tested positive for antibodies to SARS-CoV2 virus. Portable CXR shows right lower lung zone, left middle lung zone, and left upper lung zone hazy opacities; total score=3. B, Chest radiograph of a 23-year-old male with no past medical history who tested positive for COVID-19 via RT-PCR and was subsequently discharged from the emergency department with home care and isolation precautions. Portable CXR shows right and left peripheral lower lung zone hazy opacities; total score=2. C, Chest radiograph in a 32-year-old overweight (BMI=30) COVID-19 positive male with a history of childhood asthma who was subsequently admitted and intubated in the ICU for 3 days. Portable CXR shows opacities in all three right lung zones and in the left middle and lower lung zones; total score=5." - [1, Figure 2]

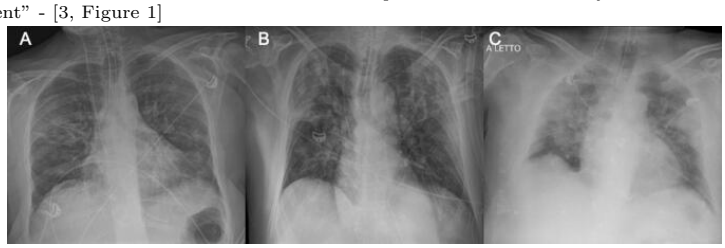


(b) "Chest radiograph in a patient with COVID-19 infection demonstrates right infrahilar airspace opacities." - [2, Figure 1]

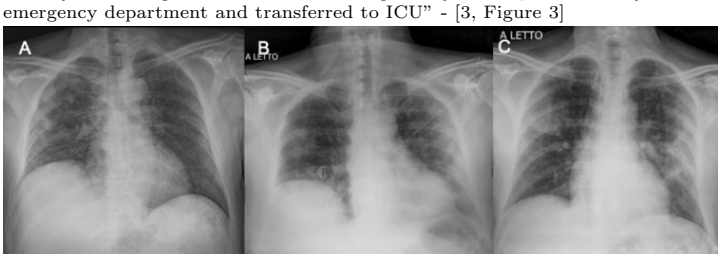
Figure 1: X-ray images of COVID-19 diagnosed lungs in [1, 2].



(a) "Diffuse lung involvement in elderly patients. Four cases of advanced lung disease with diffuse consolidations and interstitial involvement of patients older than 80 years at the emergency department" - [3, Figure 1]



(b) "Chest X-ray in three patients with severe respiratory failure, immediately intubated at the arrival in emergency department and transferred to ICU" - [3, Figure 3]



(c) "Peripheral consolidation in COVID-19 pneumonia. Images in A, B and C show three cases of male patients with subpleural consolidations and bilateral involvement" - [3, Figure 4]

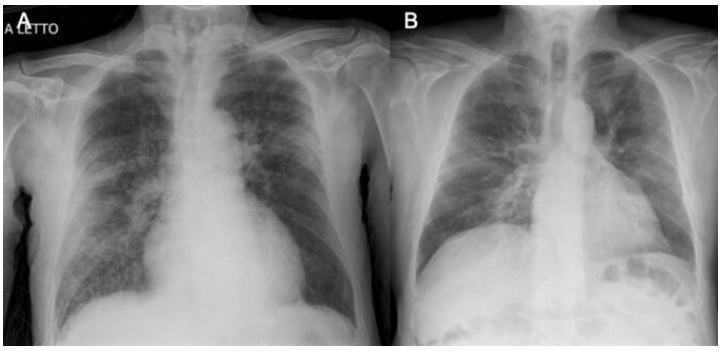
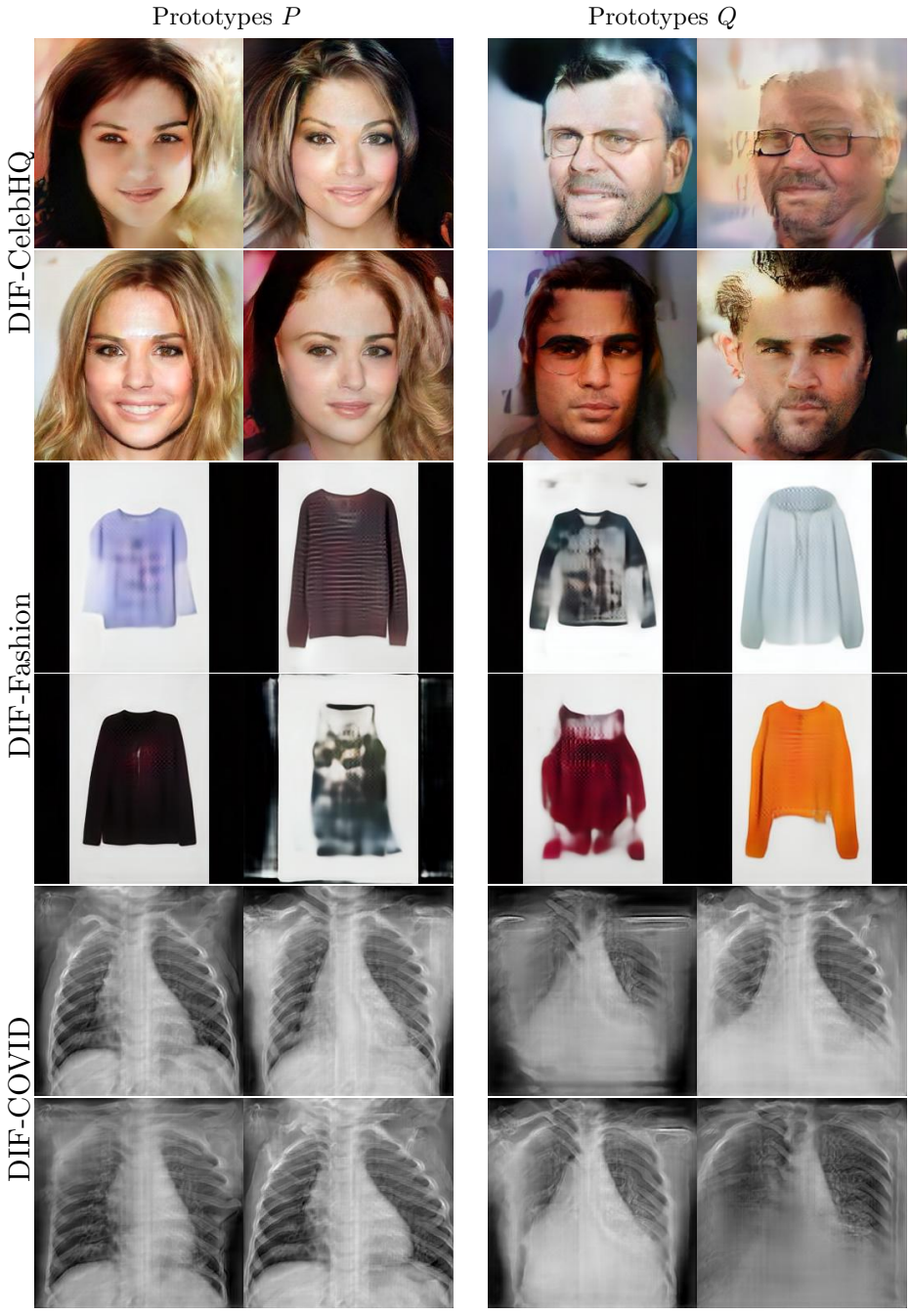


Figure 3: "Interstitial involvement in COVID-19 pneumonia. Two cases of mainly interstitial involvement in Sars-CoV-2 infection. Patient in A shows a prevalent right lung disease with diffuse reticular–nodular thickening of peribroncho-vascular interstitium. Patient in B shows a diffuse ground glass opacity mainly in the perihilar and subpleural region bilaterally" - [3, Figure 5]

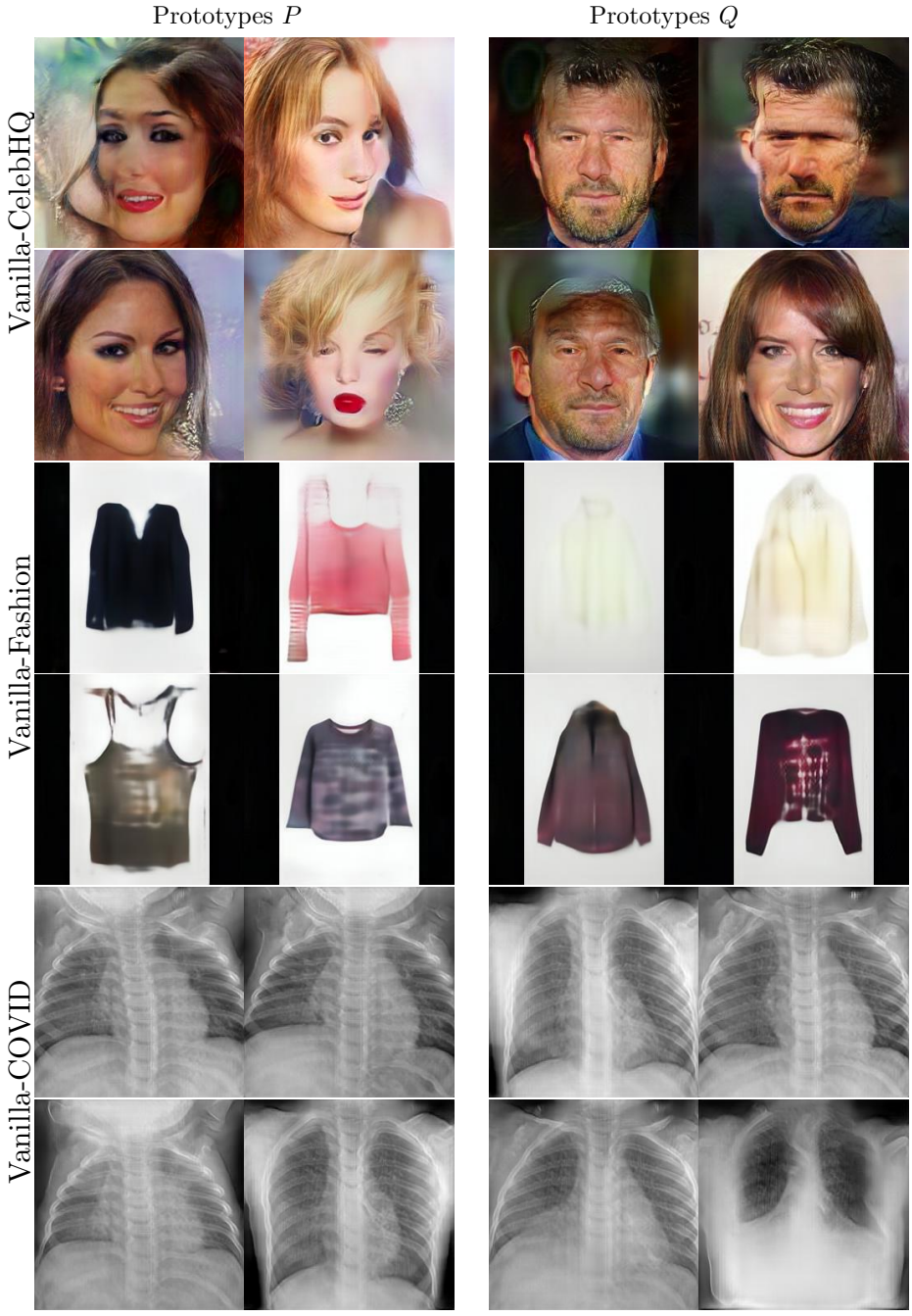
3. Additional Experimental results

3.1. Prototypes

Please see below for more prototypes.

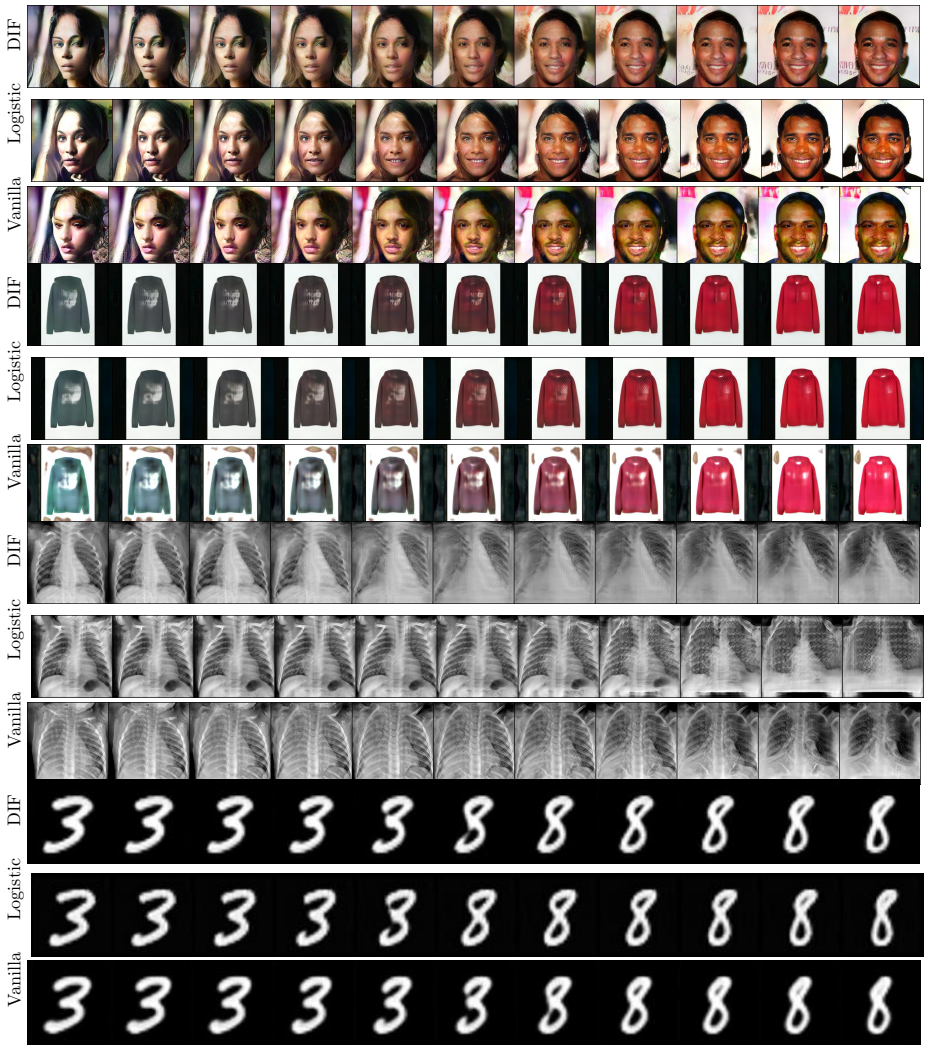

Prototypes P
Prototypes Q
DIF-CelebHQ
DIF-Fashion
DIF-COVID


Prototypes P
Prototypes Q
Logistic-CelebHQ
Logistic-Fashion
Logistic-COVID


Prototypes P
Prototypes Q
Vanilla-CelebHQ
Vanilla-Fashion
Vanilla-COVID

3.2. Image traversals

We show additional image traversals between P and Q below.



3.3. Isolating Features



Figure 8: Applying the same feature in the main paper to image 2

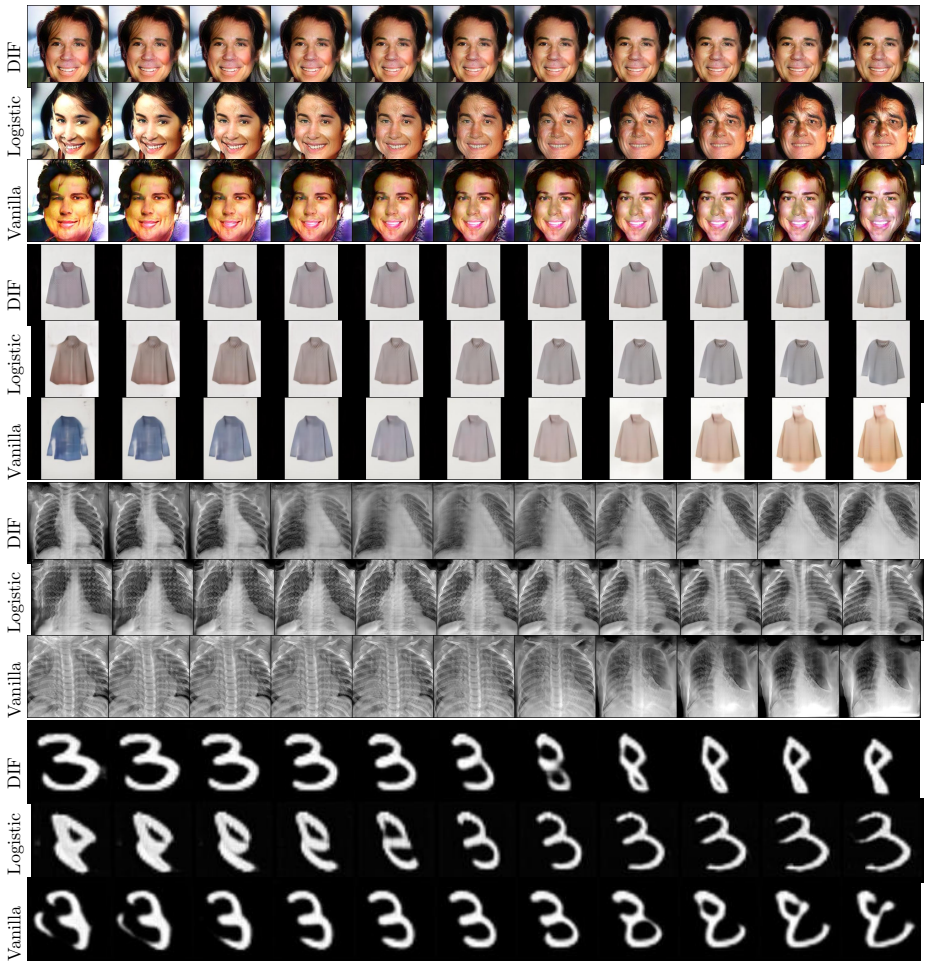


Figure 9: Applying the same feature in the main paper to image 3

3.4. Additional training behavior plots

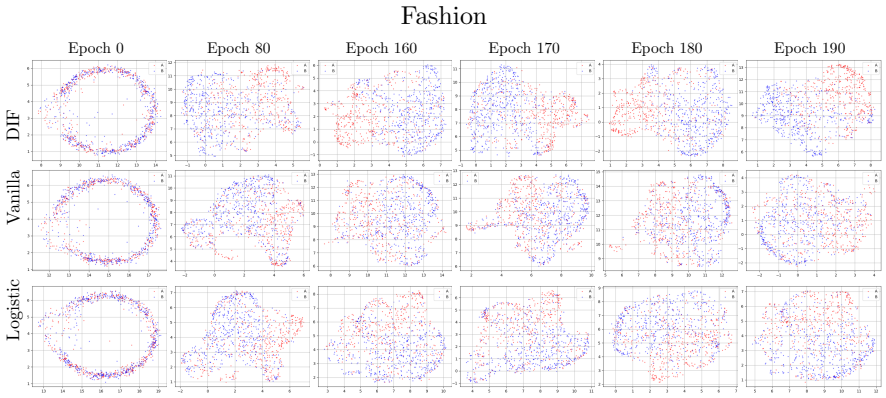


Figure 10: Evolution of latent space $\mathbf{z}$ for test data of Fashion during training.

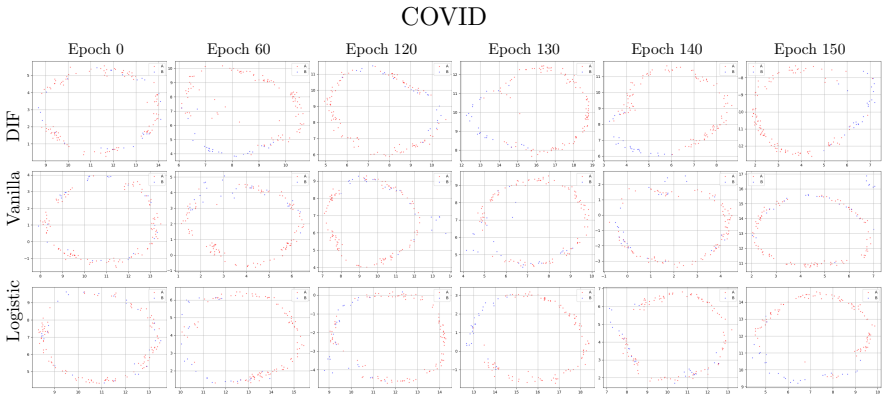
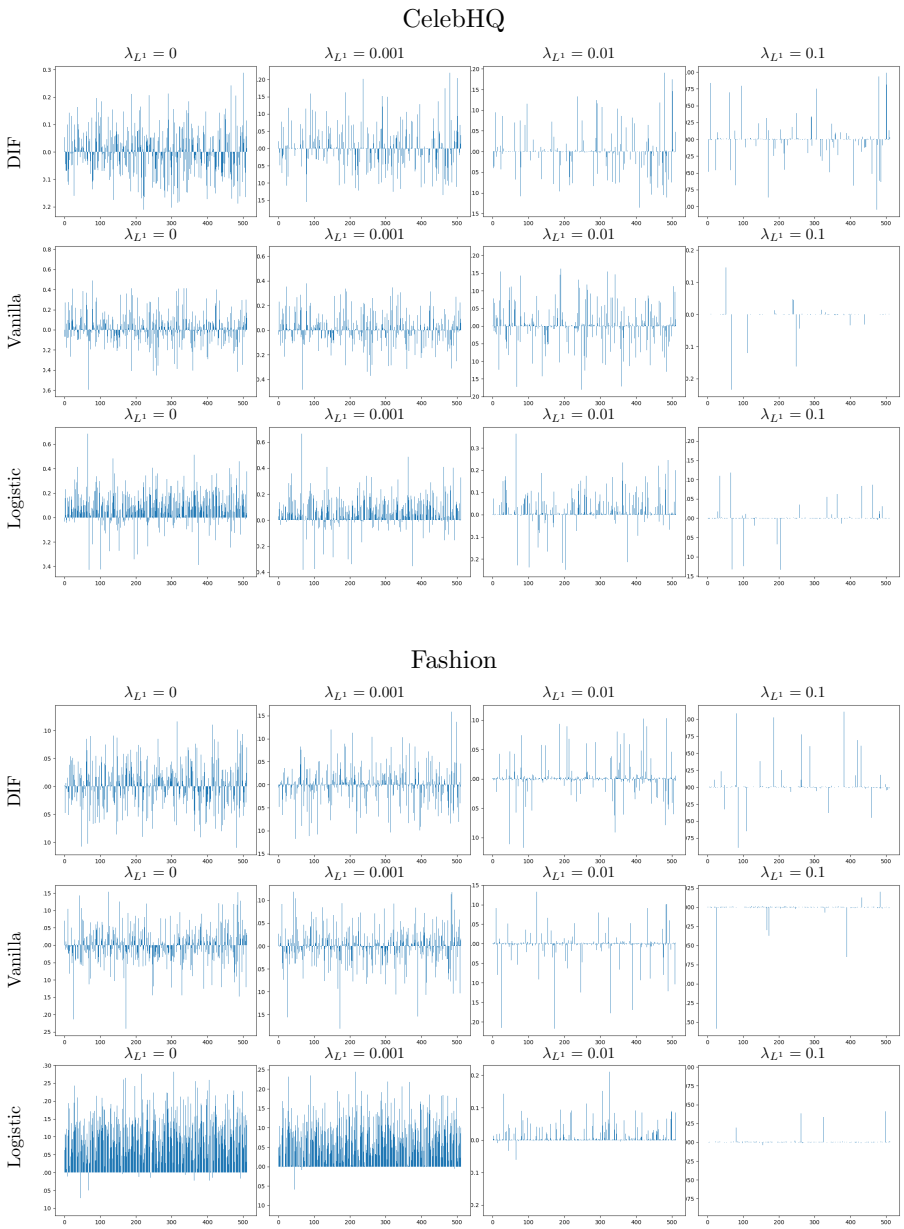
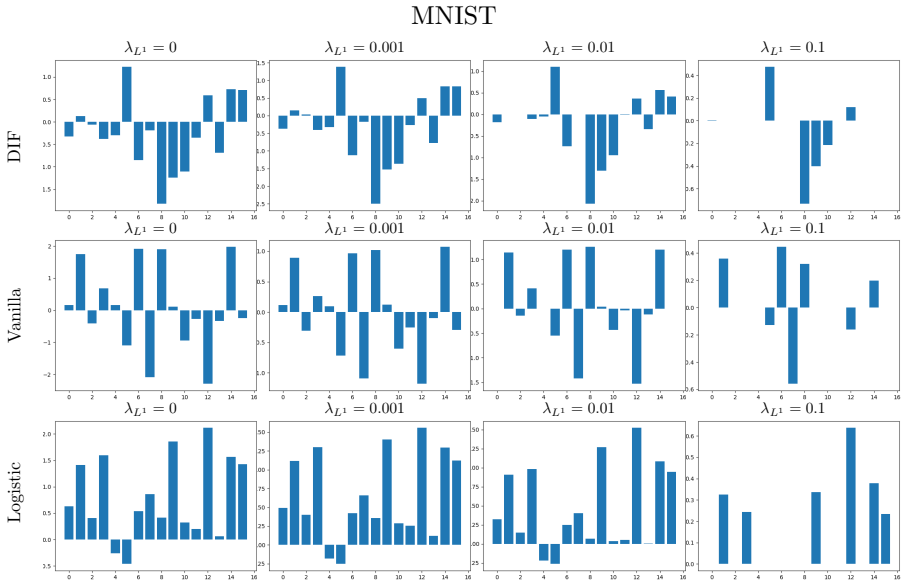
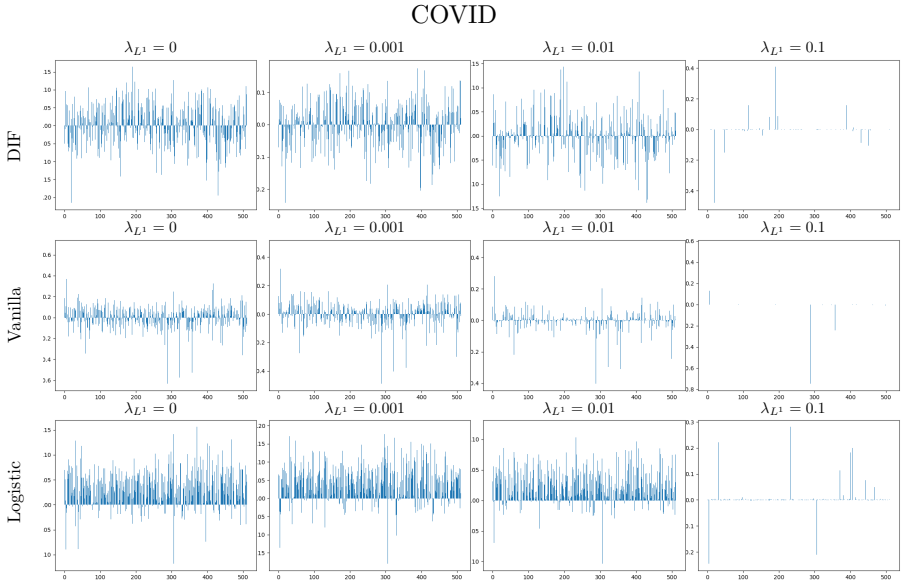


Figure 11: Evolution of latent space $\mathbf{z}$ for test data of Covid during training.

3.5. Histograms of weights for L^1 regularized logistic classification







4. Further results on training stability

We further provide training results on the datasets MNIST and COVID. The training is recorded over 100 epochs:

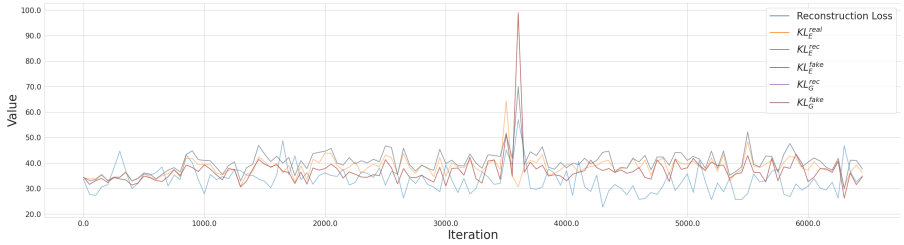


Figure 16: Training IntroVAE-DIF on MNIST.

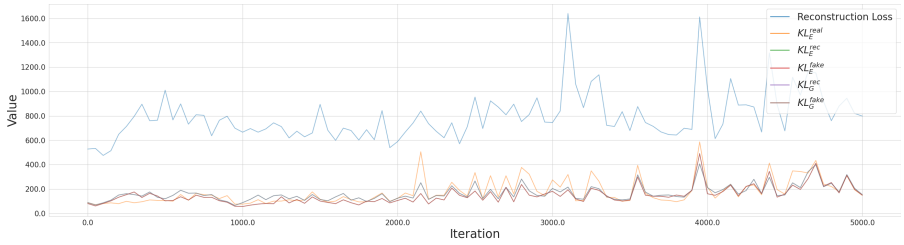


Figure 17: Training IntroVAE-DIF on COVID.

5. Comments on Reproducibility

1. **Architecture**: The architecture of the neural nets are given in the code. No modifications needed, one just needs to follow the instructions to run.

2. **Hyperparameters**: The hyperparameters for the experiments were found manually, by looking at generative quality and reconstruction quality. We also investigated the separability of the latent space through UMAP. We considered the following ranges of for hyperparameters when developing the paper: $\eta_r \in [0.01, 1]$, $\eta_\Lambda \in [0.01, 1]$, $\eta_{KL} \in [0.01, 1]$. The code contains the hyperparameters used for the results presented in the paper.

3. **Datasets**: The codebase contains instructions on how to download the preprocessed data sets. Further the code also contains preprocessing scripts for the datasets.

4. **Training**: The models are trained for an equal amount of epochs, where the number of epochs vary for each dataset. It is recommended to use a

NVIDIA RTX2080Ti to reproduce the results, since the experiments are computed using the FP16 option.

References

[1] D. Cozzi, M. Albanesi, E. Cavigli, C. Moroni, A. Bindi, S. Luvarà, S. Lucarini, S. Busoni, L. N. Mazzoni, V. Miele, Chest x-ray in new coronavirus disease 2019 (covid-19) infection: findings and correlation with clinical outcome, La radiologia medica 125 (8) (2020) 730–737. doi:10.1007/s11547-020-01232-9.
URL https://doi.org/10.1007/s11547-020-01232-9

[2] W. Kong, P. Agarwal, Chest imaging appearance of covid-19 infection, Radiology: Cardiothoracic Imaging 2 (1) (2020) e200028. arXiv:https://doi.org/10.1148/ryct.2020200028, doi:10.1148/ryct.2020200028.
URL https://doi.org/10.1148/ryct.2020200028

[3] D. Toussie, N. Voutsinas, M. Finkelstein, M. A. Cedillo, S. Manna, S. Z. Maron, A. Jacobi, M. Chung, A. Bernheim, C. Eber, J. Concepcion, Z. Fayad, Y. S. Gupta, Clinical and chest radiography features determine patient outcomes in young and middle age adults with covid-19, Radiology 0 (0) (2020) 201754, pMID: 32407255. arXiv:https://doi.org/10.1148/radiol.2020201754, doi:10.1148/radiol.2020201754.
URL https://doi.org/10.1148/radiol.2020201754

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	Towards Deep Interpretable Features
Publication Status	☐Published ☐ Accepted for Publication ☒Submitted for Publication ☐Unpublished and unsubmitted work written in a manuscript style
Publication Details	Hu, R. and Sejdinovic, D. (2019) Towards Deep Interpretable Features.

Student Confirmation

Student Name:	Robert Hu		
Contribution to the Paper	I conceived methodology, implementation and experiments. During the meetings my supervisor helped out with helpful suggestions on methodology and feedback. They also contributed a lot to the writing of the draft, checking the results, and proof-reading.		
Signature		Date	29/8-2022

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: Dino Sejdinovic			
Supervisor comments I am in agreement with the description of the contributions.			
Signature		Date	09/09/2022

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

3

A Kernel Test for Causal Association via Noise Contrastive Backdoor Adjustment

A Kernel Test for Causal Association via Noise Contrastive Backdoor Adjustment

Robert Hu ROBERT.HU@STATS.OX.AC.UK
Department of Statistics
University of Oxford
Oxford OX1 3LB, UK

Dino Sejdinovic DINO.SEJDINOVIC@STATS.OX.AC.UK
Department of Statistics
University of Oxford
Oxford OX1 3LB, UK

Robin J. Evans EVANS@STATS.OX.AC.UK
Department of Statistics
University of Oxford
Oxford OX1 3LB, UK

Abstract

Causal inference grows increasingly complex as the dimension of the number of confounders increases. Given treatments X, outcomes Y, and measured confounders Z, we develop a non-parametric method to test the *do-null* hypothesis that, after an intervention on X, there is no marginal dependence of Y on X, against the general alternative. Building on the Hilbert Schmidt Independence Criterion (HSIC) for marginal independence testing, we propose backdoor-HSIC (bd-HSIC), an *importance weighted* HSIC which combines *density ratio estimation* with kernel methods. Experiments on simulated data verify the confidence level and that the estimator has power for both binary and continuous treatments under a large number of confounding variables. Additionally, we establish convergence properties of the estimators of covariance operators used in bd-HSIC. We investigate the advantages and disadvantages of bd-HSIC against parametric tests as well as the importance of using the do-null testing in contrast to marginal or conditional independence testing. A complete implementation can be found at `https://github.com/MrHuff/kgformula`.

Keywords: Causal Inference, Noise Contrastive Estimation, Kernel Methods, Backdoor Adjustment, HSIC, Observational data

1. Introduction and Related Work

Modern causal inference often considers very large datasets with many observed confounding variables that may have vastly different properties. These settings are considered in a wide range of applications where randomized controlled trials are not always readily available: these include epidemiology (Rothman and Greenland, 2005), brain imaging (Castro et al., 2020), retail (Moriyama and Kuwano, 2021), and entertainment platforms (Dawen Liang and Blei, 2020). The inference setting is often complex, with high dimensional confounding variables needing to be accounted for. In such complex settings, non-parametric inference schemes that answer a simple query of causal association from observational data are needed as an initial step before more sophisticated causal relationships can be established.

©2000 Robert Hu and Dino Sejdinovic and Robin J. Evans.

License: CC-BY 4.0, see `https://creativecommons.org/licenses/by/4.0/`.

G-computation (Robins, 1986) is a classical method for estimating a causal effect from observational studies involving variables that are both mediators and confounders. Its popularity persists to this day (Daniel et al., 2013; Keil et al., 2020), because it allows one to test for a non-null causal effect using a variety of postulated models. The causal effect can also be identified for models fulfilling the so-called *backdoor criterion* (Pearl, 2009) and *ignorability assumptions* (Rosenbaum and Rubin, 1983). However, the use of parametric models in this context can lead to a problem known as the *g-null paradox* (Robins and Wasserman, 1997), in which the null being tested cannot logically hold because the models specified are insufficiently flexible (McGrath et al., 2021). McGrath et al. further suggest that while machine learning models might not entirely address the issue of model misspecification in g-estimation, one should avoid overly parsimonious models. Kernel methods are an example of a model class that is both flexible and not overly parsimonious.

In this paper, we propose a non-parametric approach to testing for the presence of a causal effect. In the Reproducing Kernel Hilbert Space (RKHS) and machine learning literature, the Hilbert-Schmidt Independence Criterion (HSIC) introduced by Gretton et al. (2005) is a widely used approach to non-parametric testing of independence. As the HSIC has good power properties and it is applicable to multivariate settings as well as to random variables taking values in generic domains, we use it as a foundation in this paper. Using g-computation principles, we introduce an extension of HSIC that can be applied to causal association testing.

Variations of HSIC have been proposed to test for conditional independence (Doran et al., 2014; Zhang et al., 2011; Honavar and Lee, 2017; Strobl et al., 2019) and while testing for conditional independence and causal association is different, some of the techniques used for conditional independence testing can be carried over to causal association. Kernel methods have been applied to tests for causal association in the binary treatment case in Muandet et al. (2021), and for average treatment effect estimation by Singh et al. (2020). This works aims to extend the work of Muandet et al. to a general treatment setting. We hope to bring further insights on how MMD-based approaches proposed in Singh et al. (2020) can be used for hypothesis testing.

Non-parametric tests for causal association generally come with the additional difficulty of correctly simulating the null distribution through permutations, as direct permutations often break confounding relationships as we show in Proposition 5. In this paper, we also aim to give a formal treatment on the permutation aspects of non-parametric kernel tests and expand upon ideas in Rosenbaum (1984) to consider cases beyond binary treatment.

We summarize our contributions as follows:

1. We introduce bd-HSIC, which is derived analogously to HSIC but instead uses importance weighted covariance operators, and further establish the convergence properties of the corresponding estimators in the causal setting, coupled with a novel optimization strategy to improve effective sample size.

2. We provide a novel permutation strategy for bd-HSIC that yields a permutation test with theoretically correct size for arbitrary treatment types.

3. We demonstrate that bd-HSIC has correct size and good power for different types of treatments and a large number of confounders when testing the *do-null*.

4. We analyze bd-HSIC by providing ablation studies and characterize under which circumstances it becomes invalid.

The rest of the paper is organized as follows: Section 2 describes the problem setting and provides a background on HSIC, Section 3 presents bd-HSIC and establishes convergence properties of the associated estimators, Section 4 presents additional details on the estimation procedure of bd-HSIC, Section 5 provides the experimental results and we conclude the paper in Section 6.

2. Background

Terminology We start by reviewing some terminology commonly used in causal inference:

Outcome: Intuitively represents the outcome of interest possibly caused by the treatment. It is denoted by Y and could be continuous such as salary, or binary if it represented graduation from a degree.

Treatment: Treatments intuitively refers to the variables whose effect on the outcome Y we are trying to infer. We denote *treatments* by X and in the case where treatment is placebo-controlled we can consider $X \in \{0, 1\}$ representing control and active treatment respectively. In general however, treatment could be a random variable taking values in continuous or multivariate domains.

Observed confounding variables: Intuitively represents patient characteristics that may affect both treatment and outcome. We denote these variables with Z and these could be continuous variables such as height and weight, or binary variables such as sex.

Assignment: In many confounded settings, a patient does receive treatment randomly. This means that certain patient characteristics (Z) affect *treatment assignment*. As an example, one can imagine socioeconomic factors having an affect on the availability of university level education. This creates a bias in treatment assignment which needs to be adjusted for.

Setup Consider a situation where we observe treatments X, outcomes Y and potential confounders Z defined on measurable spaces $\mathcal{X}$, $\mathcal{Y}$, and $\mathcal{Z}$, respectively. We assume these quantities are observed as $\{(z_i, x_i, y_i)\}_{i=1}^n \sim p$, where p is some joint probability density on the product space $\mathcal{Z} \times \mathcal{X} \times \mathcal{Y}$. We are interested in establishing a causal relationship between treatments X and outcomes Y, meaning that we manage to isolate whether there precisely exists a dependency between X and Y whilst holding all pre-treatment variables constant. In ideal circumstances, we would not have any confounders and such a relationship can be established straightforwardly, using e.g. regression methods. However, such circumstances are unusual outside randomized trials and would require that the treatment is assigned to units

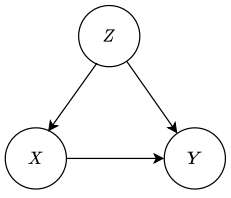


Figure 1. The relationship between treatments X, outcomes Y, and confounders Z

by some exogenous process. In the more common case of observational studies, dependencies among X, Y, Z are often depicted as in Figure 1.
The existence of confounders Z complicates establishing the causal relationship between X and Y, as they introduce dependence between X and Y which is difficult to disentangle from the postulated causal effect.

A motivating real-world problem In development economics, it is of great importance to establish causes of infant mortality (Ensor et al., 2010). The causes may often have a non-linear association with the mortality while being confounded by circumstantial factors such as the socio-economical background of the parents and the medical history of the mother.

In this setting, it is also important to test for *distributional differences* different policies might induce, as tests based on average treatment effect may fail to capture such differences as discussed in (Muandet et al., 2021; Fawkes et al., 2022). As the mean embedding framework offers a natural way to non-parametrically quantify distributional differences, kernel-based test statistics will be our preferred strategy. We will illustrate throughout the paper the importance of a non-parametric test that is able to capture non-linear dependencies for different treatments under a large number of confounders.

Definition 1. *(do-null hypothesis) Let*

$$p(y \mid do(x)) := \int p(y \mid x, z)\, p(z)\, dz$$

where in general $p(y|x) \neq p(y \mid do(x))$ *since* X *and* Z *may be dependent. We are interested in testing if the interventional distribution* $Y \mid do(X = x)$ *does not depend on the value of the treatment variable* X*. We refer to this hypothesis as a* do-null hypothesis*, which can be stated as* $Y \perp X \mid do(X = x)$*. In terms of distributions we can consider*

$$H_0 : p(y \mid do(x)) = p^*(y) \tag{1}$$

as our null hypothesis versus the general alternative, where p^* *is an arbitrary distribution that does not depend upon the value of* X*. We consider* H_0 *for all values of* $do(x)$ *and the observational regime. Note that our null hypothesis* does not *imply that* $p(y \mid do(x)) = p(y)$*, which is why we introduce* p^**.*

For a remark on why we need p^*, see Appendix A.

In this paper, we will introduce a test statistic for the hypothesis (1), but first, we review HSIC (Gretton et al., 2005), which serves as the foundation for our contribution.

2.1 Hilbert-Schmidt Independence Criterion

The Hilbert-Schmidt Independence Criterion (HSIC) is a powerful non-parametric test of independence for high-dimensional data. It considers the problem of empirically establishing whether there is any form of departure from independence between two random variables taking values on generic domains.

Marginal Independence Testing. *Let P_{xy} be a Borel probability measure defined on a domain $\mathcal{X} \times \mathcal{Y}$ and let P_x and P_y be the respective marginal distributions on $\mathcal{X}$ and $\mathcal{Y}$. Given an i.i.d. sample $(X, Y) = \{(x_1, y_1), \ldots, (x_m, y_m)\}$ of size m drawn according to P_{xy}, does P_{xy} factorize as $P_x P_y$? We usually consider the null hypothesis to be*

$$H_0 : P_{xy} = P_x P_y$$

against the general alternative

$$H_1 : P_{xy} \neq P_x P_y.$$

Since we do not have access to P_{xy}, P_x, or P_y, we need to estimate or represent these distributions through either parametric or non-parametric means. A convenient way for representing distributions is to use the RKHS formalism (Scholköpf and Smola, 2001).

HSIC can intuitively be understood as a covariance between RKHS representations of random variables.

Definition 2 (Reproducing Kernel Hilbert Spaces)**.** *Let $\mathcal{X}$ be a non-empty set and $\mathcal{H}$ a Hilbert space of functions $f : \mathcal{X} \to \mathbb{R}$. Then $\mathcal{H}$ is called a* reproducing kernel Hilbert space *endowed with dot product $\langle \cdot, \cdot \rangle$ if there exists a function $k : \mathcal{X} \times \mathcal{X} \to \mathbb{R}$ with the following properties:*

1. *k has the reproducing property*

$$\langle f, k(x, \cdot) \rangle = f(x), \quad \forall f \in \mathcal{H}, x \in \mathcal{X};$$

2. *k spans $\mathcal{H}$, that is, $\mathcal{H} = \overline{\text{span}\{k(x, \cdot) | x \in \mathcal{X}\}}$ where the bar denotes the completion of the space.*

We generally refer to the function k as a *kernel*. For certain choices of k, the corresponding RKHS is *characteristic*, i.e. the mapping of a probability measure μ, $\mu \mapsto \int_X k(x, \cdot) d\mu(x)$ is injective, i.e. μ is mapped to a unique element in the RKHS. We refer to Sriperumbudur et al. (2011) for more details. This property is very practical, as it allows us to embed probability distributions into the RKHS and calculate their expectations based on observations. These embeddings are called kernel mean embeddings, see Muandet et al. (2017) for a thorough exposition.

Definition 3. *Let $\mathcal{X}$ be a measurable space and let $\mathcal{H}_\mathcal{X}$ be an RKHS on $\mathcal{X}$ with kernel k. Let P be a Borel probability measure on $\mathcal{X}$. An element $\mu_x \in \mathcal{H}_\mathcal{X}$ such that $\mathbb{E}_{x \sim P}[f(x)] = \langle f, \mu_x \rangle$, $\forall f \in \mathcal{H}_\mathcal{X}$ is called the* ***kernel mean embedding*** *of P in $\mathcal{H}_\mathcal{X}$, where $\mu_x : P \mapsto \int_\mathcal{X} k(x, \cdot) dP(x)$.*

A sufficient condition for the existence of a kernel mean embedding is that $\mathbb{E}_{x \sim P}[\sqrt{k(x, x)}] < \infty$, which is satisfied for, e.g. bounded kernel functions.
Given some observations $\{x_i\}_{i=1}^n \sim P$, the empirical mean embedding of P is estimated as:

$$\hat{\mu}_x = \frac{1}{n} \sum_{i=1}^{n} k(x_i, \cdot).$$

Kernel mean embeddings intuitively allow us to estimate expectations under each of p_{xy}, p_x, and p_y. In order to test for marginal dependence, we consider a test statistic based on $\mathrm{Cov}[f(X), g(Y)]$, where f, g are arbitrary continuous functions evaluating random variables X, Y. Instead of picking individual functions f,g, we consider the representation of their covariance using the RKHS.

Definition 4. *Let (X, Y) be a pair of random variables defined on $\mathcal{X} \times \mathcal{Y}$ and let $\mathcal{H}_\mathcal{X}$ and $\mathcal{H}_\mathcal{Y}$ be RKHSs on $\mathcal{X}$ and $\mathcal{Y}$, respectively. An operator $C_{xy} : \mathcal{H}_\mathcal{X} \to \mathcal{H}_\mathcal{Y}$ such that*

$$\langle g, C_{xy} f \rangle = \mathrm{Cov}[f(X), g(Y)], \ \forall f \in \mathcal{H}_\mathcal{X}, g \in \mathcal{H}_\mathcal{Y}$$

*is called a **cross-covariance operator** of X and Y. The Hilbert-Schmidt Independence Criterion (HSIC) is then defined as the squared Hilbert-Schmidt (HS) norm of C_{xy}, i.e.*

$$\mathit{HSIC}(X, Y) := \|C_{xy}\|^2_{\mathrm{HS}} = \sum_i \sum_j \langle C_{xy} u_i, v_j \rangle^2$$

with u_i and v_j being orthogonal basis of $\mathcal{H}_\mathcal{X}$ and $\mathcal{H}_\mathcal{Y}$ respectively.

It is readily shown via reproducing property that C_{xy} can be written as

$$C_{xy} := \mathbb{E}\left[(k(X, \cdot)) \otimes (l(Y, \cdot))\right] - \mathbb{E}\left[k(X, \cdot)\right] \otimes \mathbb{E}\left[l(Y, \cdot)\right],$$

where k, l are kernels of $\mathcal{H}_\mathcal{X}$ and $\mathcal{H}_\mathcal{Y}$ respectively, and $\otimes$ denotes the outer product. An alternative view of HSIC is that it measures the squared RKHS distance between the kernel mean embedding of P_{xy} and $P_x P_y$. For sufficiently expressive kernels (Sriperumbudur et al., 2011), this distance is zero if and only if X and Y are independent. To see that C_{xy} indeed is a Hilbert-Schmidt operator, we refer to Muandet et al. (2017).
Given a sample $\{(x_i, y_i)\}_{i=1}^n$ from the joint distribution P_{xy}, an estimator[1] of HSIC is given by:

$$\begin{aligned}\widehat{\mathrm{HSIC}}(X, Y) = &\frac{1}{n^2} \sum_{i,j=1}^{n} k(x_i, x_j) l(y_i, y_j) + \frac{1}{n^2} \sum_{i,j=1} k(x_i, x_j) \frac{1}{n^2} \sum_{p^*,r=1}^{n} l(y_{p^*}, y_r) \\ &- \frac{2}{n^3} \sum_{i,j,r=1}^{n} k(x_i, x_j) l(y_i, y_r).\end{aligned}$$

The estimator above serves as the test statistic, for complete derivation, we refer to Gretton et al. (2005). To estimate its distribution under the null hypothesis, we resort to repeatedly permuting y_i's to obtain $\left\{\left(x_i, y_{\pi(i)}\right)\right\}_{i=1}^n$ where π is a random permutation, and recomputing HSIC on this permuted dataset. We note that the asymptotic null distribution of HSIC has a complicated form (Zhang et al., 2017), and it is hence standard practice to use a permutation approach to approximate it. For more details on HSIC, we refer to (Gretton et al., 2005).

1. This is the most commonly used, biased estimator of HSIC. An unbiased estimator also exists, cf. Song et al. (2007)

2.2 Difference between the do-null, marginal independence and conditional independence

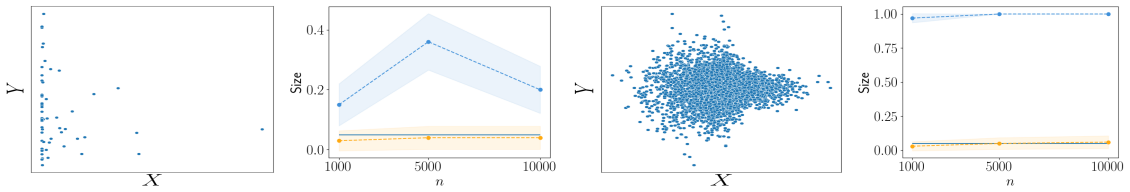
While the do-null intuitively bears many similarities to marginal independence and conditional independence, we clarify the difference between these independencies and provide an experimental demonstration that conditional independence tests (RCIT) and marginal independence tests (HSIC) cannot be used to test for the do-null.

The do-null is not marginal independence $X \perp Y$ We contrast the do-null with marginal independence $X \perp Y$, by constructing a dataset where the do-null is true, but there is marginal dependence between X and Y. To construct such a dataset, we replace the normal marginal distributions for X, Y, Z in Algorithm 4 with exponential distributions, i.e. $Y \sim \text{Exp}(X\beta_{XY})$. For an illustration, see Figure 2a.

We further plot size ($\alpha = 0.05$) against sample size when applying HSIC and bd-HSIC (true weights) in Figure 2b. HSIC is not calibrated under the do-null.

This can further be interpreted as equality in distribution of Y for all values $do(x)$ excluding the observational regime, meaning our hypothesis does not test for marginal independence between X and Y.

The do-null is not conditional independence $X \perp Y \mid Z$ We similarly contrast the do-null with conditional independence $X \perp Y \mid Z$. We simulate a dataset such that there is a conditional dependence $X \not\perp Y \mid Z$ while the do-null is true. See Figure 2c for an illustration of the dependency between X and Y. We apply the "RCIT" method, a kernel-based conditional independence test proposed in Strobl et al. (2019) and demonstrate that it is uncalibrated in relation to the do-null in Figure 2d.



(A) Marginal dependency but true do-null (B) bd-HSIC(true weights): vs HSIC: (C) Conditional dependency but true do-null, see code for more details (D) bd-HSIC(true weights): vs RCIT:

Figure 2. Difference between the do-null, marginal independence and conditional independence

3. Backdoor-HSIC

Our proposed method has two parts, a weighted HSIC test statistic and a density ratio estimation procedure. In this section, we introduce the test statistic, which we term *backdoor-HSIC* (bd-HSIC).

3.1 Overview

The core idea is simply to calculate the HSIC between

$$(X, Y) \sim p^*(x) \cdot p^*(y \mid x) \tag{2}$$

and conduct a permutation test to establish whether $p^*(y) = p^*(y \mid x)$. In the marginal independence setting, one can simply consider mean embeddings of $p^*(x)$ and $p^*(y)$. Now that we consider the interventional distribution $p^*(y \mid x)$, care must be taken to obtain the correct mean embedding. It turns out that one can use importance weights to express the mean embedding of $p^*(y \mid x)$, where the importance weights are defined as

$$w(x, z) = \frac{p^*(x)}{p(x \mid z)}, \tag{3}$$

which defines a *density ratio*. As there is an additional dependency of confounders z in the weights, a direct permutation test can not be applied. Instead, one must permute conditionally on confounders z to obtain valid samples from the null distribution. Our method then comprises of three steps:

1. Estimate the importance weights $w(x, z)$;
2. Calculate the test statistic and estimate $p(x \mid z)$ approximately;
3. Run a permutation test and calculate a p-value to establish whether $p^*(y) = p^*(y \mid x)$.

We present our proposed method in detail for steps 2 and 3 in the remainder of this section, and step 1 together with the overall procedure in Section 4.

The do-operator and identifiability We start with reviewing the meaning of the do-operator and how it lays the foundation for bd-HSIC. In the remainder of the paper, we will assume that all relevant probability distributions admit densities.

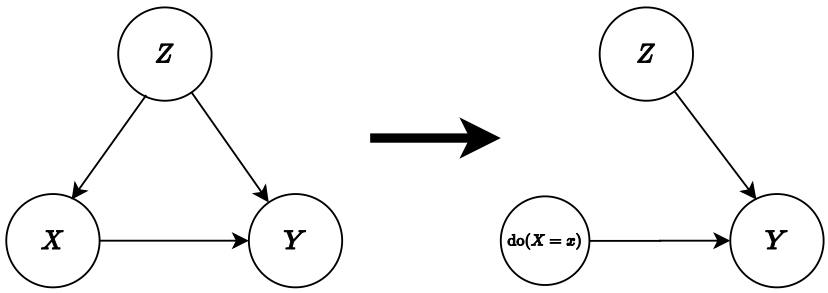


Figure 3. Illustration of the interventional distribution $p(y \mid do(x))$

In Figure 3, we first consider the graphical representation of the problem of establishing the relationship between X and Y under observed confounders Z. To adjust for confounding, we apply the do-operation (Pearl, 2009) on our treatment X, in order to remove any dependency between X and Z. The resulting distribution for the above graph, which we denote $p(z, y \mid do(x))$, can be seen as the conditional distribution of Y, Z given X where we

have made Z and X independent, but not affected the conditional distribution of Y given Z, X. This distribution can then be used to understand the *causal* relationship between X and Y.

We want to ensure the causal effect adjusted for confounders Z is identifiable for the graph in Figure 3. We revisit the *backdoor criterion* presented by Pearl (2009).

Definition 5. *A set of variables Z satisfies the back-door criterion relative to an ordered pair of variables (A, B) in a DAG G if:*

- *no node in Z is a descendant of A; and*
- *Z blocks every path between A and B that contains an arrow into A.*

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

We see that our confounders Z satisfy the backdoor criterion. By Theorem 3.3.2 in Pearl (2009) the causal effect is identifiable as

$$\begin{aligned} p^*(y|x) = p(y|do(x)) &= \int p(y|\, do(x), z) p(z \mid do(x)) dz \\ &= \int p(y|\, \underbrace{x, z}_{\text{Only causal association between } x \text{ and } y}) p(z \mid do(x)) dz \\ &= \int p(y|x, z) p(\underbrace{\quad z \quad}_{\text{Dependency between } x \text{ and } z \text{ gone}}) dz. \end{aligned}$$

Given some arbitrary density $p^*(x)$, this defines a joint density $p^*(z, x, y) = p(z)p^*(x)p(y|x, z)$.

3.2 Estimation

We are now ready to present the first link between HSIC and $p(y \mid do(x))$, as we are interested in expectations under p^* using samples from p.

Proposition 1. *Consider continuous and bounded real-valued functions f, g. The covariance between $f(X)$ and $g(Y)$ under p^* can be calculated as*

$$\begin{aligned} \mathrm{Cov}^* \left[f(X), g(Y) \right] &= \mathbb{E}_{p^*}[f(X)g(Y)] - \mathbb{E}_{p^*}[f(X)]\, \mathbb{E}_{p^*}[g(Y)] \\ &= \mathbb{E}_p[W f(X) g(Y)] - \mathbb{E}_p[f(X)]\, \mathbb{E}_p[W g(Y)], \end{aligned}$$

where $W = p^*(X)/p(X|Z)$ (provided that $p(X|Z) > 0, \forall X, Z$ s.t. $p^*(X) > 0$ and the integrals exist). Using these weights we can now calculate any expectation term under p^* in the covariance estimator.

Proof We show that $\mathbb{E}_{p^*}[g(Y)]$ and $\mathbb{E}_{p^*}[f(X)g(Y)]$ indeed can be calculated as $\mathbb{E}_p[Wg(Y)]$ and $\mathbb{E}_p[Wf(X)g(Y)]$ respectively. To see this, we have that

$$\begin{aligned}\mathbb{E}_p[Wf(X)g(Y)] &= \iiint f(x)g(y)\frac{p^*(x)}{p(x|z)}p(x,y,z)\,dx\,dy\,dz \\ &= \iiint f(x)g(y)p(z)p^*(x)p(y\mid z,x)\,dz\,dx\,dy \\ &= \iiint f(x)g(y)p(z)p^*(z,x,y)\,dz\,dx\,dy \\ &= \mathbb{E}_{p^*}[f(X)g(Y)].\end{aligned}$$

The case for $f(x) = 1$ then also follows. ■

In the above example, we will need to estimate importance weights w_i from observations $x_i \in \mathcal{X}$ and $z_i \in \mathcal{Z}$ such that $w_i = p^*(x_i)/p(x_i|z_i)$. Under the do-null, we have that $\mathbb{E}_p[Wf(X)g(Y)] = \mathbb{E}_p[f(X)]\mathbb{E}_p[Wg(Y)]$.

3.2.1 Choosing the marginal distribution of X

Calculating the covariance between two arbitrary functions can generally be tricky and may require parametric assumptions. This is obviously undesirable, so similarly to HSIC, we will consider cross-covariance operators, which represent the covariances between two functions of the variables. The key object of interest is the cross-covariance operator of treatments X and outcomes Y under $p^*(y|x)$ distribution, which we denote by $C_{p^*(y|x)}$. This operator plays an analogous role to that of the cross-covariance under an observational distribution p in standard independence testing. In particular, the squared HS norm of $C_{p^*(y|x)}$ is the population HSIC under $p^*(y|x)$ and hence the size of this operator measures departure from the do-null hypothesis. Similarly to HSIC, whenever we use characteristic kernels, we have that $\|C_{p^*(y|x)}\|^2 = 0 \iff p^*(y) = p(y\mid do(x))$. Of course, we are unable to estimate this quantity directly since we do not have access to samples from $p^*(y|x)$. The following immediate corollary to Proposition 1 relates $C_{p^*(y|x)}$ to expectations under p.

Corollary 1. *The cross-covariance operator $C_{p^*(y|x)}$ of X and Y under $p^*(y|x)$ satisfies*

$$\langle f, C_{p^*(y|x)}g\rangle = \mathbb{E}_p[W_{p^*}f(X)g(Y)] - \mathbb{E}[f(X)]\mathbb{E}_p[W_{p^*}g(Y)],\ \forall f \in \mathcal{H}_\mathcal{X}, g \in \mathcal{H}_\mathcal{Y}.$$

and $C_{p^*(y|x)}$ can be expressed as

$$C_{p^*(y|x)} := \mathbb{E}_p[W_{p^*}k(X,\cdot)\otimes l(Y,\cdot)] - \mathbb{E}[k(X_{p^*},\cdot)]\otimes\mathbb{E}_p[W_{p^*}l(Y,\cdot)]$$

with $W_{p^*} = \frac{p^*(X)}{p(X\mid Z)}$.

Following this corollary, we can empirically estimate $C_{p^*(y|x)}$ using the following expression:

$$\hat{C}_{p^*(y|x)} = \frac{1}{n}\sum_{i=1}^n \tilde{w}_i k(\cdot,x_i)\otimes l(\cdot,y_i) - \left(\frac{1}{n}\sum_{j=1}^n k\left(\cdot,x_j^{p^*}\right)\right)\otimes\left(\frac{1}{n}\sum_{i=1}^n \tilde{w}_i l\left(\cdot,y_i\right)\right), \quad (4)$$

where $\{(z_i, x_i, y_i)\}_{i=1}^n \sim p(z, x, y)$ and $\left\{x_j^{p^*}\right\} \sim p^*$. In the case where both $p^*(x_i)$ and $p(x_i|z_i)$ are known, one would simply use the "true weights" $w_i = \frac{p^*(x_i)}{p(x_i|z_i)}$. We can show that the resulting estimator (4) is consistent.

Theorem 1. *Assuming* $Var\left(\frac{p^*(x)}{p(X|Z)}\right) < \infty$, *and that* $\mathbb{E}_{X,X'}[k(X, X')] < \infty$, $\mathbb{E}_{Y,Y'}[l(Y, Y')] < \infty$, *then* $\hat{C}_{p^*(y|x)}$ *using true weights is a consistent estimator of* $C_{p^*(y|x)}$ *and satisfies*

$$\mathbb{E}\left[\|C_{p^*(y|x)} - \hat{C}_{p^*(y|x)}\|_{\text{HS}}^2\right] = \mathcal{O}\left(\frac{1}{n}\right).$$

Proof *See Appendix B.1.* ■

In practice, however, the true weights are rarely available and they would need to be estimated using density ratio estimation techniques, which we shall discuss in detail in Section 4. The weights estimation corresponds to estimating a function h, s.t. $\tilde{w}_i = h(x_i, z_i)$. We note that estimating h will need to be performed on *a different dataset* than the one used to estimate the covariance in (4), to ensure independence between h and the samples used for testing independence. We now give a result regarding the convergence rate when density ratios are estimated.

Theorem 2. *Take the conditions of Theorem 1, and assume also that* $\hat{h}_n(x, z)$ *is a consistent estimator of the density ratio* $\frac{p^*(x)}{p(x \mid z)}$ *with uniform convergence rate* $\mathcal{O}(\frac{1}{n^\alpha})$ *for* $\alpha > 0$, *i.e.*

$$\lim_{n\to\infty} \sup_{x,z} \left|\hat{h}_n(x, z) - \frac{p^*(x)}{p(x \mid z)}\right| \propto \mathcal{O}\left(\frac{1}{n^\alpha}\right).$$

Then

$$\mathbb{E}\left[\left\|C_{p^*(y|x)} - \hat{C}_{p^*(y|x)}\right\|_{\text{HS}}^2\right] = \mathcal{O}\left(\frac{1}{n^{\min(1,\alpha)}}\right).$$

Proof *See Appendix B.2.* ■

In summary, the convergence rate for the bd-HSIC estimator using estimated weights is at worst the slower rate between the estimator and the uniform convergence rate of the weight estimates. Now that we have established how to estimate the cross-covariance operator $C_{p^*(y|x)}$, analogously to HSIC, we will use the squared HS norm of $\hat{C}_{p^*(y|x)}$ as our test statistic.

Remark 1. *To the best of our knowledge, no density ratio estimator provides any uniform convergence guarantees. Theorem 2 only gives an indicative convergence rate if such density ratio estimation guarantees could be established. In the absence of such bounds, consistency will be hard to guarantee without other strong assumptions.*

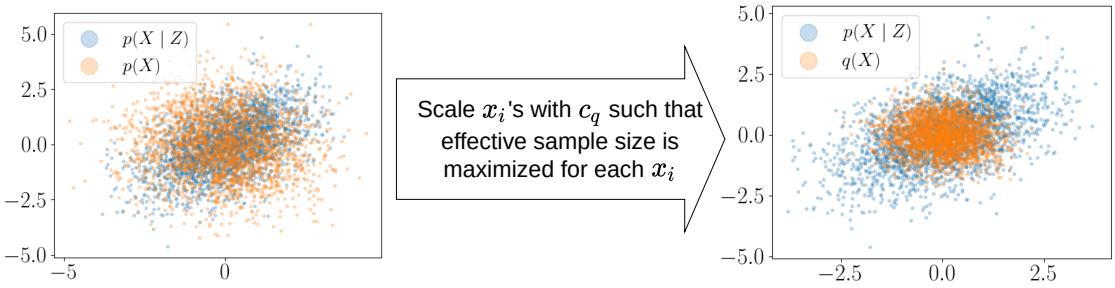


Figure 4. In the left example plot, $p(X)$ has samples where $p(X \mid Z)$ has little support. By constructing $p^*(x)$, samples from $p^*(x)$ have support where $p(X \mid Z)$ has support yield a higher effective sample size.

Proposition 2. *Let $\circ$ denote the element-wise matrix product and $\mathbf{K}_{++}$ denote the sum of all elements in the matrix $\mathbf{K}$. The squared HS norm of the estimator in (4) is given by*

$$\|\hat{C}_{p^*(y|x)}\|^2_{HS} = \frac{1}{n^2}\tilde{\mathbf{w}}^\top (\mathbf{K} \circ \mathbf{L})\, \tilde{\mathbf{w}} + \frac{1}{n^4}(\mathbf{K}^{X_{p^*}, X_{p^*}})_{++}(\mathbf{L} \circ \tilde{\mathbf{W}})_{++} - \frac{2}{n^3} \cdot \tilde{\mathbf{w}}^\top \left(\mathbf{K}^{X, X_{p^*}} \mathbf{1}_n \circ \mathbf{L}\tilde{\mathbf{w}}\right)$$

where $\tilde{\mathbf{w}} = [h(x_1, z_1), \ldots, h(x_n, z_n)]$, $\tilde{\mathbf{W}} = \tilde{\mathbf{w}}^\top \tilde{\mathbf{w}}$, $\mathbf{K} = [k(x_i, x_j)]_{i,j=1}^n$, $\mathbf{L} = [l(y_i, y_j)]_{i,j=1}^n$, $\mathbf{K}^{X_{p^*}, X_{p^*}} = [k(x_i^{p^*}, x_j^{p^*})]_{i,j=1}^n$, $\mathbf{K}^{X, X_{p^*}} = [k(x_i, x_j^{p^*})]_{i,j=1}^n$ and $\mathbf{1}_n$ is a vector of ones with length n.

Proof *See Appendix C.* ■

The above expression can be viewed as a weighted version of HSIC. We note that a weighted form of HSIC has previously been considered in a different context – when testing for independence on right-censored data (Rindt et al., 2020).

3.2.2 The choice of the p^*-marginal

Using $p^*(x)$ vs $p(x)$ In general, we would expect $p^*(x)$ to provide a higher effective sample size of w's if chosen appropriately. To maximize effective sample size we can choose $p^*(x)$ such that samples $x_i^{p^*} \sim p^*(x)$ are given by $x_i^{p^*} = c_{p^*} \cdot x_i$. For continuous x_i with mean 0, this can be viewed as scaling the variance of samples x_i. We illustrate this in Figure 4. We describe how to choose an optimal c_{p^*} for continuous densities in the paragraphs below.

Rule based selection of c_{p^*} To find an optimal c_{p^*} for univariate and multivariate X, Z, we optimize the effective sample size of $w_i = \frac{p^*(x_i)}{p(x_i \mid z_i)}$. We choose the effective sample size to be as large as possible to try to maximize the power of the test. Since the marginals of X, Z are generally unknown, we derive a heuristic based on Gaussian distributions.

Proposition 3. *Assuming standard normal marginals for univariate X, Z with correlation ρ we choose the c_{p^*} such that the effective sample size (ESS)*

$$ESS := \frac{(\sum_i w_i)^2}{\sum_i w_i^2}$$

is maximized, where w_i is the weight used to re-sample observations. The optimal c_{p^*} is then given by $c_{p^*} = \sqrt{1-2\rho^2}$. Here, $c_{p^*} \in \mathbb{R}_{>0}$. We further remark that for $\rho \geq \frac{1}{\sqrt{2}}$, this corresponds to X and Z being too correlated to yield a reasonable effective sample size.

Proof See Appendix D.1. ■

Here the optimal c_{p^*} is found analytically.

Proposition 4. *Assuming $(X, Z) \sim \mathcal{N}_{m+n}(0, \Sigma)$, where we take $\Sigma = \begin{bmatrix} \Sigma_{xx} & \Sigma_{xz} \\ \Sigma_{zx} & \Sigma_{zz} \end{bmatrix}$ and assume that $\Sigma_{xx} = I_m$ and $\Sigma_{zz} = I_n$, the optimal new covariance matrix is found by maximizing the quantity*

$$\det(D) \det\left(2T^{-1} - (I_m - \Sigma_{xz}\Sigma_{zx})^{-1} - BD^{-1}B^\top\right)$$

with respect to the positive definite matrix T, where $B := (I_m - \Sigma_{xz}\Sigma_{zx})^{-1}\Sigma_{xz}$ and $D := I_n - \Sigma_{zx}(I_m - \Sigma_{xz}\Sigma_{zx})^{-1}\Sigma_{xz}$.

Proof See Appendix D.2. ■

In practice in the multivariate case, we choose $T = I_m \cdot c_{p^*}$, and perform the optimization using gradient descent with $c_{p^*} \in \mathbb{R}_{>0}$.

3.3 The importance of using the correct permutation

For HSIC, Rindt et al. (2021) shows that permutation tests indeed provide the correct size under the null. However, when permuting under confounders their Theorem 2 does not hold anymore due to the extra dependency on Z between X and Y. We show that naïvely permuting Y will result in the incorrect size.

Proposition 5. *(Incorrect type 1 error rate for bd-HSIC using finite permutations) Consider i.i.d. samples $(z_i, x_i, y_i) \sim p$ and that H_0 is true, i.e. $p(y \mid do(x)) = p^*(y)$. If a permutation test with finite permutation samples is applied to bd-HSIC to test for causal association at level α, this test is not* guaranteed *to reject with probability less than α if x_i's or y_i's are permuted directly, assuming* $\mathrm{Var}(W) > 0$

Proof Our proof strategy is to show that if you would permute on x_i's or y_i's directly it would violate the exchangeability and hence the guarantee that the test will have the correct size in expectation, due to the additional dependencies with z_i's. To see this, permuting y_i's directly breaks the required dependency with z_i. Further permuting x_i's directly breaks the interventional distribution $do(X = x)$ as it is obtained by re-weighting x_i's according to $\frac{p^*(x_i)}{p(x_i|z_i)}$. Thus we have that, for a random permutation D,

$$(\underbrace{D\mathbf{x}}_{\text{permuting } x_i\text{'s}}, \mathbf{y}) \overset{d}{\neq} (\mathbf{x}, \mathbf{y})$$

and

$$(\mathbf{x}, \underbrace{D\mathbf{y}}_{\text{permuting } y_i\text{'s}}) \overset{d}{\neq} (\mathbf{x}, \mathbf{y}).$$

	$z=0$	$z=1$
$x=0$	p	$1-p$
$x=1$	$1-p$	p

Table 1. Probabilities for $p(X = x \mid Z = z)$

Hence the size of the test is not guaranteed to be at most α. ■

We consider a similar approach to Doran et al. (2014) who permute only where the covariates Z are similar. As bd-HSIC uses the density ratio as an importance weight, we show that permuting Y, Z against X where $p(x_i \mid z_i)$ is similar yields an exchangeable sample.

Theorem 3. *(Correct type 1 error rate for bd-HSIC using finite permutations) Assume we have i.i.d. samples $(z_i, x_i, y_i) \sim p$ and that H_0 is true, i.e. $p(y \mid do(x)) = p^*(y)$. If a permutation test with finite samples is applied to bd-HSIC to test for causal association at level α, this test will reject with probability at most α if only y_i's with the same conditional density $p(x_i|z_i)$ are being permuted.*

Proof Let D' be a permutation such that only y_i's with the same conditional density $p(\cdot|z_i)$ are being permuted. As the samples are permuted within $p(\cdot|z_i)$, the following holds

$$(\mathbf{x}, \mathbf{y}) \stackrel{d}{=} (\mathbf{x}, \underbrace{D'\mathbf{y}}_{\text{permuting } y_i\text{'s}}).$$

for any permutation D'. The computed test statistics (unpermuted and permuted data) are now exchangeable under the null, and thus the rankings will be uniformly distributed. ■

It might seem counter-intuitive that the weights w_i from the density ratio estimation cannot be used for the permutation test. We provide a corollary below why using such w_i's would fail.

Corollary 2. *Assuming a non-constant true density ratio weights, permuting "within" the density ratio weights w_i yields incorrect permutations.*

Proof We provide an example which breaks exchangeability when permuting within density ratio weights. Consider binary Z and X with $p = 0.5$ marginally, with Y being a clone of Z. Let $p(X = x \mid Z = z)$ be defined as in Table 1 with $p \neq 0.5$. As Y is a clone of Z, $p(X = x \mid Y = y)$ has the same probabilities as above. Under these assumptions, it is easy to see that (x, y) samples $(0, 0)$ and $(1, 1)$ (as well as $(1, 0)$ and $(0, 1)$) will have to permute since $p(X = 0 \mid Y = 0) = p(X = 1 \mid Y = 1) = p$. If one considers a situation in which the same number of pairs ($0, 0$ and $1, 1$) and ($1, 0$ and $0, 1$) appear in a sample, the resulting conditional distribution under permutation will be $p(Y = y \mid X = x) = 0.5$ for all $x, y \in \{0, 1\}$. Since we took $p \neq 0.5$, the permuted samples are no longer exchangeable with the original sample, even though the do-null clearly holds.

■

MMD clustering In practice, we are unlikely to get a group of samples with exactly the same density ratio. To approximately permute y_i's within $p(x_i \mid z_i)$, we propose a maximum mean discrepancy (MMD, Gretton et al., 2005) based k-means clustering method (MacQueen, 1967), which uses non-parametric kernel conditional density estimation. The procedure can be summarized as follows:

1. On training data: Estimate the conditional mean embedding $\hat{\mu}_{X|Z=z}(\cdot) = \sum_i^{n_{tr}} w_i(z)k(x_i, \cdot)$, with $\mathbf{w}(z) = (\mathbf{K}_Z + \lambda I)^{-1}\mathbf{k}_z$ and $\mathbf{k}_z = (k(z_1, z), \ldots, k(z_{n_{tr}}, z))^\top$.
2. On training data: Find cluster centers C_k by running k-means and optimize the MMD metric $\sum_i \|\mu_{X|Z=z_i}(\cdot) - \mu_{X|Z=C_k}(\cdot)\|^2 = \sum_i^{n_{tr}} (\mathbf{w}(z_i) - \mathbf{w}(C_k))^\top \mathbf{K}_X (\mathbf{w}(z_i) - \mathbf{w}(C_k))$. Note that one can optimize in "weight space" directly by taking $\mathbf{w}(C_k) = \mathbf{w}_k$.
3. On test data z_j: Assign observations to clusters by taking $c_j = \min_k (\mathbf{w}(z_j) - \mathbf{w}(C_k))^\top \mathbf{K}_X (\mathbf{w}(z_j) - \mathbf{w}(C_k))$, where c_j is the cluster assignment to test data point j. Permute y_j's within assigned clusters to approximately get a D' permutation.

To select the optimal number of clusters, we maximize the silhouette score (Rousseeuw, 1987) over a fixed number of clusters.

Consistency of bd-HSIC So far we have provided results on correct type 1 errors for permutation tests. To prove the asymptotic consistency of bd-HSIC, it suffices to show that Lemma 1 in Rindt et al. (2021) holds for bd-HSIC, assuming correctly estimated weights and access to $p(x_i|z_i)$. Consistency then follows from Theorem 3 in Rindt et al. (2021).

Proposition 6. *Let ψ be a random permutation of y_i's such that they only are permuted within groups that have the same $p(x_i \mid z_i)$. Further let $w_i = \frac{p^*(x_i)}{p(x_i|z_i)}$. Then*

$$\psi(\|\hat{C}_{p^*(y|x)}\|^2_{HS}) \to 0$$

in probability.

Proof See Appendix B.3. ■

Remark 2. *It should be noted that we have not used samples $x^{p^*} \sim p^*$ in the test statistic for the proposition above. If on were to use x^{p^*} in the test statistic, the proof strategy can be repeated up to the final sum of ζ's, which instead will be*

$$\mathbb{E}\left[l(Y, Y')\right]\left(\mathbb{E}\left[k(X, X')\right] + \mathbb{E}\left[k(X^{p^*}, X'^{p^*})\right] - 2\mathbb{E}\left[k(X, X^{p^*})\right]\right)$$

which generally will not sum to 0, implying that we lose consistency. We use samples x^{p^*} to empirically improve power at the cost of being biased.

While the assumptions seem unrealistic, i.e. requiring perfect estimation of the weights and a conditional density, it at least shows that when these quantities are being estimated in a consistent manner, we can expect the power of the test to improve with sample size.

4. Estimation of Weights

Since we will generally never have access to the true weights needed for backdoor adjustment, we need to estimate them from observed data. We denote estimated weights as $\tilde{w}_i$. In this section, we describe how to estimate these $\tilde{w}_i$ for both categorical and continuous X.

4.1 Categorical treatment variable X

For categorical x_i we estimate w_i using the ratio $\frac{p^*(x_i)}{p(x_i|z_i)}$, where we take $p^*(x) = p(x)$. We first estimate $p(x)$ by simply taking the empirical probabilities for each category using the training data. For $p(x \mid z)$, we fit a probabilistic classifier mapping from z to each class of x. When X consists of multiple categorical dimensions, i.e. $d_X > 1$, we consider the $p^*(x)$ and $p(x \mid z)$ over the joint space of $\mathcal{X}_1 \times \ldots \times \mathcal{X}_{d_X}$. In the cases where d_X is large (≥ 8), we assume each x^d to be independent of each other and take $p(x) = \prod_{d=1}^{d_X} p(x_d)$ and $p(x \mid z) = \prod_{d=1}^{d_X} p(x_d \mid z)$.

4.2 Continuous treatment variable X

In the continuous case, we can no longer estimate $p(x \mid z)$ or $p^*(x)$ using a classifier straightforwardly. This complicates the estimation of the density ratio $w = \frac{p^*(x)}{p(x \mid z)}$, as we could either estimate $p^*(x)$ and $p(x \mid z)$ separately using density estimation or estimate the density ratio directly. We review some existing methods below.

Direct density estimation of $p^*(x)$ and $p(x \mid z)$ allows for a broad range of methods such as normalizing flows (Rezende and Mohamed, 2015), generative adversarial networks (Goodfellow et al., 2014) and kernel density estimation (Botev et al., 2010) among many. While these methods provide accurate density estimation, they tend to be computationally expensive and hard to train (Mescheder et al., 2018). In this paper, we do not explore them further since densities themselves are not of direct interest.

RuLSIF (Yamada et al., 2011) propose using kernel ridge regression to directly estimate the density ratio $r(x) = \frac{p_1(x)}{p_2(x)}$ between distributions $p_1(x)$ and $p_2(x)$. While this method offers an analytical approach to estimating the density ratio, a regression may often not be flexible enough to learn our density ratio of interest in a high dimensional setting. We compare against RuLSIF in the experiment section.

Noise contrastive density estimation (NCE) (Gutmann and Hyvärinen, 2012) considers the problem of estimating an unknown density $p_{\text{true}}(\mathbf{x}; \theta)$ with parameters θ from samples $\mathbf{x} \in \mathbb{R}^d$. The key idea of NCE is to convert a density estimation problem to a classification problem by selecting an auxiliary noise contrastive distribution $p_{\text{noise}}(\mathbf{x})$ to compare with samples from p_{true}. This noise contrastive distribution is used to train a density ratio $\hat{p}(\mathbf{x}; \theta')$ with parametrization θ' of p_{true} to distinguish between fake samples $\mathbf{z} \sim p_{\text{noise}}$ and observations $\mathbf{x}_i \sim p_{\text{true}}$ through binary classification. We can then derive p_{true} by using this estimated density ratio. NCE has numerous desirable properties, including consistency under mild assumptions. We will propose and use a slightly modified NCE method to estimate our desired density ratio. We detail this method in the next section.

Telescoping density ratio (TRE) (Rhodes et al., 2020) considers the problem of estimating the density ratio between distributions p_0 and p_m using samples $\mathbf{x}_0 \sim p_0$ and $\mathbf{x}_m \sim p_m$. However, these density ratio problems tend to become pathological when p_0 and p_m are too far apart, exhibiting a phenomenon coined *density chasm*. The main idea of TRE is then to decompose this density ratio into several sub-tasks through a telescoping product

$$\frac{p_0(\mathbf{x})}{p_m(\mathbf{x})} = \frac{p_0(\mathbf{x})}{p_1(\mathbf{x})}\frac{p_1(\mathbf{x})}{p_2(\mathbf{x})} \cdots \frac{p_{m-2}(\mathbf{x})}{p_{m-1}(\mathbf{x})}\frac{p_{m-1}(\mathbf{x})}{p_m(\mathbf{x})},$$

and estimate each individual ratio with separate estimators $r_k(\mathbf{x}; \theta_k) \approx p_k(\mathbf{x})/p_{k+1}(\mathbf{x})$ for $k = 0, \ldots, m-1$; we can then compose the original density ratio as

$$r(\mathbf{x}; \theta) = \prod_{k=0}^{m-1} r_k(\mathbf{x}; \theta_k) \approx \prod_{k=0}^{m-1} \frac{p_k(\mathbf{x})}{p_{k+1}(\mathbf{x})} = \frac{p_0(\mathbf{x})}{p_m(\mathbf{x})}.$$

To train each estimator $r_k(\mathbf{x}; \theta_k)$ we require a gradual transformation of samples between $\mathbf{x}_0$ and $\mathbf{x}_m$, resulting in intermediate samples $\mathbf{x}_k \sim p_k, \quad k = 0, \ldots, m-1$. We define these samples as a linear combination of $\mathbf{x}_0$ and $\mathbf{x}_m$

$$\mathbf{x}_k = \sqrt{1 - \alpha_k^2}\mathbf{x}_0 + \alpha_k \mathbf{x}_m, \quad k = 0, \ldots, m$$

where the α_k's form an increasing sequence from 0 to 1. The training objective

$$\mathcal{L}_{\text{TRE}}(\theta) = \frac{1}{m}\sum_{k=0}^{m-1} \mathcal{L}_k(\theta_k)$$

$$\mathcal{L}_k(\theta_k) = -\mathbb{E}_{\mathbf{x}_k \sim p_k} \log\left(\frac{r_k(\mathbf{x}_k; \theta_k)}{1 + r_k(\mathbf{x}_k; \theta_k)}\right) - \mathbb{E}_{\mathbf{x}_{k+1} \sim p_{k+1}} \log\left(\frac{1}{1 + r_k(\mathbf{x}_{k+1}; \theta_k)}\right)$$

is the average of all m losses of the subtasks.

4.3 NCE for bd-HSIC

In the do-null context, we have access to samples $\{(x_i, y_i, z_i)\}_{i=1}^n \sim p$. To calculate $\|\hat{C}_{p^*(y|x)}\|^2$ we need to estimate the density ratio $\tilde{w}_i = \frac{p^*(x_i)}{p(x_i|z_i)}$ from our observations. Here $p^*(x)$ is a chosen marginal distribution of X. We can express the density ratio as

$$w_i = \frac{p^*(x_i)}{p(x_i|z_i)} = \frac{p^*(x_i)p(z_i)}{p(x_i, z_i)}.$$

If we take $p^*(x) = p(x)$, the problem translates into finding the density ratios between the product of the marginals and the joint density.

We can make use of the NCE framework by taking joint samples $D_1 = \{(x_i, z_i)\}_{i=1}^{n_1}$ and approximate samples from the product of the marginals $D_2 = \left\{\left(x_i, z_{\pi(i)}\right)\right\}_{i=1}^{n_2} \sim p(x)p(z)$, for a randomly drawn permutation π. We obtain D_1 and D_2 by splitting the dataset to ensure independence between positive samples and negative samples in NCE.

By setting $p_{\text{true}}(x,z) = p(x,z)$ and $p_{\text{noise}}(x,z) = p(x)p(z)$, we can parametrize the noise contrastive classifier as $h(x,z;\theta) = \sigma(\ln(\frac{1}{\nu}\frac{p(x)p(z)}{p(x,z;\theta)})) = \sigma(\ln(\frac{1}{\nu}r(x,z;\theta)))$, where $r(x,z;\theta)$ is a classifier parametrized by θ, $\sigma(x) = \frac{1}{1+e^{-x}}$ and $\nu = \frac{n_2}{n_1}$, the ratio between samples from the product of the marginal to the joint distribution. By using NCE to directly estimate the density ratio between the product of the marginals and the joint density we gain the advantage that we do not need to specify an explicit noise contrastive distribution (and potentially introduce bias), as we can already obtain samples approximately through permutation. The problem is then reduced to a classification problem where we have to discriminate between samples from the product of marginals and the joint distribution. We introduce two modifications which take advantage of using the marginal $p^*(x)$.

NCE-p^* Consider any $p^*(x) \neq p(x)$. We then build a classifier to discriminate datasets $D_1 = \{(x_i, z_i)\}$ from $D_2^{p^*} = \{(x_j^{p^*}, z_j)\}$ where $\{x_j^{p^*}\} \sim p^*$ independently of z_j, so $D_2^{p^*}$ contains samples from $p^*(x)\,p(z)$. When we pass any new pair (x,z) (i.e. regardless where it comes from, and in particular it can come from $p(x,z)$) to the classifier, it gives us the density ratio

$$\frac{p^*(x)p(z)}{p(x,z)} = \frac{p^*(x)}{p(x \mid z)},$$

which is then parametrized as

$$h(x,z;\theta) = \sigma\left(\ln\left\{\frac{1}{\nu}\frac{p^*(x)p(z)}{p(x,z;\theta)}\right\}\right) = \sigma\left(\ln\left\{\frac{1}{\nu}r(x,z;\theta)\right\}\right).$$

TRE-p^* We can apply TRE to density ratio estimations between joint samples $\mathbf{x}_0 = (x,z)$ and product of marginals $\mathbf{x}_m = \left(X^{p^*}, z\right)$. For our particular context involving a chosen $p^*(x)$, we generate intermediate samples $\mathbf{x}_k = \left(\sqrt{1-\alpha_k^2}x + \alpha_k X^{p^*}, z\right)$ by fixing z for $k = 0, \ldots, m$.

It should be noted that neither TRE-p^* or NCE-p^* provides any guarantees in regard to uniform convergence rates of consistency.

4.4 Mixed treatment X

When X contains both continuous and categorical treatments, modifications to the continuous method are needed. We observe that if we take $X = x_{\text{cat}} \cup x_{\text{cont}}$ we have

$$w(x_{\text{cat}}, x_{\text{cont}}) = \frac{p(x_{\text{cat}}, x_{\text{cont}})}{p(x_{\text{cat}}, x_{\text{cont}} \mid z)} = \frac{p(x_{\text{cat}}, x_{\text{cont}})}{p(x_{\text{cat}} \mid z, x_{\text{cont}})p(x_{\text{cont}} \mid z)} \tag{5}$$

$$= \underbrace{\frac{p(x_{\text{cat}} \mid x_{\text{cont}})}{p(x_{\text{cat}} \mid z, x_{\text{cont}})}}_{\text{Classifiers}} \cdot \underbrace{\frac{p(x_{\text{cont}})}{p(x_{\text{cont}} \mid z)}}_{\text{NCE}}. \tag{6}$$

We note that we can decompose the density ratio into a *product* of density ratios estimated using classifiers and density ratio estimation methods. We will use this as the main method for mixed treatment data since this composition allows us to simplify the problem by avoiding estimating density ratios over joint categorical and continuous treatment data which

could induce density chasms (Rhodes et al., 2020). We compare our proposed method to dimension-wise *mixing*, proposed in Rhodes et al. (2020). The same techniques can also be applied to NCE-p^*.

4.5 Algorithmic procedure

We describe the procedures for estimating weights $\tilde{w}$ and our proposed testing procedure.

Training the density ratio estimator We train our density ratio estimators $r(x, z; \theta)$ through gradient descent. We parameterize all our estimators as Neural Networks (NN), due to their ability to fit almost any function. We summarize the training procedure for categorical data in Algorithm 1 and continuous data in Algorithm 2. We take our validation criteria in Algorithm 2 to be out-of-sample loss.

Algorithm 1: Training a density ratio estimator for categorical X

Input: Data $\mathcal{D} = \{x_i, z_i, x_i^{p^*}\}_{i=1}^{n//2}$
Output: Trained estimator $r(x, z; \theta')$
Partition data into training and validation $\mathcal{D} = \mathcal{D}_{tr} \cup \mathcal{D}_{val}$
Initialise estimator $r(x, z, \theta')$
if $D > 8$ **then**
  for *each categorical* x^d **do**
    Estimate $p(x^d)$ using empirical probabilities
    Estimate $p(x^d \mid z)$ using a classifier with parameters θ_d
    Set $r^d(x^d, z; \theta_d) = \frac{p(x^d)}{p(x^d \mid z; \theta_d)}$
  end
  Set $r(x, z) = \prod_{d=1}^{n} r^d(x^d, z; \theta_d)$
end
else
  Estimate $p(x)$ over joint space
  Estimate $p(x \mid z)$ using a classifier θ
  Set $r(x, z) = \frac{p(x)}{p(x \mid z; \theta)}$
end
Return: Trained estimator $r(x, z)$

Testing procedure The entire procedure of the test can be summarized in Algorithm 3.

It should be noted that we partition the data such that the data used for the estimation of weights are independent of the data used for the permutation test.

5. Simulations

We run experiments using bd-HSIC in the following contexts of do-null testing:

1. Linear X, Y dependencies under multiple treatments and treatment types, multiple confounders and multiple outcomes

Algorithm 2: Training an NCE-based density ratio estimator

Input: Data $\mathcal{D} = \{x_i, z_i, x_i^{p^*}\}_{i=1}^{n//2}$
Output: Trained estimator $r(x, z; \theta)$
Partition data into training and validation $\mathcal{D} = \mathcal{D}_{tr} \cup \mathcal{D}_{val}$
Initialize estimator $r(x, z, \theta)$
Partition data into joint and product of the margin samples $\mathcal{D}_{tr} = \mathcal{D}_{tr}^{\text{pom}} \cup \mathcal{D}_{tr}^{\text{joint}}$, $\mathcal{D}_{val} = \mathcal{D}_{val}^{\text{pom}} \cup \mathcal{D}_{val}^{\text{joint}}$
while *validation criteria* ν *not converged* **do**
  Sample positive and negative samples $\delta_+ \subset \mathcal{D}_{tr}^{\text{pom}}, \delta_- \subset \mathcal{D}_{tr}^{\text{joint}}$;
  Calculate loss $l = \mathcal{L}(r(\delta_+), r(\delta_-))$;
  Gradient Descent $\theta = \theta + \frac{\partial l}{\partial \theta}$;
  Calculate validation loss $\mathcal{L}(r(\mathcal{D}_{val}^{\text{pom}}), r(\mathcal{D}_{val}^{\text{joint}}))$;
end
Return: Trained estimator $r(x, z)$

Algorithm 3: Testing $H_0 : p(y \mid do(x)) = p^*(y)$

Input: Data $\{(x_i, y_i, z_i)\}_{i=1}^n$, Distribution $p^*(x)$, density ratio estimator $r(\cdot)$, number of permutations n_q
Output: p-value for H_0
Find optimal c_{p^*} for continuous X
Sample $\{X_i^{p^*}\}_{i=1}^n \sim p^*$
Partition data into $\mathcal{D}_1 = \left\{(x_i, y_i, z_i, X_i^{p^*})\right\}_{i=1}^{\lfloor n/2 \rfloor}$ and $\mathcal{D}_2 = \left\{(x_i, y_i, z_i, X_i^{p^*})\right\}_{i=\lfloor n/2 \rfloor + 1}^{n}$
Train r on $\mathcal{D}_1$
Estimate $p(x \mid z)$ on $\mathcal{D}_1$ using modified k-means
Get weights $\{\tilde{w}_i\}_{i=\lfloor n/2 \rfloor + 1}^n = r(\left\{(x_i, z_i, X_i^{p^*})\right\}_{i=\lfloor n/2 \rfloor + 1}^{n})$
Calculate $\|\hat{C}_{p^*(y|x)}\|^2$ using $\mathcal{D}_2$
Calculate permuted test statistics $\{\|\hat{C}_{p^*(y|x)}\|_1^2, \ldots, \|\hat{C}_{p^*(y|x)}\|_{n_q}^2\}$
Calculate the p-value as
$p = 2 \min\left(1 - \frac{1 + \sum_{i=1}^{n_q} 1_{\|\hat{C}_{p^*(y|x)}\|^2 < \|\hat{C}_{p^*(y|x)}\|_i^2}}{1 + n_q}, \frac{1 + \sum_{i=1}^{n_q} 1_{\|\hat{C}_{p^*(y|x)}\|^2 < \|\hat{C}_{p^*(y|x)}\|_i^2}}{1 + n_q}\right)$
Return: p

2. Non-linear X, Y dependencies under multiple treatments, multiple confounders and multiple outcomes

To extend the exposition of bd-HSIC, we contrast against post-double selection (PDS), which serves as a representative benchmark with pathologies (including, but not limited to the g-null paradox) that bd-HSIC attempts to amend.

Comparison against parametric methods We compare against the popular post-double selection (PDS) method (Belloni et al., 2014), a parametric lasso-based model that is widely used for causal estimation. PDS considers the following setup

$$\begin{aligned} y_i &= d_i\alpha_0 + g(z_i) + \zeta_i, & \mathbb{E}[\zeta_i \mid z_i, d_i] = 0 \\ d_i &= m(z_i) + v_i, & \mathbb{E}[v_i \mid z_i] = 0 \end{aligned} \tag{7}$$

where y_i is the outcome variable, d_i is the policy/treatment variable whose impact α_0 is the quantity of interest, z_i represents confounding factors , and ζ_i and v_i are disturbances. The problem is then recast into a linear form

$$\begin{aligned} y_i &= d_i\alpha_0 + \underbrace{x_i'\beta_{g0} + r_{gi}}_{g(z_i)} + \zeta_i, \\ d_i &= \underbrace{x_i'\beta_{m0} + r_{mi}}_{m(z_i)} + v_i, \end{aligned} \tag{8}$$

where $x_i'\beta_{g0}$ and $x_i'\beta_{m0}$ are approximations to $g(z_i)$ and $m(z_i)$, and r_{gi} and r_{mi} are the corresponding approximation errors. PDS then uses lasso to estimate $m(z_i)$ and $g(z_i)$ and for selecting non-zero control variables, and then regresses y_i on d_i together with the union of selected non-zero control variables. Note that PDS is a lasso-based model, which makes it susceptible to non-linear confounding effects and causal effects. Given the above contexts, the coverage of PDS includes linear X, Y dependencies with multiple treatments under multiple confounders.

To make comparisons straightforward, we only compare to PDS in univariate treatment, confounder and outcome cases.

Comparisons against CfME and Singh et al. (2020) We additionally compare against CfME proposed in Muandet et al. (2021) for the binary treatment case and Singh et al. (2020) for general treatment. While it is not immediately clear how to adapt Singh et al. (2020) for a hypothesis test, we provide a derivation in Appendix E. We will refer to the newly derived test as *backdoor-CME* (bd-CME). For both these cases, we consider direct permutation on Y following the implementation of CfME.

An inherent limitation of CfME is that it only can be used for binary treatment cases. For bd-CME, which is entirely kernel dependent, problems could arise when the data is of both categorical and continuous variables, as it becomes unclear how to adequately select kernels for both of these data types. One could of course consider a product kernel, but such a kernel may break certain dependencies that render the test invalid.

Ablation study We further compare to $\tilde{w}_i \sim \text{Uniform}(0, 1)$. While this choice of weights is not a very principled approach, it serves as a reference to see whether bd-HSIC actually needs correctly estimated weights to have power.

Studying the size under H_0 When the null hypothesis is true, we would expect the p-value distribution of the test run on several datasets to have the correct size. Here we use level $\alpha = 0.05$.

Studying the power under H_1 Under the alternative, a desired property is high power across all parameters of the data generation. We will conduct experiments to demonstrate when the test has high power and when it does not. We calculate power for level $\alpha = 0.05$.

We present our results in bivariate plots, where we plot β_{XY} (x-axis) against the power at level $\alpha = 0.05$ (y-axis). An ideal plot would be a discontinuous function with height 0.05 at $\beta_{XY} = 0$ and height 1.0 for $\beta_{XY} > 0$. We would then expect that as the sample size increases, the plot would approach this ideal.

5.1 Results

We divide our experiments into two steps: we first take the *true weights* and use them in the subsequent permutation test for bd-HSIC. In this initial step, we chose $p^* = p$. This is intended as a unit test to validate that the parameter selection used for data generation is working. It should be noted that generally the choice of p^* and estimation of c_{p^*} must be done cautiously to avoid double use of data.

In the second step we estimate weights from the data and investigate the effectiveness of our proposed density ratio estimation procedure. We consider the following methods for weight estimation: RuLSIF, random uniform, NCE-p^* and TRE-p^* as described in Section 4.3 and Section 4.

5.1.1 Binary treatment X

We simulate univariate binary data according to Appendix F.1. We vary the dependence β_{XY} between X and Y to be $\{0.0, 0.005, 0.01, 0.02, 0.04, 0.06, 0.08, 0.1\}$. We plot the power for the level $\alpha = 0.05$ against β_{XY} in Figure 5.

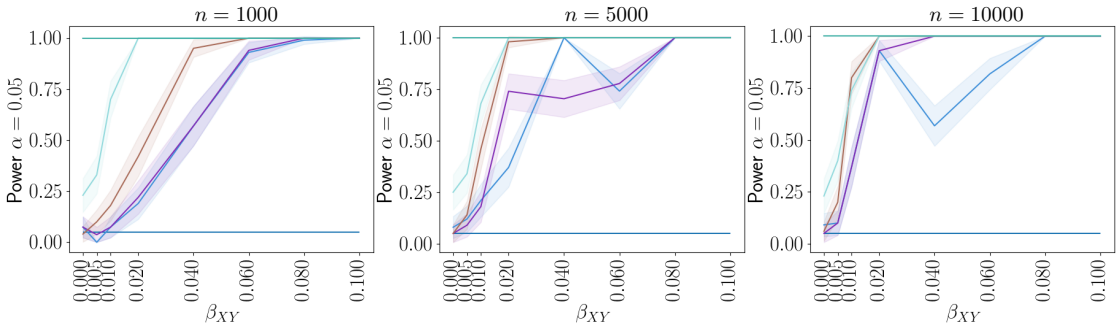


Figure 5. bd-HSIC(NCE-p^*): bd-HSIC(true weights): PDS: CfME: bd-CME:
Binary treatment for $n = 1000, 5000, 10000$ using an RBF kernel in bd-HSIC. Both CfME and bd-CME has incorrect size.

5.1.2 Continuous treatment X

Linear dependency between X and Y The data are simulated for $(d_z, d_x, d_y) \in \{(1,1,1),(3,3,3),(15,3,3),(50,3,3)\}$. In our experiments, we found that a strong confounding effect (i.e. large β_{XZ}) led to a smaller effective sample size, making it harder to obtain a consistent test under H_0. For H_1, the difficulty was mostly controlled by the magnitude of β_{XY}, where small magnitudes often led to a test with little or no power. We have chosen rejection sampling parameters $\theta, \phi, \beta_{XZ}, \beta_{YZ}$ such that the tests are non-trivial but not a failure mode, where θ, ϕ control variance of the marginal distribution of the treatment and the variance of proposal distribution respectively. For exact simulation details, we refer to the appendix. We consider $\beta_{XY} \in [0.0, 0.05]$, with $\beta_{XY} = 0.0$ corresponding to H_0. We present results for continuous treatment in Figure 6. We note that bd-CME has inflated type 1 errors for $d_Z = 15$.

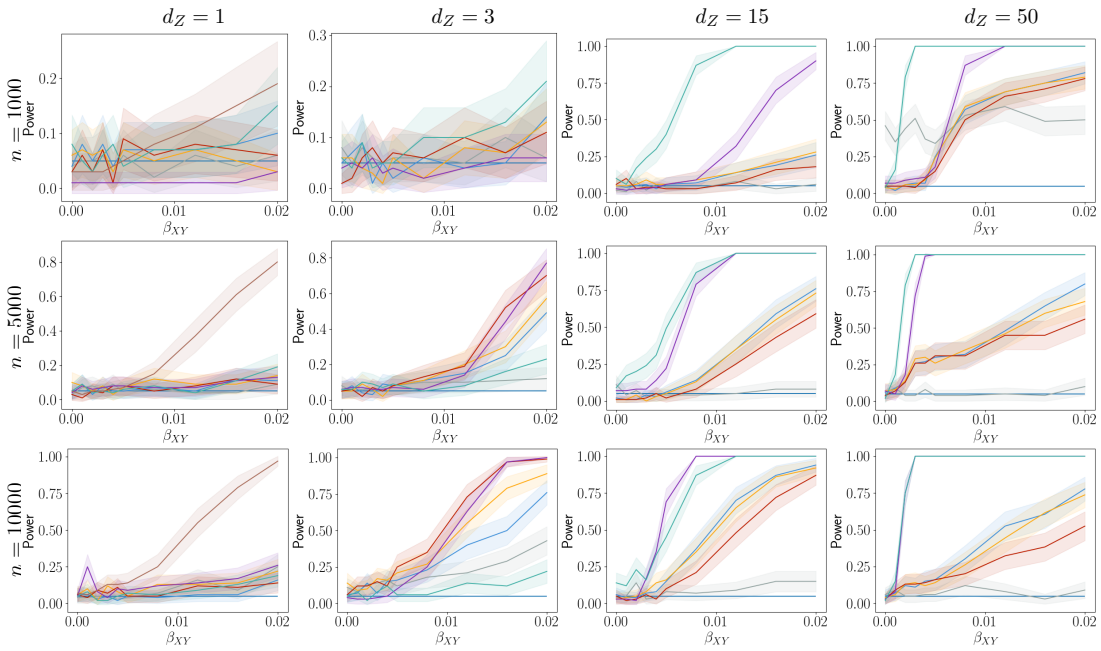
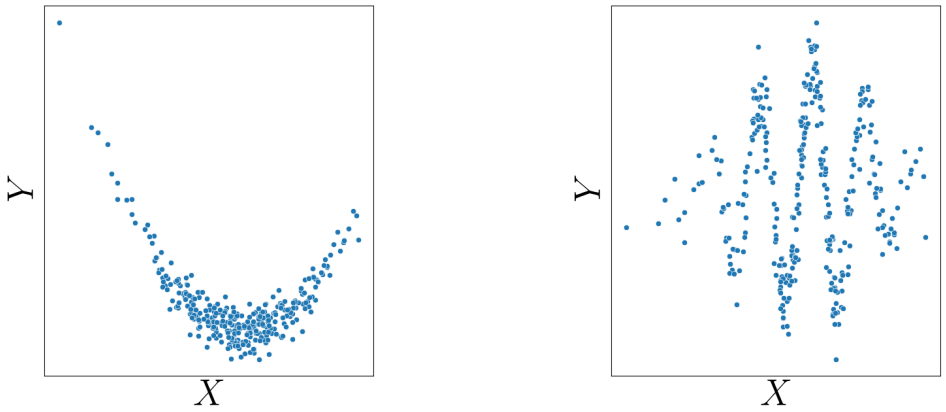


Figure 6. bd-HSIC(NCE-p^*): bd-HSIC(TRE-p^*): bd-HSIC(random uniform): bd-HSIC(RuLSIF): bd-HSIC(true weights): PDS: bd-CME:
Continuous treatment results. We find that RuLSIF has incorrect size under the null and that uniform weights has less power. NCE-p^* seems to have the best power while being calibrated under the null. We generally note that random uniform weights have less or no power when compared to "true weights" and estimated weights.

Non-linear dependency between X and Y Here we simulate data for $(d_z, d_x, d_y) \in \{(1,1,1),(50,3,3)\}$ using the same parameters as in the linear case. The only difference now is that we consider a non-linear dependency between X and Y illustrated in Figure 7. Figure 7a illustrates a U-shaped dependency between X and Y, which can be found in relationships between happiness vs. age (Kostyshak, 2017), and BMI vs. fragility (Watanabe



(A) U-shaped dependency, $Y = X^2\beta_{XY}$

(B) General symmetric non-linear dependency, $Y = \exp(-0.1X^2)\cos(X\pi)\beta_{XY}$

FIGURE 7. Samples from p^* under H_1 where the dependency is non-linear.

et al., 2020) to name a few. The U-shaped dependency can be generalized to symmetric non-linear relationships between X and Y illustrated in Figure 7b. We show the results in Figure 8 and Figure 9. We note that PDS has no power against the alternative when the dependency between X and Y is symmetric and non-linear, which is expected due to the linear nature of PDS.

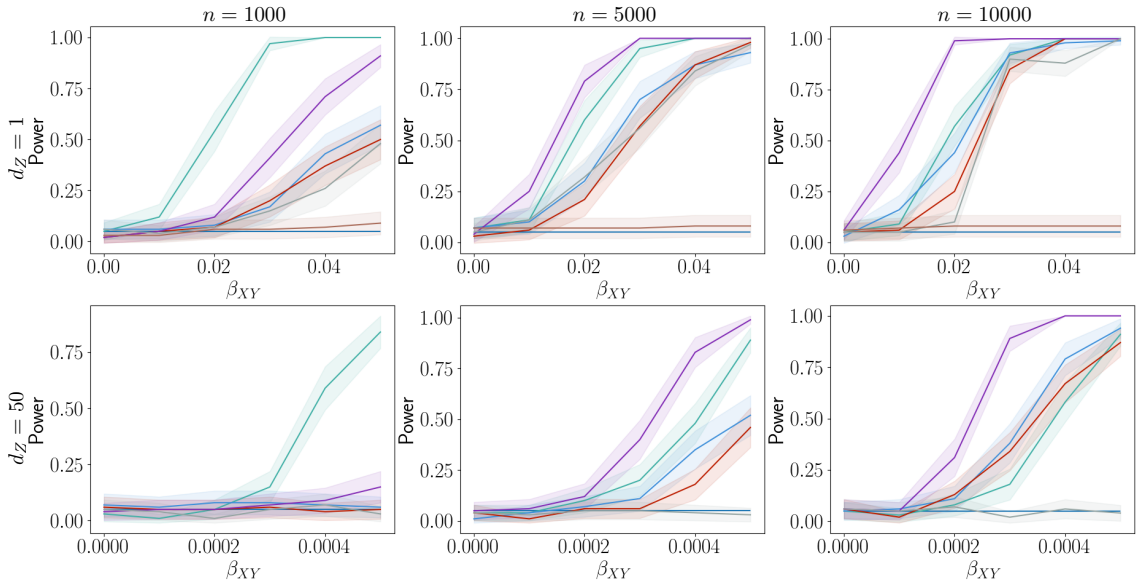


FIGURE 8. bd-HSIC(NCE-p^*): bd-HSIC(random uniform): bd-HSIC(RuLSIF): bd-HSIC(true weights): PDS: bd-CME:

Experiments for U-shaped dependency between X and Y.

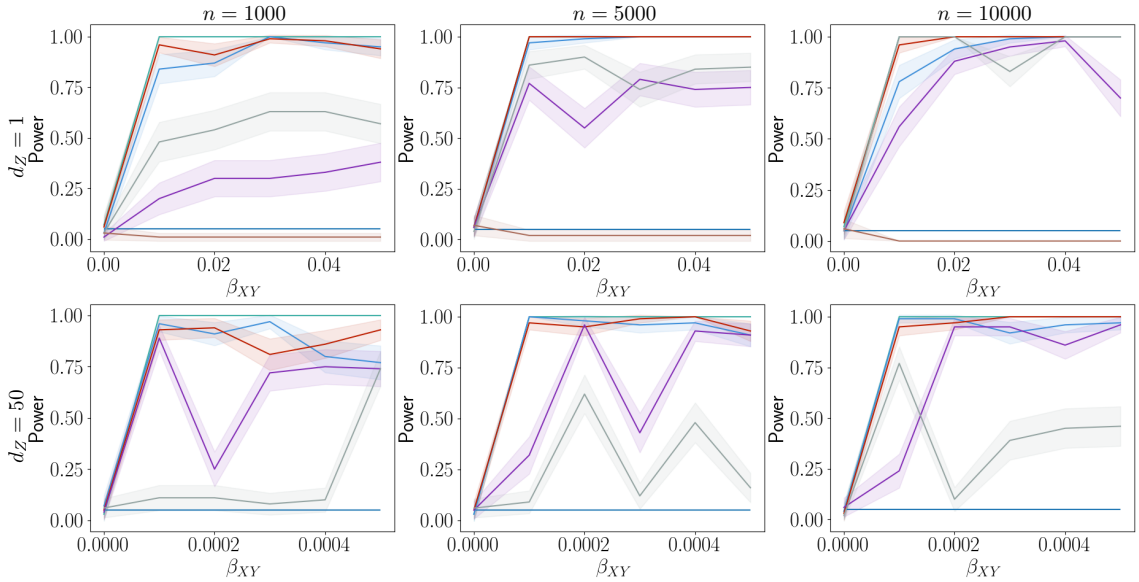


Figure 9. bd-HSIC(NCE-p^*): bd-HSIC(random uniform): bd-HSIC(RuLSIF): bd-HSIC(true weights): PDS: bd-CME:

Experiments for general non-linear symmetric dependency between X and Y.

5.1.3 Mixed treatment X

The data are simulated for $(d_z, d_x, d_y) \in \{(2,2,2),(4,4,3),(15,6,6),(50,8,8)\}$. We simulate the mixed data according to Algorithm F.3. Here we fix half of the X's to be continuous and the other half binary. We consider $\beta_{XY} \in [0.0, 0.1]$. We follow the same principles as in the continuous case when selecting $\theta, \phi, \beta_{XZ}, \beta_{YZ}$.

In our experiments, we compare against RuLSIF and randomly sampled uniform weights. We also compare between estimating the density ratio of the binary treatments separately (denoted with suffix "prod") and all treatments simultaneously (no suffix). The "mixed" suffix is a reference to the *dimension-wise mixing* proposed in Rhodes et al. (2020), which is applied when using TRE-p^*. We present the results in Figure 10. We note that bd-CME has an inflated type 1 error rate for $d_Z = 50$.

5.2 Pitfalls

Choice of kernels, a cautionary tale To improve the power of bd-HSIC, we could instead use a linear kernel. Figure 11 illustrates that bd-HSIC has comparable power to PDS, when using a linear kernel for testing the do-null when one considers the univariate binary treatment case.

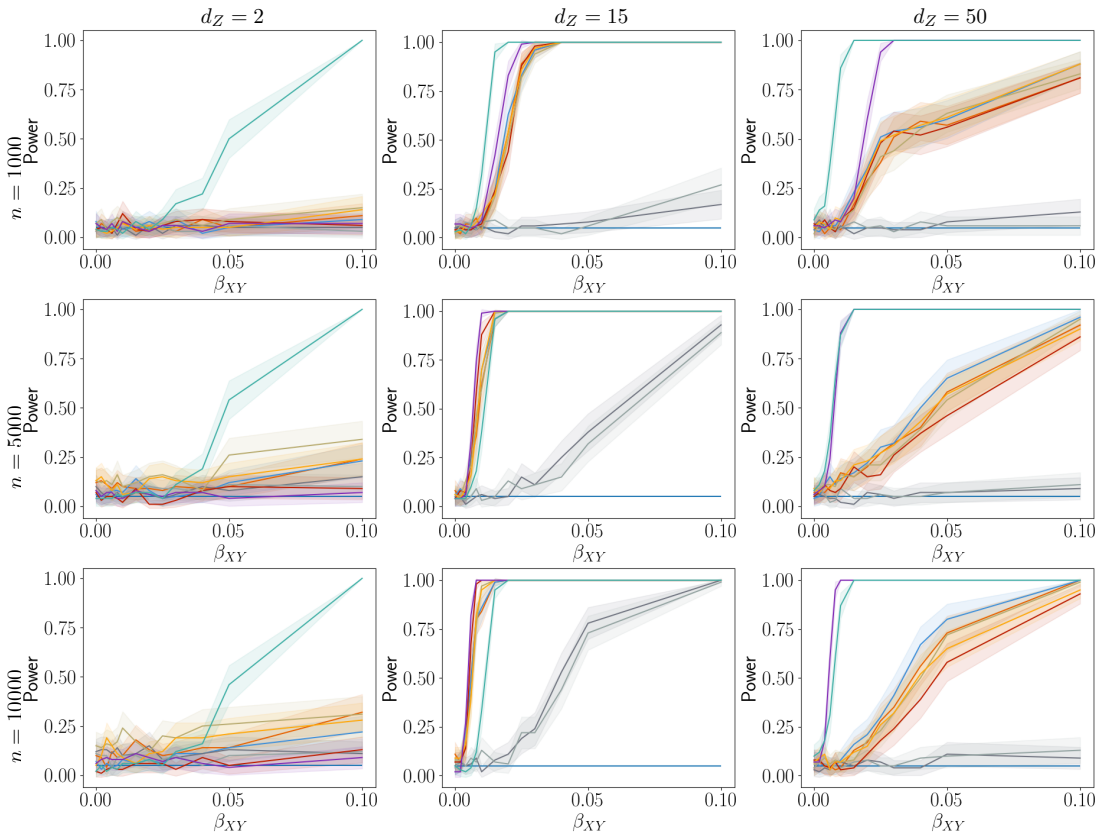


Figure 10. bd-HSIC(NCE-p^*, mixing): bd-HSIC(TRE-p^*, mixing): bd-HSIC(random uniform): bd-HSIC(RuLSIF): bd-HSIC(true weights): bd-HSIC(NCE-p^* prod): bd-HSIC(TRE-p^* product): bd-HSIC(RuLSIF product): bd-CME:
Mixed treatment results. We find that RuLSIF has an incorrect size under the null. TRE-p^* prod seems to have the best power while being calibrated under the null.

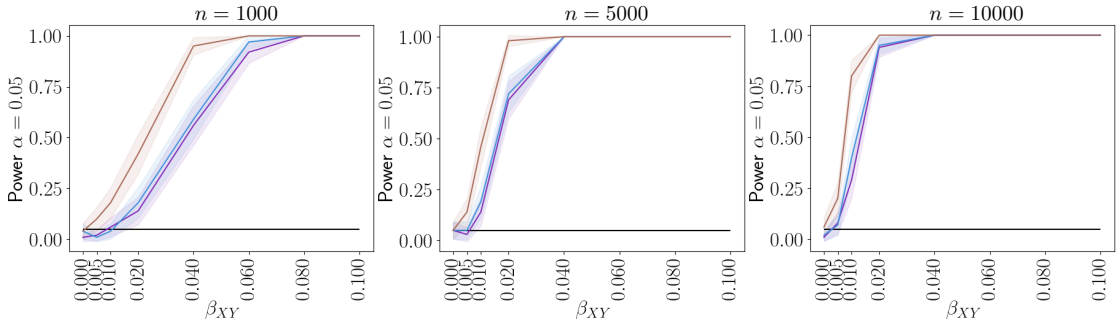
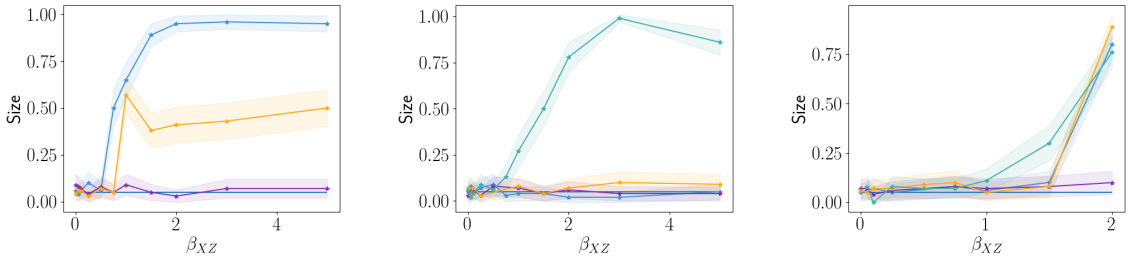


Figure 11. bd-HSIC(NCE-p^*): bd-HSIC(true weights): PDS: Binary treatment for $n = 1000, 5000, 10000$ using an linear kernel in bd-HSIC. Compared to Figure 5, bd-HSIC now has similar power to PDS.

However, when applying the linear kernel to univariate data for a continuous treatment the test becomes uncalibrated. In fact, the linear kernel makes bd-HSIC much more sensitive to confounding, exhibited in Figure 12a.

When does bd-HSIC break? We demonstrate a typical failure mode of bd-HSIC, when the value of β_{XZ} is so strong that the density ratio estimation fails. We show that NCE-p^* and TRE-p^* have incorrect size when β_{XZ} becomes large enough in Figure 12c. Here we generate data under the null for $\beta_{XZ} \in \{0.0, 0.05, 0.1, 0.15, 0.25, 0.5, 0.75, 1.0, 1.5, 2.0\}$ for $d_X = d_Y = 3, d_Z = 15$.



(A) Applying a linear kernel to univariate continuous treatment data. The linear kernel may cause the test to have incorrect size.

(B) Applying a RBF kernel to univariate continuous treatment data.

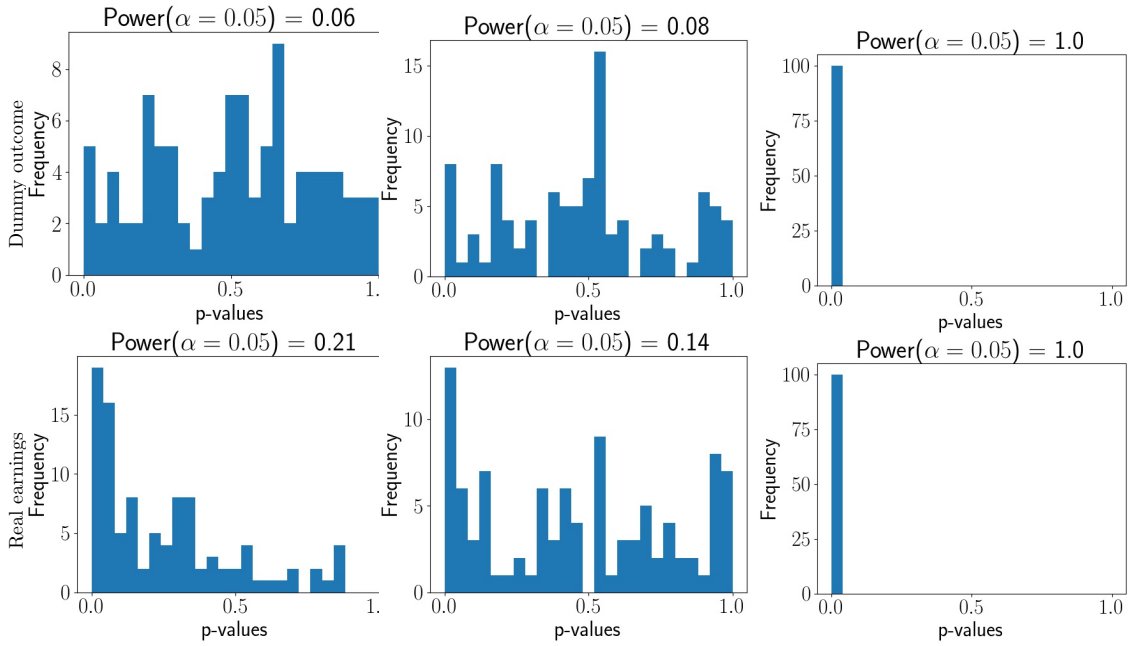
(C) bd-HSIC with estimated density ratios exhibits incorrect size for a certain amount of confounding. True weights are consistent.

Figure 12. bd-HSIC(NCE-p^*): bd-HSIC(TRE-p^*): bd-HSIC(true weights): bd-CME:

5.3 Experiments on real-world data

We apply bd-HSIC to two real-world data and compare against PDS and bd-CME.

Lalonde dataset experiments The Lalonde dataset comes from a study that looked at the effectiveness of a job training program (the treatment) on the real earnings of an

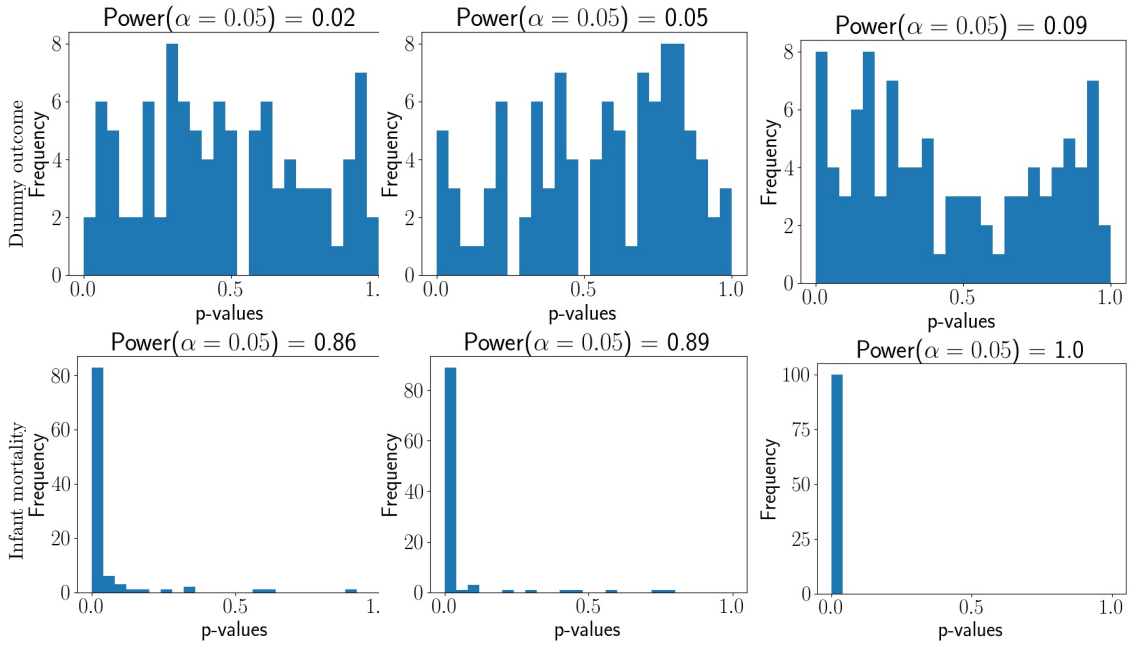


(A) PDS (B) bd-HSIC (TRE-p^*) (C) bd-CME

FIGURE 13. PDS suggests that there exists a causal association between receiving training and real earnings. bd-CME appears to have incorrect size.

individual, a couple of years after completion of the program (the outcome). Each individual has several descriptive covariates such as age, academic background, which confound the relationship between the treatment and the outcome. We compare the power between PDS and bd-HSIC on the Lalonde dataset in Figure 13. This is done by calculating the p-value on 100 bootstrap sampled subsets of the dataset. We further generate a random independent dummy outcome to verify that our tests are calibrated. We note that both PDS and bd-HSIC have the correct type 1 control for dummy outcome, while their power is comparable for the real earnings outcome.

Twins dataset experiments The twins dataset (Louizos et al., 2017) considers data of twin births in the US between 1989–1991. Here the treatment is being born the heavier twin and the outcome is mortality. Besides treatment and outcome, there are also descriptive confounders such as the smoking habits of the parents, education level of the parents, and medical risk factors of the children among many. For our experiments, we construct a slight variation of the experiment presented in Louizos et al. (2017), where we instead take the treatment to be $T = \left(\text{Weight}_{\text{heavier twin}}, \text{Weight}_{\text{lighter twin}}, \text{Weight}_{\text{heavier twin}} - \text{Weight}_{\text{lighter twin}}\right)$ and the outcome to be in the set $\{-1, 0, 1\}$, where -1 indicates that the lighter twin died, 1 that the heavier twin died, and 0 that either, neither or both died; everything else is kept the same. Similar to the Lalonde datasets we calculate the p-value on 100 bootstrap sampled subsets of the dataset and present results in Figure 14. Similar to the Lalonde dataset, we generated a random independent dummy outcome to verify that our tests are



(A) PDS (B) bd-HSIC (TRE-p^*) (C) bd-CME

FIGURE 14. All three methods suggest that there exists a causal association between weight and mortality.

calibrated. All three methods suggest there exists a causal association between infant weight and mortality.

6. Conclusion

We present a novel non-parametric method termed backdoor-HSIC (bd-HSIC), which is an importance-weighted covariance-based statistic to test the causal null hypothesis, or *do-null*. We first show that our proposed estimator for bd-HSIC is consistent. Experiments on a variety of synthetic datasets, including linear and non-linear dependencies, with different numbers of confounders and treatments, show that bd-HSIC is a flexible method with wider coverage of scenarios than parametric methods such as PDS. Finally, we compare bd-HSIC to PDS on two real-world datasets. Assuming a valid choice of confounders Z satisfying the back door criterion, the test evaluates whether there is any effect on the distribution of the outcome.

A major benefit of bd-HSIC is that can serve as a powerful tool in causal inference as it complements parametric methods such as PDS. For example, PDS generally has better power when the underlying dependency is linear but fails if the dependency is symmetrically non-linear. Even Double Machine Learning, proposed in Chernozhukov et al. (2018) and a generalization of PDS, will not straightforwardly capture such non-linear causal dependencies without further assumptions. This is inherent to the design of Double Machine Learning, which is based on semi-parametric partially linear models. bd-HSIC is fully non-

parametric and can be used as a general test for non-linear causal association, which is of broad interest to all statistically inclined sciences.

By combining with kernel conditional independence tests, this work could be extended to testing for null *conditional* average treatment effects, as well as to testing more general *nested* independence constraints (Richardson et al., 2017).

Appendix A. Remark on $p^*(y)$

Why do we have to consider $H_0 : p(y|do(x)) = p^*(y)$ instead of $H_0 : p(y|do(x)) = p(y)$? We show in the proposition below that we need arbitrary distributions $p^*(y)$ to formulate the hypothesis in a distributional sense since we may have that the do-null hypothesis holds but $p(y \mid do(x)) \neq p(y)$.

Proposition 7. $p(y|do(x)) = p(y) \quad \not\Longleftarrow \quad Y \perp X \mid do(X = x)$.

Proof. We give an example as proof. Consider a backdoor setting. Let X, Y, Z all be binary random variables, and assume that X, Z are Bernoulli distributed with $p = 0.5$. Then let's consider the following distribution:

$$\begin{aligned} p(Y = 1 \mid X = 1, Z = 1) &= p + \varepsilon \\ p(Y = 1 \mid X = 0, Z = 1) &= p \\ p(Y = 1 \mid X = 1, Z = 0) &= p - \varepsilon \\ p(Y = 1 \mid X = 0, Z = 0) &= p \end{aligned} \tag{9}$$

Calculating $p(Y = 1 \mid do(X = x)) = \sum_Z p(Y = 1 \mid X, Z)p(Z) = p$ for both $X = 0, 1$. Now let $p(X = 1 \mid Z) = \frac{1}{4} + \frac{Z}{2}$. But then the marginal

$$\begin{aligned} p(Y = 1) &= \sum_{X,Z} p(Y = 1 \mid X, Z)p(X \mid Z)p(Z) \\ &= 0.5(0.75(p + \varepsilon) + 0.25p + 0.25(p - \varepsilon) + 0.75p) = p + \frac{\varepsilon}{4} \end{aligned} \tag{10}$$

Thus we may have that the do-null hypothesis holds but $p(y \mid do(x)) \neq p(y)$. □

Appendix B. Consistency proofs

B.1 Proof of Theorem 1

Proof We first define the kernel mean embeddings used in our proposed estimator:

$$\begin{aligned} \mathbb{E}[W_{p^*} k(x, \cdot) l(y, \cdot)] = \mu_{xy}(\cdot) &= \int W_{p^*} k(x, \cdot) l(y, \cdot) \, d\mathbb{P}_{\mathcal{X}\otimes\mathcal{Y}\otimes\mathcal{X}}(x, y, z) \\ &= \int \frac{p^*(x)}{p(x \mid z)} k(x, \cdot) l(y, \cdot) \, d\mathbb{P}_{\mathcal{X}\otimes\mathcal{Y}\otimes\mathcal{X}}(x, y, z) \end{aligned}$$

$$\mathbb{E}[k(X^{p^*}, \cdot)] = \mu_{X^{p^*}}(\cdot) = \int k(X^{p^*}, \cdot) \, d\mathbb{P}^*(x)$$

$$\mathbb{E}[W_{p^*} l(y, \cdot)] = \mu_y(\cdot) = \int \frac{p^*(x)}{p(x \mid z)} l(y, \cdot) d\mathbb{P}_{\mathcal{X}\otimes\mathcal{Y}\otimes\mathcal{X}}(x, y, z).$$

Then we write the following for shorthand:

$$C_{p^*(y|x)} = \mu_{xy}(\cdot) - \mu_{X^{p^*}}(\cdot) \otimes \mu_y(\cdot)$$

Thus

$$\mathbb{E}\left[\| C_{p^*(y|x)} - \hat{C}_{p^*(y|x)} \|^2_{\mathrm{HS}} \right]$$

Which is

$$\mathbb{E}\Bigg[\Bigg\|\mu_{xy}(\cdot)-\mu_{X^{p^*}}(\cdot)\otimes\mu_y(\cdot)- \Bigg(\underbrace{\frac{1}{n}\sum_{i=1}^{n}\frac{p^*(x_i)}{p(x_i\,|\,z_i)}k(\cdot,x_i)\otimes l(\cdot,y_i)}_{\hat{\mu}_{xy}}-\underbrace{\left(\frac{1}{n}\sum_{j=1}^{n}k\left(\cdot,x_j^{p^*}\right)\right)}_{\hat{\mu}_{X^{p^*}}}\otimes\underbrace{\left(\frac{1}{n}\sum_{i=1}^{n}\frac{p^*(x_i)}{p(x_i\,|\,z_i)}l\left(\cdot,y_i\right)\right)}_{\hat{\mu}_y}\Bigg)\Bigg\|^2\Bigg].$$

Note that $w_i=\frac{p^*(x_i)}{p(x_i|z_i)}$, and for now we consider we have access to the true weights. Rearranging terms we get

$$\mathbb{E}\left[\left\|\underbrace{\mu_{xy}(\cdot)-\hat{\mu}_{xy}(\cdot)}_{A}+\underbrace{(\hat{\mu}_{X^{p^*}}(\cdot)\otimes\hat{\mu}_y(\cdot)-\mu_{X^{p^*}}(\cdot)\otimes\mu_y(\cdot))}_{B}\right\|^2\right]=\mathbb{E}\left[\langle A,A\rangle+2\langle A,B\rangle+\langle B,B\rangle\right].$$

The proof strategy is to obtain convergence rates for each term. First term $\langle A,A\rangle$:

$$\mathbb{E}\left[\langle A,A\rangle\right]=\mathbb{E}\left[\langle\mu_{xy}(\cdot),\mu_{xy}(\cdot)\rangle\right]-2\mathbb{E}\left[\langle\mu_{xy}(\cdot),\hat{\mu}_{xy}(\cdot)\rangle\right]+\mathbb{E}\left[\langle\hat{\mu}_{xy}(\cdot),\hat{\mu}_{xy}(\cdot)\rangle\right]$$

Let $x',y',z'\sim p$ be an independent copy of $x,y,z\sim p$. The first term is then

$$\mathbb{E}\left[\langle\mu_{xy}(\cdot),\mu_{x'y'}(\cdot)\rangle\right]=\mathbb{E}_{x,y,z,x',y',z'}\left[\frac{p^*(x)}{p(x\,|\,z)}\frac{p^*(x')}{p(x'\,|\,z')}k(x,x')l(y,y')\right],$$

the second one:

$$\begin{aligned}\mathbb{E}\left[\langle\mu_{xy}(\cdot),\hat{\mu}_{xy}(\cdot)\rangle\right]&=\mathbb{E}\left[\frac{1}{n}\sum_{i=1}^{n}\frac{p^*(x_i)}{p(x_i\,|\,z_i)}\mathbb{E}_{x,y,z}\left[\frac{p^*(x)}{p(x\,|\,z)}k(x,x_i)l(y,y_i)\right]\right]\\&=\mathbb{E}_{x',y',z'}\left[\frac{p^*(x')}{p(x'\,|\,z')}\mathbb{E}_{x,y,z}[\frac{p^*(x)}{p(x\,|\,z)}k(x,x')l(y,y')]\right]\\&=\mathbb{E}_{x,y,z,x',y',z'}\left[\frac{p^*(x)}{p(x\,|\,z)}\frac{p^*(x')}{p(x'\,|\,z')}k(x,x')l(y,y')\right],\end{aligned}$$

and the final one:

$$\mathbb{E}\left[\frac{1}{n^2}\sum_{i,j}w_iw_jk(x_i,x_j)l(y_i,y_j)\right]=\underbrace{\mathbb{E}\left[\frac{1}{n^2}\sum_{i\neq j}w_iw_jk(x_i,x_j)l(y_i,y_j)\right]}_{(a)}+\underbrace{\mathbb{E}\left[\frac{1}{n^2}\sum_{i}w_i^2k(x_i,x_i)l(y_i,y_i)\right]}_{(b)}.$$

Then (a) is

$$(a)=\frac{(n-1)}{n}\mathbb{E}_{x,y,z,x',y',z'}\left[\frac{p^*(x)}{p(x\,|\,z)}\frac{p^*(x')}{p(x'\,|\,z')}k(x,x')l(y,y')\right].$$

(b):

$$
\begin{aligned}
(b) &= \frac{1}{n}\mathbb{E}\left[\left(\frac{p^*(x)}{p(x\mid z)}\right)^2 k(x,x)l(y,y)\right] \\
&\leq \frac{1}{n}\underbrace{\sup_{x\in X} k(x,x)\sup_{y\in Y} k(y,y)}_{\text{Assumed to be bounded by } C}\underbrace{\mathbb{E}\left[\left(\frac{p^*(x)}{p(x\mid z)}\right)^2\right]}_{\text{finite variance of density ratio, i.e. bounded by some constant } D} \\
&= \frac{1}{n}CD
\end{aligned}
$$

So for $\langle A, A\rangle$ the convergence rate is $\mathcal{O}(\frac{1}{n})$. $\langle A, B\rangle$:

$$
\begin{aligned}
\langle A, B\rangle = &\underbrace{\langle \mu_{xy}(\cdot), \hat{\mu}_{X^{p^*}}(\cdot)\otimes\hat{\mu}_y(\cdot)\rangle}_{(1)} + \underbrace{\langle \hat{\mu}_{xy}(\cdot), \mu_{X^{p^*}}(\cdot)\otimes\mu_y(\cdot)\rangle}_{(2)} \\
&- \underbrace{\langle \hat{\mu}_{xy}(\cdot), \hat{\mu}_{X^{p^*}}(\cdot)\otimes\hat{\mu}_y(\cdot)\rangle}_{(3)} - \underbrace{\langle \mu_{xy}(\cdot), \mu_{X^{p^*}}(\cdot)\otimes\mu_{y'}(\cdot)\rangle}_{(4)}
\end{aligned}
$$

Thus

$$
\begin{aligned}
(1) &= \mathbb{E}\left[\frac{1}{n^2}\sum_{i,j}\frac{p^*(x_j)}{p(x_j\mid z_j)}\mathbb{E}_{x,y,z}\left[\frac{p^*(x)}{p(x\mid z)}k(x,x_i^{p^*})l(y,y_j)\right]\right] \\
&= \mathbb{E}_{x',y',z'}\left[\frac{p^*(x')}{p(x'\mid z')}\mathbb{E}_{x,y,z}\left[\frac{p^*(x)}{p(x\mid z)}k(x,x_i^{p^*})l(y,y_j)\right]\right] \\
(2) &= \mathbb{E}\left[\frac{1}{n}\sum_{i}^{n}\frac{p^*(x_i)}{p(x_i\mid z_i)}\mathbb{E}_{x_q}[k(x_i,X^{p^*})]\mathbb{E}_{x,y,z}\left[\frac{p^*(x)}{p(x\mid z)}l(y,y_i)\right]\right] \\
&= \mathbb{E}_{x',y',z'}\left[\frac{p^*(x')}{p(x'\mid z')}\mathbb{E}_{x_q}[k(x',X^{p^*})]\mathbb{E}_{x,y,z}\left[\frac{p^*(x)}{p(x\mid z)}l(y,y')\right]\right]
\end{aligned}
$$

$$
\begin{aligned}
(3) &= \mathbb{E}\left[\frac{1}{n^3}\sum_{i,k,j}\frac{p^*(x_i)}{p(x_i\mid z_i)}\frac{p^*(x_j)}{p(x_j\mid z_j)}k(x_i,X_k^{p^*})l(y_i,y_j)\right] \\
&= \mathbb{E}\left[\frac{1}{n^3}\sum_{k,i\neq j}\frac{p^*(x_i)}{p(x_i\mid z_i)}\frac{p^*(x_j)}{p(x_j\mid z_j)}k(x_i,X_k^{p^*})l(y_i,y_j)\right] + \mathbb{E}\left[\frac{1}{n^3}\sum_{k,i=j}\left(\frac{p^*(x_i)}{p(x_i\mid z_i)}\right)^2 k(x_i,X_k^{p^*})l(y_i,y_i)\right] \\
&= \frac{(n-1)}{n}\mathbb{E}_{x',y',z'}\left[\frac{p^*(x')}{p(x'\mid z')}\mathbb{E}_{x,y,z}[\frac{p^*(x)}{p(x\mid z)}\mathbb{E}_{X^{p^*}}[k(x,X^{p^*})]l(y,y')]\right] \\
&+ \frac{1}{n}\mathbb{E}_{X^{p^*},x,y,z}\left[\left(\frac{p^*(x)}{p(x\mid z)}\right)^2 k(x,X^{p^*})l(y,y)\right]
\end{aligned}
$$

$$
(4) = \mathbb{E}_{x,y,z}\left[\frac{p^*(x)}{p(x\mid z)}\mathbb{E}_{x_q}[k(x,X^{p^*})]\mathbb{E}_{x',y',z'}\left[\frac{p^*(x')}{p(x'\mid z')}l(y,y')\right]\right]
$$

We can use the same argument as in $\langle A, A\rangle$, consequently $\langle A, B\rangle \propto \mathcal{O}(\frac{1}{n})$ Let $X^{p^*}{}', x', y', z'$ be independent copies of X^{p^*}, x, y, z. Then

$$\langle B, B\rangle = \underbrace{\langle \hat{\mu}_{X^{p^*}}(\cdot)\otimes\hat{\mu}_y(\cdot), \hat{\mu}_{X^{p^*}}(\cdot)\otimes\hat{\mu}_y(\cdot)\rangle}_{(1)} - \underbrace{2\langle \hat{\mu}_{X^{p^*}}(\cdot)\otimes\hat{\mu}_y(\cdot), \mu_{X^{p^*}}(\cdot)\otimes\mu_y(\cdot)\rangle}_{(2)} + \underbrace{\langle \mu_{X^{p^*}{}'}(\cdot)\otimes\mu_{y'}(\cdot), \mu_{X^{p^*}}(\cdot)\otimes\mu_y(\cdot)\rangle}_{(3)}$$

Where

(1)

$$= \mathbb{E}\left[\frac{1}{n^2}\sum_{u,v} k(X_u^{p^*}, X_v^{p^*})\frac{1}{n^2}\sum_{i,j}\frac{p^*(x_i)}{p(x_i\,|\,z_i)}\frac{p^*(x_j)}{p(x_j\,|\,z_j)}l(y_i,y_j)\right]$$

$$= \mathbb{E}\Bigg[\frac{1}{n^2}\left(\sum_{u\neq v} k(X_u^{p^*}, X_v^{p^*}) + \sum_{u=v} k(X_u^{p^*}, X_u^{p^*})\right)$$

$$\frac{1}{n^2}\left(\sum_{i\neq j}\frac{p^*(x_i)}{p(x_i\,|\,z_i)}\frac{p^*(x_j)}{p(x_j\,|\,z_j)}l(y_i,y_j) + \sum_{i=j}\left(\frac{p^*(x_i)}{p(x_i\,|\,z_i)}\right)^2 l(y_i,y_i)\right)\Bigg]$$

$$= \underbrace{\mathbb{E}\left[\frac{1}{n^4}\sum_{u\neq v} k(X_u^{p^*}, X_v^{p^*})\sum_{i\neq j}\frac{p^*(x_i)}{p(x_i\,|\,z_i)}\frac{p^*(x_j)}{p(x_j\,|\,z_j)}l(y_i,y_j)\right]}_{a} + \underbrace{\mathbb{E}\left[\frac{1}{n^4}\sum_{u=v} k(X_u^{p^*}, X_u^{p^*})\sum_{i=j}\left(\frac{p^*(x_i)}{p(x_i\,|\,z_i)}\right)^2 l(y_i,y_i)\right]}_{b}$$

$$+ \underbrace{\mathbb{E}\left[\frac{1}{n^4}\sum_{u\neq v} k(X_u^{p^*}, X_v^{p^*})\sum_{i=j}\left(\frac{p^*(x_i)}{p(x_i\,|\,z_i)}\right)^2 l(y_i,y_i)\right]}_{c} + \underbrace{\mathbb{E}\left[\frac{1}{n^4}\sum_{u=v} k(X_u^{p^*}, X_u^{p^*})\sum_{i\neq j}\frac{p^*(x_i)}{p(x_i\,|\,z_i)}\frac{p^*(x_j)}{p(x_j\,|\,z_j)}l(y_i,y_j)\right]}_{d}$$

Each term is then:

$$a = \mathbb{E}\left[\frac{1}{n^4}\sum_{u\neq v} k(X_u^{p^*}, X_v^{p^*})\sum_{i\neq j}\frac{p^*(x_i)}{p(x_i\mid z_i)}\frac{p^*(x_j)}{p(x_j\mid z_j)}l(y_i,y_j)\right]$$
$$= \frac{(n-1)^2}{n^2}\mathbb{E}_{X^{p^*},X^{p^{*\prime}}}\left[k(X^{p^*},X^{p^{*\prime}})\right]\mathbb{E}_{x,y,z,x',y',z'}\left[\frac{p^*(x')}{p(x'\mid z')}\frac{p^*(x)}{p(x\mid z)}l(y,y')\right]$$
$$b = \mathbb{E}\left[\frac{1}{n^4}\sum_{u=v} k(X_u^{p^*}, X_u^{p^*})\sum_{i=j}\left(\frac{p^*(x_i)}{p(x_i\mid z_i)}\right)^2 l(y_i,y_i)\right] = \frac{1}{n^2}\mathbb{E}_{X^{p^*}}\left[k(X^{p^*},X^{p^*})\right]\mathbb{E}_{x,y,z}\left[\left(\frac{p^*(x)}{p(x\mid z)}\right)^2 l(y,y)\right]$$
$$c = \mathbb{E}\left[\frac{1}{n^4}\sum_{u\neq v} k(X_u^{p^*}, X_v^{p^*})\sum_{i=j}\left(\frac{p^*(x_i)}{p(x_i\mid z_i)}\right)^2 l(y_i,y_i)\right]$$
$$= \frac{n-1}{n^2}\mathbb{E}_{X^{p^*},X^{p^{*\prime}}}\left[k(X^{p^*},X^{p^{*\prime}})\right]\mathbb{E}_{x,y,z}\left[\left(\frac{p^*(x)}{p(x\mid z)}\right)^2 l(y,y)\right]$$
$$d = \mathbb{E}\left[\frac{1}{n^4}\sum_{u=v} k(X_u^{p^*}, X_u^{p^*})\sum_{i\neq j}\frac{p^*(x_i)}{p(x_i\mid z_i)}\frac{p^*(x_j)}{p(x_j\mid z_j)}l(y_i,y_j)\right]$$
$$= \frac{n-1}{n^2}\mathbb{E}_{X^{p^*}}\left[k(X^{p^*},X^{p^*})\right]\mathbb{E}_{x,y,z,x',y',z'}\left[\frac{p^*(x')}{p(x'\mid z')}\frac{p^*(x)}{p(x\mid z)}l(y,y')\right]$$

$$(2) = \mathbb{E}\left[\frac{1}{n}\sum_i \mathbb{E}_{X^{p^*}}[k(X^{p^*},X_i^{p^*})]\frac{1}{n}\sum_j\frac{p^*(x_j)}{p(x_j\mid z_j)}\mathbb{E}_{x,y,z}[\frac{p^*(x)}{p(x\mid z)}l(y,y_j)]\right]$$
$$= \mathbb{E}_{X^{p^*},X^{p^{*\prime}}}\left[k(X^{p^*},X^{p^{*\prime}})\right]\mathbb{E}_{x,y,z,x',y',z'}\left[\frac{p^*(x)}{p(x\mid z)}\frac{p^*(x')}{p(x'\mid z')}l(y,y')\right]$$

$$(3) = \mathbb{E}_{X^{p^*},X^{p^{*\prime}}}\left[k(X^{p^*},X^{p^{*\prime}})\right]\mathbb{E}_{x,y,z,x',y',z'}\left[\frac{p^*(x)}{p(x\mid z)}\frac{p^*(x')}{p(x'\mid z')}l(y,y')\right]$$

We note that the $\mathbb{E}_{X^{p^*},X^{p^{*\prime}}}\left[k(X^{p^*},X^{p^{*\prime}})\right]\mathbb{E}_{x,y,z,x',y',z'}\left[\frac{p^*(x)}{p(x\mid z)}\frac{p^*(x')}{p(x'\mid z')}l(y,y')\right]$-terms collapse for $(1),(2),(3)$. Thus:

$$\langle B,B\rangle = \left(\frac{-2}{n}+\frac{1}{n^2}\right)\mathbb{E}_{X^{p^*},X^{p^{*\prime}}}\left[k(X^{p^*},X^{p^{*\prime}})\right]\mathbb{E}_{x,y,z,x',y',z'}\left[\frac{p^*(x)}{p(x\mid z)}\frac{p^*(x')}{p(x'\mid z')}l(y,y')\right]$$
$$+\frac{1}{n^2}\mathbb{E}_{X^{p^*}}\left[k(X^{p^*},X^{p^*})\right]\mathbb{E}_{x,y,z}\left[\left(\frac{p^*(x)}{p(x\mid z)}\right)^2 l(y,y)\right]$$
$$+\frac{n-1}{n^2}\mathbb{E}_{X^{p^*},X^{p^{*\prime}}}\left[k(X^{p^*},X^{p^{*\prime}})\right]\mathbb{E}_{x,y,z}\left[\left(\frac{p^*(x)}{p(x\mid z)}\right)^2 l(y,y)\right]$$
$$+\frac{n-1}{n^2}\mathbb{E}_{X^{p^*}}\left[k(X^{p^*},X^{p^*})\right]\mathbb{E}_{x,y,z,x',y',z'}\left[\frac{p^*(x')}{p(x'\mid z')}\frac{p^*(x)}{p(x\mid z)}l(y,y')\right]$$

It suffices now to upper bound all the remaining expectations. First note that $\mathbb{E}_{X^{p^*},X^{p^{*\prime}}}\left[k(X^{p^*},X^{p^{*\prime}})\right] \leq \sup_{X^{p^*},X^{p^{*\prime}}} k(X^{p^*},X^{p^{*\prime}}) \propto C_1$ and $\mathbb{E}_{X^{p^*}}\left[k(X^{p^*},X^{p^*})\right] \leq \sup_{X^{p^*}} k(X^{p^*},X^{p^*}) \propto C_2$.

Further $\mathbb{E}_{x,y,z,x',y',z'}\left[\frac{p^*(x')}{p(x'\mid z')}\frac{p^*(x)}{p(x\mid z)}l(y,y')\right] \leq \mathbb{E}_{x',y',z'}\left[\frac{p^*(x')}{p(x'\mid z')}\right]\mathbb{E}_{x,y,z}\left[\frac{p^*(x)}{p(x\mid z)}\right]\sup_{y,y'} l(y,y') = \sup_{y,y'} \propto C_3$. Finally $\mathbb{E}_{x,y,z}\left[\left(\frac{p^*(x)}{p(x\mid z)}\right)^2 l(y,y)\right] \leq C_4$ using the same arguments as before (finite variance of the density ratio). Then

$$
\begin{aligned}
\langle B,B\rangle &= \left(\frac{-2}{n}+\frac{1}{n^2}\right)\mathbb{E}_{X^{p^*},X^{p^{*\prime}}}\left[k(X^{p^*},X^{p^{*\prime}})\right]\mathbb{E}_{x,y,z,x',y',z'}\left[\frac{p^*(x)}{p(x\mid z)}\frac{p^*(x')}{p(x'\mid z')}l(y,y')\right]\\
&+\frac{1}{n^2}\mathbb{E}_{X^{p^*}}\left[k(X^{p^*},X^{p^*})\right]\mathbb{E}_{x,y,z}\left[\left(\frac{p^*(x)}{p(x\mid z)}\right)^2 l(y,y)\right]\\
&+\frac{n-1}{n^2}\mathbb{E}_{X^{p^*},X^{p^{*\prime}}}\left[k(X^{p^*},X^{p^{*\prime}})\right]\mathbb{E}_{x,y,z}\left[\left(\frac{p^*(x)}{p(x\mid z)}\right)^2 l(y,y)\right]\\
&+\frac{n-1}{n^2}\mathbb{E}_{X^{p^*}}\left[k(X^{p^*},X^{p^*})\right]\mathbb{E}_{x,y,z,x',y',z'}\left[\frac{p^*(x')}{p(x'\mid z')}\frac{p^*(x)}{p(x\mid z)}l(y,y')\right]\\
&\leq \left(\frac{-2}{n}+\frac{1}{n^2}\right)C_1C_3+\frac{1}{n^2}C_2C_4+\frac{n-1}{n^2}C_1C_4+\frac{n-1}{n^2}C_2C_3 \propto \mathcal{O}(\frac{1}{n})
\end{aligned}
$$

As $\hat{C}_{p^*(y|x)}$ is asymptotically unbiased in L^2 norm, it follows from Chebyshev's inequality that it is a consistent estimator. ■

B.2 Proof for Theorem 2

Proof Again consider:

$$\mathbb{E}\left[\|C_{p^*(y|x)}-\hat{C}_{p^*(y|x)}\|^2_{\mathrm{HS}}\right]$$

However we take the estimator to be:

$$\hat{C}_{p^*(y|x)} = \frac{1}{n}\sum_{i=1}^n \hat{h}_n(x_i,z_i)k(\cdot,x_i)\otimes l(\cdot,y_i) - \left(\frac{1}{n}\sum_{j=1}^n k\left(\cdot,x_j^{p^*}\right)\right)\otimes\left(\frac{1}{n}\sum_{i=1}^n \hat{h}_n(x_i,z_i)l(\cdot,y_i)\right).$$

Then we have

$$
\begin{aligned}
&\mathbb{E}\Bigg[\Bigg\|\mu_{xy}(\cdot)-\mu_{X^{p^*}}(\cdot)\otimes\mu_y(\cdot)-\\
&\Bigg(\underbrace{\frac{1}{n}\sum_{i=1}^n \hat{h}_n(x_i,z_i)k(\cdot,x_i)\otimes l(\cdot,y_i)}_{\hat{\mu}_{xy}} - \underbrace{\left(\frac{1}{n}\sum_{j=1}^n k\left(\cdot,x_j^{p^*}\right)\right)}_{\hat{\mu}_{X^{p^*}}}\otimes\underbrace{\left(\frac{1}{n}\sum_{i=1}^n \hat{h}_n(x_i,z_i)l(\cdot,y_i)\right)}_{\hat{\mu}_y}\Bigg)\Bigg\|^2\Bigg]
\end{aligned}
$$

We follow the same steps as in Theorem 1.

$$\mathbb{E}\left[\left\|\underbrace{\mu_{xy}(\cdot)-\hat{\mu}_{xy}(\cdot)}_{A}+\underbrace{(\hat{\mu}_{X^{p^*}}(\cdot)\otimes\hat{\mu}_y(\cdot)-\mu_{X^{p^*}}(\cdot)\otimes\mu_y(\cdot))}_{B}\right\|^2\right]=\mathbb{E}\left[\langle A,A\rangle+2\langle A,B\rangle+\langle B,B\rangle\right]$$

First term $\langle A,A\rangle$:

$$\mathbb{E}\left[\langle A,A\rangle\right]=\mathbb{E}\left[\langle\mu_{xy}(\cdot),\mu_{xy}(\cdot)\rangle\right]-2\mathbb{E}\left[\langle\mu_{xy}(\cdot),\hat{\mu}_{xy}(\cdot)\rangle\right]+\mathbb{E}\left[\langle\hat{\mu}_{xy}(\cdot),\hat{\mu}_{xy}(\cdot)\rangle\right]$$

Let $(x',y',z')\sim p$ be an independent copy of $(x,y,z)\sim p$. Then the first term above is:

$$\mathbb{E}\left[\langle\mu_{xy}(\cdot),\mu_{x'y'}(\cdot)\rangle\right]=\mathbb{E}_{x,y,z,x',y',z'}\left[\frac{p^*(x)}{p(x\mid z)}\frac{p^*(x')}{p(x'\mid z')}k(x,x')l(y,y')\right];$$

the second one is:

$$\begin{aligned}\mathbb{E}\left[\langle\mu_{xy}(\cdot),\hat{\mu}_{xy}(\cdot)\rangle\right]&=\mathbb{E}\left[\frac{1}{n}\sum_{i=1}^{n}\hat{h}_n(x_i,z_i)\mathbb{E}_{x,y,z}\left[\frac{p^*(x)}{p(x\mid z)}k(x,x_i)l(y,y_i)\right]\right]\\&=\mathbb{E}_{x,y,z,x',y',z'}\left[\hat{h}_n(x,z)\frac{p^*(x')}{p(x'\mid z')}k(x,x')l(y,y')\right]\\&=(1+\mathcal{O}(\frac{1}{n^\alpha}))\mathbb{E}_{x,y,z,x',y',z'}\left[\frac{p^*(x)}{p(x\mid z)}\frac{p^*(x')}{p(x'\mid z')}k(x,x')l(y,y')\right];\end{aligned}$$

and the last term is:

$$\begin{aligned}&\mathbb{E}\left[\frac{1}{n^2}\sum_{i,j}\hat{h}_n(x_i,z_i)\hat{h}_n(x_j,z_j)k(x_i,x_j)l(y_i,y_j)\right]\\&=\underbrace{\mathbb{E}\left[\frac{1}{n^2}\sum_{i\neq j}\hat{h}_n(x_i,z_i)\hat{h}_n(x_j,z_j)k(x_i,x_j)l(y_i,y_j)\right]}_{(a)}+\underbrace{\mathbb{E}\left[\frac{1}{n^2}\sum_{i}\hat{h}_n(x_i,z_i)^2k(x_i,x_i)l(y_i,y_i)\right]}_{(b)}.\end{aligned}$$

We bound (a):

$$\begin{aligned}(a)&=\frac{(n-1)}{n}\mathbb{E}_{x,y,z,x',y',z'}\left[\hat{h}_n(x,z)\hat{h}_n(x',z')k(x,x')l(y,y')\right]\\&=\frac{(n-1)}{n}\mathbb{E}_{x,y,z,x',y',z'}\left[\frac{p^*(x)}{p(x\mid z)}\frac{p^*(x')}{p(x'\mid z')}k(x,x')l(y,y')\right]\left(1+2\mathcal{O}(\frac{1}{n^\alpha})+\mathcal{O}\left(\frac{1}{n^{2\alpha}}\right)\right)\end{aligned}$$

then (b)

$$\begin{aligned}(b)&=\frac{1}{n}\mathbb{E}\left[\left(\hat{h}_n(x,z)\right)^2k(x,x)l(y,y)\right]\\&\leq\frac{1}{n}\underbrace{\sup_{x\in X}k(x,x)\sup_{y\in Y}k(y,y)}_{\text{Assumed to be bounded by } C}\underbrace{\mathbb{E}\left[\left(\hat{h}_n(x,z)\right)^2\right]}_{\text{finite variance of density ratio, i.e. bounded by some } D}\\&=\frac{CD}{n}\left(1+2\mathcal{O}(\frac{1}{n^\alpha})+\mathcal{O}\left(\frac{1}{n^{2\alpha}}\right)\right)\end{aligned}$$

So for $\langle A, A\rangle$ the convergence rate is $\mathcal{O}\left(\frac{1}{n^{\min(1,\alpha)}}\right)$, since that is the slowest decaying term. For the $\langle A, B\rangle$ part we have:

$$\begin{aligned}\langle A, B\rangle = &\underbrace{\langle \mu_{xy}(\cdot), \hat{\mu}_{X^{p^*}}(\cdot)\otimes\hat{\mu}_y(\cdot)\rangle}_{(1)} + \underbrace{\langle \hat{\mu}_{xy}(\cdot), \mu_{X^{p^*}}(\cdot)\otimes\mu_y(\cdot)\rangle}_{(2)} \\ &- \underbrace{\langle \hat{\mu}_{xy}(\cdot), \hat{\mu}_{X^{p^*}}(\cdot)\otimes\hat{\mu}_y(\cdot)\rangle}_{(3)} - \underbrace{\langle \mu_{xy}(\cdot), \mu_{X^{p^*}}(\cdot)\otimes\mu_{y'}(\cdot)\rangle}_{(4)}\end{aligned}$$

Each part can then be written:

$$\begin{aligned}(1) &= \mathbb{E}\left[\frac{1}{n^2}\sum_{i,j}\hat{h}_n(x_j, z_j)\mathbb{E}_{x,y,z}\left[\frac{p^*(x)}{p(x\mid z)}k(x, x_i^{p^*})l(y, y_j)\right]\right] \\ &= \mathbb{E}_{x',y',z'}\left[\hat{h}_n(x', z')\mathbb{E}_{x,y,z}\left[\frac{p^*(x)}{p(x\mid z)}k(x, x_i^{p^*})l(y, y_j)\right]\right] \\ &= \mathbb{E}_{x',y',z'}\left[\frac{p^*(x')}{p(x'\mid z')}\mathbb{E}_{x,y,z}\left[\frac{p^*(x)}{p(x\mid z)}k(x, x_i^{p^*})l(y, y_j)\right]\right]\left(1+\mathcal{O}\left(\frac{1}{n^\alpha}\right)\right)\end{aligned}$$

$$\begin{aligned}(2) &= \mathbb{E}\left[\frac{1}{n}\sum_{i}^{n}\hat{h}_n(x_i, z_i)\mathbb{E}_{x_q}[k(x_i, X^{p^*})]\mathbb{E}_{x,y,z}\left[\frac{p^*(x)}{p(x\mid z)}l(y, y_i)\right]\right] \\ &= \mathbb{E}_{x',y',z'}\left[\hat{h}_n(x', z')\mathbb{E}_{x_q}[k(x', X^{p^*})]\mathbb{E}_{x,y,z}\left[\frac{p^*(x)}{p(x\mid z)}l(y, y')\right]\right] \\ &= \mathbb{E}_{x',y',z'}\left[\frac{p^*(x')}{p(x'\mid z')}\mathbb{E}_{x_q}[k(x', X^{p^*})]\mathbb{E}_{x,y,z}\left[\frac{p^*(x)}{p(x\mid z)}l(y, y')\right]\right]\left(1+\mathcal{O}\left(\frac{1}{n^\alpha}\right)\right)\end{aligned}$$

$$\begin{aligned}(3) &= \mathbb{E}\left[\frac{1}{n^3}\sum_{i,k,j}\hat{h}_n(x_i, z_i)\hat{h}_n(x_j, z_j)k(x_i, X_k^{p^*})l(y_i, y_j)\right] \\ &= \mathbb{E}\left[\frac{1}{n^3}\sum_{k,i\neq j}\hat{h}_n(x_i, z_i)\hat{h}_n(x_j, z_j)k(x_i, X_k^{p^*})l(y_i, y_j)\right] + \mathbb{E}\left[\frac{1}{n^3}\sum_{k,i=j}\hat{h}_n(x_i, z_i)^2k(x_i, X_k^{p^*})l(y_i, y_i)\right] \\ &= \left(1+2\mathcal{O}\left(\frac{1}{n^\alpha}\right)+\mathcal{O}\left(\frac{1}{n^{2\alpha}}\right)\right)\frac{(n-1)}{n}\mathbb{E}_{x',y',z'}\left[\frac{p^*(x')}{p(x'\mid z')}\mathbb{E}_{x,y,z}\left[\frac{p^*(x)}{p(x\mid z)}\mathbb{E}_{X^{p^*}}[k(x, X^{p^*})]l(y, y')\right]\right] \\ &+ \left(1+2\mathcal{O}(\frac{1}{n^\alpha})+\mathcal{O}\left(\frac{1}{n^{2\alpha}}\right)\right)\frac{1}{n}\mathbb{E}_{X^{p^*},x,y,z}\left[\left(\frac{p^*(x)}{p(x\mid z)}\right)^2k(x, X^{p^*})l(y, y)\right].\end{aligned}$$

$$(4) = \mathbb{E}_{x,y,z}\left[\frac{p^*(x)}{p(x\mid z)}\mathbb{E}_{x_q}[k(x, X^{p^*})]\mathbb{E}_{x',y',z'}\left[\frac{p^*(x')}{p(x'\mid z')}l(y, y')\right]\right]$$

We can use the same arguments as in $\langle A, A\rangle$, consequently $\langle A, B\rangle = \mathcal{O}\left(\frac{1}{n^{\min(1,\alpha)}}\right)$. The final term $\langle B, B\rangle$ is

$$\langle B, B\rangle = \underbrace{\langle \hat{\mu}_{X^{p^*}}(\cdot)\otimes\hat{\mu}_y(\cdot), \hat{\mu}_{X^{p^*}}(\cdot)\otimes\hat{\mu}_y(\cdot)\rangle}_{(1)} - \underbrace{2\langle \hat{\mu}_{X^{p^*}}(\cdot)\otimes\hat{\mu}_y(\cdot), \mu_{X^{p^*}}(\cdot)\otimes\mu_y(\cdot)\rangle}_{(2)} + \underbrace{\langle \mu_{X^{p^*\prime}}(\cdot)\otimes\mu_{y'}(\cdot), \mu_{X^{p^*}}(\cdot)\otimes\mu_y(\cdot)\rangle}_{(3)},$$

and we have

$$\begin{aligned}
&(1) = \\
&\mathbb{E}\left[\frac{1}{n^2}\sum_{u,v} k(X_u^{p^*}, X_v^{p^*})\frac{1}{n^2}\sum_{i,j}\hat{h}_n(x_i,z_i)\hat{h}_n(x_j,z_j)l(y_i,y_j)\right] \\
&= \mathbb{E}\left[\frac{1}{n^2}\left(\sum_{u\neq v} k(X_u^{p^*}, X_v^{p^*}) + \sum_{u=v} k(X_u^{p^*}, X_u^{p^*})\right)\right. \\
&\left.\frac{1}{n^2}\left(\sum_{i\neq j}\hat{h}_n(x_i,z_i)\hat{h}_n(x_j,z_j)l(y_i,y_j) + \sum_{i=j}\hat{h}_n(x_i,z_i)^2 l(y_i,y_i)\right)\right] \\
&= \underbrace{\mathbb{E}\left[\frac{1}{n^4}\sum_{u\neq v} k(X_u^{p^*}, X_v^{p^*})\sum_{i\neq j}\hat{h}_n(x_i,z_i)\hat{h}_n(x_j,z_j)l(y_i,y_j)\right]}_{(a)} + \underbrace{\mathbb{E}\left[\frac{1}{n^4}\sum_{u=v} k(X_u^{p^*}, X_u^{p^*})\sum_{i=j}\hat{h}_n(x_i,z_i)^2 l(y_i,y_i)\right]}_{(b)} \\
&+ \underbrace{\mathbb{E}\left[\frac{1}{n^4}\sum_{u\neq v} k(X_u^{p^*}, X_v^{p^*})\sum_{i=j}\hat{h}_n(x_i,z_i)^2 l(y_i,y_i)\right]}_{(c)} + \underbrace{\mathbb{E}\left[\frac{1}{n^4}\sum_{u=v} k(X_u^{p^*}, X_u^{p^*})\sum_{i\neq j}\hat{h}_n(x_i,z_i)\hat{h}_n(x_j,z_j)l(y_i,y_j)\right]}_{(d)}
\end{aligned}$$

Each of these terms is:

$$
\begin{aligned}
(a) &= \mathbb{E}\left[\frac{1}{n^4}\sum_{u\neq v} k(X_u^{p^*}, X_v^{p^*})\sum_{i\neq j}\hat{h}_n(x_i,z_i)\hat{h}_n(x_j,z_j)l(y_i,y_j)\right] \\
&= \frac{(n-1)^2}{n^2}\mathbb{E}_{X^{p^*},X^{p^{*\prime}}}\left[k(X^{p^*},X^{p^{*\prime}})\right]\mathbb{E}_{x,y,z,x',y',z'}\left[\frac{p^*(x')}{p(x'\,|\,z')}\frac{p^*(x)}{p(x\,|\,z)}l(y,y')\right]\left(1+2\mathcal{O}\left(\frac{1}{n^\alpha}\right)+\mathcal{O}\left(\frac{1}{n^{2\alpha}}\right)\right) \\
(b) &= \mathbb{E}\left[\frac{1}{n^4}\sum_{u=v} k(X_u^{p^*}, X_u^{p^*})\sum_{i=j}\hat{h}_n(x_i,z_i)^2 l(y_i,y_i)\right] \\
&= \frac{1}{n^2}\mathbb{E}_{X^{p^*}}\left[k(X^{p^*},X^{p^*})\right]\mathbb{E}_{x,y,z}\left[\left(\frac{p^*(x)}{p(x\,|\,z)}\right)^2 l(y,y)\right]\left(1+2\mathcal{O}(\frac{1}{n^\alpha})+\mathcal{O}\left(\frac{1}{n^{2\alpha}}\right)\right) \\
(c) &= \mathbb{E}\left[\frac{1}{n^4}\sum_{u\neq v} k(X_u^{p^*}, X_v^{p^*})\sum_{i=j}\hat{h}_n(x_i,z_i)^2 l(y_i,y_i)\right] \\
&= \frac{n-1}{n^2}\mathbb{E}_{X^{p^*},X^{p^{*\prime}}}\left[k(X^{p^*},X^{p^{*\prime}})\right]\mathbb{E}_{x,y,z}\left[\left(\frac{p^*(x)}{p(x\,|\,z)}\right)^2 l(y,y)\right]\left(1+2\mathcal{O}(\frac{1}{n^\alpha})+\mathcal{O}\left(\frac{1}{n^{2\alpha}}\right)\right) \\
(d) &= \mathbb{E}\left[\frac{1}{n^4}\sum_{u=v} k(X_u^{p^*}, X_u^{p^*})\sum_{i\neq j}\hat{h}_n(x_i,z_i)\hat{h}_n(x_j,z_j)l(y_i,y_j)\right] \\
&= \frac{n-1}{n^2}\mathbb{E}_{X^{p^*}}\left[k(X^{p^*},X^{p^*})\right]\mathbb{E}_{x,y,z,x',y',z'}\left[\frac{p^*(x')}{p(x'\,|\,z')}\frac{p^*(x)}{p(x\,|\,z)}l(y,y')\right]\left(1+2\mathcal{O}\left(\frac{1}{n^\alpha}\right)+\mathcal{O}\left(\frac{1}{n^{2\alpha}}\right)\right).
\end{aligned}
$$

Returning to the expression for $\langle B, B\rangle$:

$$
\begin{aligned}
(2) &= \mathbb{E}\left[\frac{1}{n}\sum_i \mathbb{E}_{X^{p^*}}[k(X^{p^*},X_i^{p^*})]\frac{1}{n}\sum_j \hat{h}_n(x_j,z_j)\mathbb{E}_{x,y,z}[\frac{p^*(x)}{p(x\,|\,z)}l(y,y_j)]\right] \\
&= \mathbb{E}_{X^{p^*},X^{p^{*\prime}}}\left[k(X^{p^*},X^{p^{*\prime}})\right]\mathbb{E}_{x,y,z,x',y',z'}\left[\frac{p^*(x)}{p(x\,|\,z)}\frac{p^*(x')}{p(x'\,|\,z')}l(y,y')\right]\left(1+\mathcal{O}\left(\frac{1}{n^\alpha}\right)\right),
\end{aligned}
$$

and

$$
(3) = \mathbb{E}_{X^{p^*},X^{p^{*\prime}}}\left[k(X^{p^*},X^{p^{*\prime}})\right]\mathbb{E}_{x,y,z,x',y',z'}\left[\frac{p^*(x)}{p(x\,|\,z)}\frac{p^*(x')}{p(x'\,|\,z')}l(y,y')\right].
$$

We use the same arguments as in Theorem 1. We note that the the sums collapse similarly in (1) with some added negligible terms converging faster than $\mathcal{O}(\frac{1}{n})$. What remains is then the slowest converging term

$$
\mathbb{E}_{X^{p^*},X^{p^{*\prime}}}\left[k(X^{p^*},X^{p^{*\prime}})\right]\mathbb{E}_{x,y,z,x',y',z'}\left[\frac{p^*(x)}{p(x\,|\,z)}\frac{p^*(x')}{p(x'\,|\,z')}l(y,y')\right]\mathcal{O}\left(\frac{1}{n^\alpha}\right) = \mathcal{O}\left(\frac{1}{n^\alpha}\right)
$$

compared to $\mathcal{O}(\frac{1}{n})$. Hence $\langle B, B\rangle = \mathcal{O}\left(n^{-\min(1,\alpha)}\right)$. As $\langle A, A\rangle, \langle A, B\rangle, \langle B, B\rangle$ all have the same convergence rate of their bias terms, we conclude that $\mathbb{E}\left[\|C_{p^*(y|x)} - \hat{C}_{p^*(y|x)}\|^2_{\mathrm{HS}}\right] = \mathcal{O}\left(\frac{1}{n^{\min(1,\alpha)}}\right)$. ■

B.3 Proof of Proposition 6

Proof. Since $\psi(\|\hat{C}_{p^*(y|x)}\|^2_{\mathrm{HS}}) \geq 0$, it suffices by Markov's inequality to show that

$$\lim_{n\to\infty} \mathbb{E}\left[\psi(\|\hat{C}_{p^*(y|x)}\|^2_{\mathrm{HS}})\right] = 0.$$

Define $\mathbf{M}_k(n) := \{1,\ldots,n\}^k$ as the k-fold Cartesian product of the set $\{1,\ldots,n\}$. We then have

$$\begin{aligned}\psi(\|\hat{C}_{p^*(y|x)}\|^2_{\mathrm{HS}}) = &\underbrace{\frac{1}{n^2}\sum_{i,j\in\mathbf{M}_2(n)} k(x_i,x_j)l(y_{\psi(i)},y_{\psi(j)})w_iw_j}_{A_n}\\ &+\underbrace{\frac{1}{n^4}\sum_{i,j,k,l\in\mathbf{M}_4(n)} k(x_k,x_l)l(y_{\psi(i)},y_{\psi(j)})w_iw_j}_{B_n}\\ &-2\underbrace{\frac{1}{n^3}\sum_{i,j,k\in\mathbf{M}_3(n)} w_ik(x_i,x_j)w_kl(y_{\psi(i)},y_{\psi(k)})}_{C_n},\end{aligned} \tag{11}$$

which is abbreviated as $A_n + B_n - 2C_n$. It suffices to show that $\lim_n \mathbb{E}A_n = \lim_n \mathbb{E}B_n = \lim_n \mathbb{E}C_n = \zeta$. Where

$$\zeta = \mathbb{E}\left[k(X,X')\right]\mathbb{E}\left[l(Y,Y')\right]$$

where X', Z' is an identical independent copy of X, Z respectively. We employ the same strategy as in Rindt et al. (2021) and partition the summing over the indices as

$$A_n = \frac{1}{n^2}\sum_{i,j\in U(2,\psi,n)} k(x_i,x_j)l(y_{\psi(i)},y_{\psi(j)})w_iw_j + \frac{1}{n^2}\sum_{i,j\in R(2,\psi,n)} k(x_i,x_j)l(y_{\psi(i)},y_{\psi(j)})w_iw_j \tag{12}$$

with

$$U(2,\psi,n) := \{(i,j)\in M_2(n) : (i,j,\psi(i),\psi(j)) \text{ are 4 distinct elements}\}$$

and

$$R(2,\psi,n) := M_2(n)\backslash U(2,\psi,n).$$

Here 4 distinct elements simply means that $i, j, \psi(i), \psi(j)$ are all different, i.e. 1,2,3,4. The main observation here is that as $n \to \infty$, almost all terms in sums A_n, B_n, C_n will have distinct terms. We then take

$$
\begin{aligned}
\mathbb{E}\left[A_n\right] =& \mathbb{E}\left[\mathbb{E}\left[A_n \mid \psi\right]\right] \\
=& \mathbb{E}\left[\frac{1}{n^2}\sum_{U(2,\psi,n)} \mathbb{E}\left[k(x_i,x_j)l(y_{\psi(i)},y_{\psi(j)})w_iw_j \mid \psi\right]\right] \\
&+ \mathbb{E}\left[\frac{1}{n^2}\sum_{R(2,\psi,n)} \mathbb{E}\left[k(x_i,x_j)l(y_{\psi(i)},y_{\psi(j)})w_iw_j \mid \psi\right]\right] \\
=& \mathbb{E}\left[\frac{|U(2,\psi,n)|}{n^2}\right]\mathbb{E}\left[k(X,X')\right]\mathbb{E}\left[l(Y,Y')\right] + \mathbb{E}\left[\frac{|R(2,\psi,n)|}{n^2}\right]\mathcal{O}(1) \to \zeta.
\end{aligned}
\tag{13}
$$

Since the permutation occurs within $p(x \mid z)$, there will always be a dependency between w_i, w_j and $l(y_{\psi(i)}, y_{\psi(j)})$ meaning that distinct indices don't factorize directly unlike the setting in Rindt et al. (2021). However, by calculating the expectation we have that

$$
\begin{aligned}
&\mathbb{E}\left[k(x_i,x_j)l(y_{\psi(i)},y_{\psi(j)})w_iw_j \mid \psi\right] \\
&= \mathbb{E}\left[k(X,X')l(Y,Y')WW'\right] \\
&= \int k(x,x')l(y,y')\frac{p^*(x)}{p(x \mid z)}\frac{p^*(x')}{p(x' \mid z')}p(x,y,z)p(x',y',z')dxdydzdx'dy'dz' \\
&= \int k(x,x')l(y,y')p^*(x)p^*(x')\underbrace{\left(\int p(y \mid x,z)p(z)dz\right)\left(\int p(y' \mid x',z')p(z')dz'\right)}_{y \text{ and } x \text{ are rendered independent by having distinct indices}}dxdydx'dy' \\
&= \int k(x,x')l(y,y')p^*(x)p^*(x')p(y)p(y')dxdx'dydy' \\
&= \mathbb{E}\left[k(X,X')\right]\mathbb{E}\left[l(Y,Y')\right] = \zeta.
\end{aligned}
\tag{14}
$$

Now we can repeat the argument in Rindt et al. (2021) with

$$
\begin{aligned}
\frac{\mathbb{E}|U(2,\psi,n)|}{n^2} &= \frac{n(n-1)}{n^2}\cdot\mathbb{P}\left((i,j,\psi(i),\psi(j)) \text{ are 4 distinct elements }\right) \\
&= \frac{n(n-1)}{n^2}\frac{\binom{n-2}{2}\binom{n-4}{2}\cdots\binom{n-2d+2}{2}}{\binom{n}{2}\binom{n}{2}\cdots\binom{n}{2}} \\
&\longrightarrow 1.
\end{aligned}
\tag{15}
$$

Hence $\lim_{n\to\infty}\mathbb{E}\left(A_n\right) = \mathbb{E}\left[k(X,X')\right]\mathbb{E}\left[l(Y,Y')\right] = \zeta$. We can repeat the argument for B_n and C_n and hence the sum of ζ's collapse into 0. Throughout the proof, we have assumed that $\mathbb{E}\left[k(X,X')l(Y,Y')WW'\right] < C$, where C is some constant.

□

Appendix C. Proof of Proposition 2

We first establish some "RKHS calculus" before we proceed with the calculations.

Mean Some rules following Riesz representation theorem and RKHS spaces.

1. $\langle \mu_x, f \rangle_{\mathcal{F}} = \mathbf{E}_x[\langle \phi(x), f \rangle_{\mathcal{F}}] = \mathbf{E}_x[f(x)]$
2. $\langle \mu_y, g \rangle_{\mathcal{G}} = \mathbf{E}_y[\langle \psi(y), g \rangle_{\mathcal{G}}] = \mathbf{E}_y[g(x)]$
3. $\|\mu_x\|^2_{\mathcal{F}} = \mathbf{E}_{x,x'}[\langle \phi(x), \phi(x') \rangle_{\mathcal{F}}] = \mathbf{E}_{x,x'}[k(x,x')] = \frac{1}{n^2} \sum_{i,j} k(x_i, x_j)$

Tensor operator $\otimes$ We may employ any $f \in \mathcal{F}$ and $g \in \mathcal{G}$ to define a tensor product operator $f \otimes g : \mathcal{G} \to \mathcal{F}$ as follows: $(f \otimes g)h := f\langle g, h \rangle_{\mathcal{G}}$ for all $h \in \mathcal{G}$

Lemma 1. *For any $f_1, f_2 \in \mathcal{F}$ and $g_1, g_2 \in \mathcal{G}$ the following equation holds:* $\langle f_1 \otimes g_1, f_2 \otimes g_2 \rangle_{\mathrm{HS}} = \langle f_1, f_2 \rangle_F \langle g_1, g_2 \rangle_G$

Using this lemma, one can simply show the norm of $f \otimes g$ equals $\|f \otimes g\|^2_{\mathrm{HS}} = \|f\|^2_{\mathcal{F}} \|g\|^2_{\mathcal{G}}$

Let $\hat{\mu}_{P_x} = \frac{1}{n} \sum_{i=1}^n k(\mathbf{x}_i, \cdot)$ and $\hat{\mu}_{P_y} = \frac{1}{n} \sum_{i=1}^n l(\mathbf{y}_i, \cdot)$. We are now ready to proceed with the derivation.

Proof For

$$\hat{C}_{p^*(y|x)} = \frac{1}{n} \sum_{i=1}^n \tilde{w}_i k(\cdot, x_i) \otimes l(\cdot, y_i) - \left(\frac{1}{m_x} \sum_{j=1}^{m_x} k\left(\cdot, x_j^{p^*}\right) \right) \otimes \left(\frac{1}{n} \sum_{i=1}^n \tilde{w}_i l\left(\cdot, y_i\right) \right).$$

we have that:

$$\|\hat{C}_{p^*(y|x)}\|^2 = A + B - 2C$$

where

$$A = \frac{1}{n^2} \sum_{i=1}^n \sum_{j=1}^n \tilde{w}_i \tilde{w}_j \left\langle k(\cdot, x_i) \otimes l(\cdot, y_i), k(\cdot, x_j) \otimes l(\cdot, y_j) \right\rangle = \tilde{w}^\top \left(K \circ L \right) \tilde{w} = tr\left(D_{\tilde{w}} K D_{\tilde{w}} L \right),$$

$$B = \frac{1}{m_x^2 n^2} \left\| \sum_{i=1}^n k\left(\cdot, x_i^{p^*}\right) \right\|^2 \left\| \sum_{i=1}^n \tilde{w}_i l\left(\cdot, y_i\right) \right\|^2 = \frac{1}{m_x^2 n^2} (\mathbf{K}^{p^*})_{++} (\mathbf{L} \circ \tilde{W})_{++}$$

$$\begin{aligned} C &= \frac{1}{n^2 m_x} \left\langle \sum_{i=1}^n \tilde{w}_i k(\cdot, x_i) \otimes l(\cdot, y_i), \left(\sum_{j=1}^{m_x} k\left(\cdot, x_j^{p^*}\right) \right) \otimes \left(\sum_{j=1}^n \tilde{w}_j l\left(\cdot, y_j\right) \right) \right\rangle \\ &= \frac{1}{n^2 m_x} \sum_{i=1}^n \tilde{w}_i \left(\sum_{j=1}^{m_x} k\left(x_i, x_j^{p^*}\right) \right) \left(\sum_{r=1}^n \tilde{w}_r l\left(y_i, y_r\right) \right) = \tilde{w}^\top \left(K^q 1_{m_x} \circ L\tilde{w} \right) = tr\left(D_{\tilde{w}} K^q 1_{m_x} \tilde{w}^\top L \right), \end{aligned}$$

It should be noted that we can choose the number of samples m_x for $x_i^{p^*} \sim p^*$ to use. For practical purposes we set $m_x = n$. ■

Appendix D. Optimal c_{p^*} Choice

D.1 Derivation of univariate c_{p^*}

Suppose that we wish to choose an 'optimal' c_{p^*} value for rescaling the distribution. One criterion for optimality is to maximize the effective sample size

$$\mathrm{ESS} := \frac{\left(\sum_i w_i\right)^2}{\sum_i w_i^2}$$

where $w_i = p^*\left(x;\phi^*\right)/p_{X|Z}(x;\phi)$ is the weight used to resample observations. This is essentially equivalent to minimizing the variance of the individual weights:

$$\arg\min_{\phi\cdot} \mathrm{Var}\, w\left(X_i, Z_i; \phi^*\right).$$

Note that

$$\begin{aligned}\mathbb{E}w\left(X_i, Z_i; \phi^*\right) &= \int \frac{p^*\left(x,z,\phi^*\right)}{p_{X|Z}(x,z,\phi)} p_{X|Z}(x,z,\phi)dxdz \\ &= \int p^*\left(x,z,\phi^*\right)dxdz \\ &= 1\end{aligned}$$

so this is equvalent to minimizing the squared expectation of $w()$. If we assume that everything is Gaussian, and that X, Z have standard normal marginal distributions with correlation ρ, this becomes equivalent to minimizing

$$\frac{1}{\tau^2}\mathbb{E}\left[\frac{\phi\left(\frac{X}{\tau}\right)}{\phi\left(\frac{X-\rho Z}{\sqrt{1-\rho^2}}\right)}\right]^2$$

with respect to τ. We can rewrite this expression as:

$$\begin{aligned}f(\tau) &= \frac{1}{2\pi\sqrt{1-\rho^2\tau^2}}\iint_{-\infty}^{\infty}\exp\left\{-\frac{1}{2}\begin{pmatrix}x\\z\end{pmatrix}^T\begin{pmatrix}\frac{2}{\tau^2}-\frac{1}{1-\rho^2} & \frac{\rho}{1-\rho^2}\\ \frac{\rho}{1-\rho^2} & \frac{1-2\rho^2}{1-\rho^2}\end{pmatrix}\begin{pmatrix}x\\z\end{pmatrix}\right\}dxdz \\ &= \frac{1}{\sqrt{1-\rho^2\tau^2}}\begin{vmatrix}\frac{2}{\tau^2}-\frac{1}{1-\rho^2} & \frac{\rho}{1-\rho^2}\\ \frac{\rho}{1-\rho^2} & \frac{1-2\rho^2}{1-\rho^2}\end{vmatrix}^{-1/2} \\ &= \frac{1}{\tau^2}\begin{vmatrix}\frac{2\left(1-\rho^2\right)}{\tau^2}-1 & \rho\\ \rho & 1-2\rho^2\end{vmatrix}^{-1/2}\end{aligned}$$

Note that minimizing f is the same as maximizing $1/f^2$, so we need to maximize

$$\begin{aligned}1/f(\tau)^2 &= \tau^4\begin{vmatrix}\frac{2\left(1-\rho^2\right)}{\tau^2}-1 & \rho\\ \rho & 1-2\rho^2\end{vmatrix} \\ &= \tau^4\left(\left(2\frac{1-\rho^2}{\tau^2}-1\right)\left(1-2\rho^2\right)-\rho^2\right)\right) \\ &= \tau^2 2\left(1-\rho^2\right)\left(1-2\rho^2\right)-\left(1-\rho^2\right)\tau^4.\end{aligned}$$

This is maximized at $\tau^2 = 1-2\rho^2$, or $\tau = \sqrt{1-2\rho^2}$.

D.2 Derivation of multivariate c_{p^*}

Now suppose that $(X, Z) \sim N_{p+p^*}(0, \Sigma)$ where we take $\Sigma = \begin{bmatrix} \Sigma_{xx} & \Sigma_{xz} \\ \Sigma_{zx} & \Sigma_{zz} \end{bmatrix}$ and assume that $\Sigma_{xx} = I_p$ and $\Sigma_{zz} = I_{p^*}$ (again, this can be achieved by rescaling). By the same reasoning as above, we want to minimize the squared expectation of the weights, which amounts to minimizing

$$f(\tau) =$$
$$\frac{1}{|T|} \iint_{-\infty}^{\infty} \exp\left\{ -\frac{1}{2} \begin{pmatrix} x \\ z \end{pmatrix}^T \begin{pmatrix} 2T^{-1} - (I_p - \Sigma_{xz}\Sigma_{zx})^{-1} & (I_p - \Sigma_{xz}\Sigma_{zx})^{-1} \sum_{xz} \\ \Sigma_{zx} (I_p - \Sigma_{xz}\Sigma_{zx})^{-1} & I_{p^*} - \Sigma_{zx} (I_p - \Sigma_{xz}\Sigma_{zx})^{-1} \Sigma_{xz} \end{pmatrix} \begin{pmatrix} x \\ z \end{pmatrix} \right\} dxdz$$
$$\propto \frac{1}{|T|} \begin{vmatrix} 2T^{-1} - (I_p - \Sigma_{xz}\Sigma_{zx})^{-1} & (I_p - \Sigma_{xz}\Sigma_{zx})^{-1} \Sigma_{xz} \\ \Sigma_{zx} (I_p - \Sigma_{xz}\Sigma_{zx})^{-1} & I_{p^*} - \Sigma_{zx} (I_p - \Sigma_{xz}\Sigma_{zx})^{-1} \Sigma_{xz} \end{vmatrix}^{-1/2}$$

or maximizing

$$|T|^2 \begin{vmatrix} 2T^{-1} - (I_p - \Sigma_{xz}\Sigma_{zx})^{-1} & (I_p - \Sigma_{xz}\Sigma_{zx})^{-1} \Sigma_{xz} \\ \sum_{zx} (I_p - \Sigma_{xz}\Sigma_{zx})^{-1} & I_{p^*} - \Sigma_{zx} (I_p - \Sigma_{xz}\Sigma_{zx})^{-1} \Sigma_{xz} \end{vmatrix}$$

Note that in this case we will choose a whole matrix T, rather than just a scaling constant, but we could simplify to assume that $T = c_{p^*} I_p$ for some scalar c_{p^*}.

We take $A := 2T^{-1} - (I_p - \Sigma_{xz}\Sigma_{zx})^{-1}$, $B := (I_p - \Sigma_{xz}\Sigma_{zx})^{-1} \Sigma_{xz}$, $D := I_{p^*} - \Sigma_{zx} (I_p - \Sigma_{xz}\Sigma_{zx})^{-1} \Sigma_{xz}$. The block matrix is then

$$M := \begin{vmatrix} A & B \\ B^\top & D \end{vmatrix}$$

Assuming D is invertible, we can use the Schur complement

$$\det(M) = \det(D) \det\left(A - BD^{-1}B^\top\right)$$

Then

$$\det(A - BD^{-1}B^\top) = \det(2T^{-1} - (I_p - \Sigma_{xz}\Sigma_{zx})^{-1} - BD^{-1}B^\top).$$

which can easily be optimized with gradient descent with respect to $T = c_{p^*} I_p$.

Appendix E. bd-CME test derivation

We observe $\{(x_i, y_i, z_i)\}_{i=1}^n \sim p$, some probability density on the joint space $\mathcal{X} \times \mathcal{Y} \times \mathcal{Z}$. Define

$$p(y|do(x)) \propto \tilde{p}(y|do(x)) = \int p(y|x, z)p(z)dz.$$

Note that $p(y|do(x)) \neq p(y|x)$ in general since X and Z need not be independent. We would like to test

$$H_0 : \ p(y|do(x)) \text{ does not depend on } x$$

versus general alternative. Let k, l, m be kernels on $\mathcal{X}, \mathcal{Y}, \mathcal{Z}$, respectively. We first fit the conditional mean embedding

$$\mu_{p(\cdot|x,z)} \quad = \quad \int l(\cdot, y) p(y|x, z) dy,$$

by learning a regression function $k(\cdot, x) \otimes m(\cdot, z) \mapsto l(\cdot, y)$. This is given by $\hat{\mu}_{p(\cdot|x,z)} = \sum_{i=1}^{n} a(x, z)_i l(\cdot, y_i)$, where

$$a(x, z) = \left(K_{\mathbf{xx}} \circ M_{\mathbf{zz}} + \epsilon_n I\right)^{-1} \left(K_{\mathbf{x}x} \circ M_{\mathbf{z}z}\right),$$

$[K_{\mathbf{xx}}]_{ij} = k(x_i, x_j)$ is the $n \times n$ Gram matrix and $K_{\mathbf{x}x} \in \mathbb{R}^n$, with $[K_{\mathbf{x}x}]_i = k(x_i, x)$, and similarly for kernel m. Now, to obtain an estimate of

$$\mu_{\tilde{p}(\cdot|do(x))} \quad = \quad \int l(\cdot, y) \int p(y|x, z) p(z) dz dy,$$

we simply average over the empirical $\hat{p}(z)$ to obtain:

$$\hat{\mu}_{\tilde{p}(\cdot|do(x))} = w(x)^\top L_{\mathbf{y}\cdot} = \sum_{i=1}^{n} w(x)_i l(\cdot, y_i) = \frac{1}{n} \sum_{i=1}^{n} \sum_{j=1}^{n} a(x, z_j)_i l(\cdot, y_i),$$

i.e.

$$w(x) = \frac{1}{n} \left(K_{\mathbf{xx}} \circ M_{\mathbf{zz}} + \epsilon_n I\right)^{-1} \left(K_{\mathbf{x}x} \circ M_{\mathbf{zz}} \mathbf{1}\right).$$

Denote

$$W_{\mathbf{x}} = [w(x_1) \cdots w(x_n)]^\top = \frac{1}{n} \left(K_{\mathbf{xx}} \circ \mathbf{1}\mathbf{1}^\top M_{\mathbf{zz}}\right) \left(K_{\mathbf{xx}} \circ M_{\mathbf{zz}} + \epsilon_n I\right)^{-1}.$$

We also denote $\bar{w} = \frac{1}{n} \sum_{i=1}^{n} w(x_i)$. Then $\hat{\mu}_{\tilde{p}} = \bar{w}^\top L_{\mathbf{y}\cdot}$ estimates the embedding of $\tilde{p}(y) = \int p(y|x, z) p(x) p(z) dx dz$. Now, under the null, $\tilde{p} = \tilde{p}(\cdot|do(x)), \forall x$. Thus, we define the statistic as the sum of the RKHS distances of the corresponding embeddings

$$\begin{aligned}
S &= \sum_{i=1}^{n} \left\| \hat{\mu}_{\tilde{p}(\cdot|do(x)=x_i)} - \hat{\mu}_{\tilde{p}} \right\|_{\mathcal{H}_l}^2 \\
&= \sum_{i=1}^{n} \left\| \left(w(x_i) - \bar{w}\right)^\top L_{\mathbf{y}\cdot} \right\|_{\mathcal{H}_l}^2 \\
&= \sum_{i=1}^{n} \mathrm{Tr}\left[\left(w(x_i) - \bar{w}\right)^\top L_{\mathbf{yy}} \left(w(x_i) - \bar{w}\right) \right] \\
&= \mathrm{Tr}\left[L_{\mathbf{yy}} \sum_{i=1}^{n} \left(w(x_i) - \bar{w}\right) \left(w(x_i) - \bar{w}\right)^\top \right] \\
&= \mathrm{Tr}\left[L_{\mathbf{yy}} W_{\mathbf{x}}^\top H W_{\mathbf{x}} \right].
\end{aligned}$$

The test statistic S uses a mean embedding that bears many similarities to Singh et al. (2020). It should be noted that Singh et al. (2020) considers the linear kernel applied to Y as they want to calculate the ATE directly. In this context, we instead consider an RBF kernel on Y instead, which instead takes the outcome to the RKHS.

Appendix F. Simulation algorithms

F.1 Binary Treatment

Fix $\beta_{XY} > 0$, and $\tau^2 > 0$. H_0 case:

$$\begin{aligned} Z_i &\sim \mathcal{N}(0,1), \\ X_i|Z_i &\sim \text{Bernoulli}\left(\frac{1}{1+e^{-Z_i}}\right), \\ Y_i|Z_i &\sim \mathcal{N}\left(\beta_{XY} Z_i, \tau^2\right). \end{aligned}$$

H_1 case:

$$\begin{aligned} Z_i &\sim \mathcal{N}(0,1), \\ X_i|Z_i &\sim \text{Bernoulli}\left(\frac{1}{1+e^{-Z_i}}\right), \\ Y_i|Z_i &\sim \mathcal{N}\left(\beta_{XY}\left(2X_i - 1\right)|Z_i|, \tau^2\right). \end{aligned}$$

Note that in the alternative case, X_i directly modulates the sign of the mean of Y_i. However, $2X_i - 1$ will be strongly positively correlated with the sign of Z_i implying that there is only a slight change in the dependence structure of (X_i, Y_i, Z_i). In addition, it is clear that all the marginals are the same.

F.2 Continuous treatment

We simulate data from the do-null and the alternative using rejection sampling (Evans and Didelez, 2021), described in Algorithm 4.

Algorithm 4: Generating continuous data for H_0 and H_1

Input: Number of samples n, dependencies β_{XY}, β_{XZ}, β_{YZ}, variance parameters $\theta, \phi \in \mathbb{R}^+$, dimensions d_x, d_y, d_z

Initialize data container $\mathcal{D} = \{\}$

Set $\boldsymbol{\beta}_{XZ} = [\underbrace{\beta_{XZ}}_{1:3}, \underbrace{0}_{4:d_Z}]$

while # *of samples* $< n$ **do**

- Set $p_X = \mathcal{N}(\mathbf{0}, \theta \cdot \phi \cdot \mathbf{I}_{d_x})$ Sample $\{x_i\}_{i=1}^N \sim p_X$
- Sample $\{y_i, z_i\}_{i=1}^N \sim \mathcal{N}(\mathbf{0}, \Sigma_{d_y+d_z})$
- Transform $\{y_i'\}_{i=1}^N = \text{CDF}_{\mathcal{N}(0,1)}(\{y_i\}_{i=1}^N)$
- Define $p_{Y|X} = \mathcal{N}(X\beta_Y, 1)$
- Set $\{y_i\}_{i=1}^N = \text{ICDF}_{p_{Y|X}}(\{y_i'\}_{i=1}^N)$
- Set $\mu_{X|Z} = Z \cdot \boldsymbol{\beta}_{XZ}$
- Define $p_{X|Z} = \mathcal{N}(\mu_{X|Z}, \phi)$
- Calculate $\omega_i = \frac{p_{X|Z}(x_i)}{p_X(x_i)}$
- Run rejection sampling using ω_i and obtain $\mathcal{D}' = \{x_i, y_i, z_i\}_{i=1}^{N'} \sim p^*$
- Append data $\mathcal{D} = \mathcal{D} \cup \mathcal{D}'$

end

Return: $\mathcal{D}$

F.2.1 Parameter explanation

There are several parameters used in the data generation algorithm primarily used to control for the difficulty of the problem and the ground truth hypothesis.

1. β_{XY}: Controls the dependency between X and Y. A $\beta_{XY} > 0$ implies H_1 ground truth and $\beta_{XY} = 0$ implies H_0 ground truth
2. β_{XZ}: Controls the dependency between X and Z. A high β_{XZ} implies a stronger dependency on Z for X, implying a harder problem.
3. β_{YZ}: Controls the dependency between Y and Z. Fixed at a high value to ensure Y is being confounded by Z.
4. θ, ϕ: Controls the variance of p_X and $p_{X|Z}$. $\theta > \phi$ ensures a higher *Effective Sample Size* for true weights.
5. d_x, d_y, d_z: Dimensionality of X, Y, Z. Higher dimensions imply a harder problem.
6. $\Sigma_{d_y+d_z}$: The covariance matrix that links Z and Y. It should be noted that this covariance matrix is a function of X.

F.3 Mixed treatment

Algorithm 5: Generating mixed data for H_0 and H_1

Input: Number of samples n, dependencies β_{XY}, β_{XZ}, β_{YZ}, variance parameters $\theta, \phi \in \mathbb{R}^+$, dimensions d_x, d_y, d_z

Initialize data container $\mathcal{D} = \{\}$

Set $\boldsymbol{\beta}_{XZ} = [\underbrace{\beta_{XZ}}_{1:3}, \underbrace{0}_{4:d_Z}]$

while # *of samples* $< n$ **do**

- Set $p_X = \mathcal{N}(\mathbf{0}, \theta \cdot \phi \cdot \mathbf{I}_{d_x})$
- Set $p_X^{\text{bin}} = \text{Bin}(p = 0.5)$
- Sample $\{x_i^{\text{cont}}\}_{i=1}^N \sim p_X$
- Sample $\{x_i^{\text{bin}}\}_{i=1}^N \sim \text{Bin}(p = 0.5)$
- Concatenate $X = X_{\text{cont}} \cup X_{\text{bin}}$
- Sample $\{y_i, z_i\}_{i=1}^N \sim \mathcal{N}(\mathbf{0}, \Sigma_{d_y+d_z})$
- Transform $\{y_i'\}_{i=1}^N = \text{CDF}_{\mathcal{N}(0,1)}(\{y_i\}_{i=1}^N)$
- Define $p_{Y|X} = \mathcal{N}(X\beta_Y, 1)$
- Set $\{y_i\}_{i=1}^N = \text{ICDF}_{p_{Y|X}}(\{y_i'\}_{i=1}^N)$
- Set $\mu_{X|Z} = Z \cdot \boldsymbol{\beta}_{XZ}$
- Set $\nu_{X|Z} = \frac{1}{1+e^{-\mu_{X|Z}}}$
- Define $p_{X|Z} = \mathcal{N}(\mu_{X|Z}, \phi)$
- Define $p_{X|Z}^{\text{bin}} = \text{Bin}(p = \nu_{X|Z})$
- Calculate $\omega_i = \frac{p_{X|Z}(x_i^{\text{cont}})}{p_X(x_i^{\text{cont}})} \cdot \frac{p_{X|Z}^{\text{bin}}(x_i^{\text{bin}})}{p_X^{\text{bin}}(x_i^{\text{bin}})}$
- Run rejection sampling using ω_i and obtain $\mathcal{D}' = \{x_i, y_i, z_i\}_{i=1}^{N'} \sim p^*$
- Append data $\mathcal{D} = \mathcal{D} \cup \mathcal{D}'$

end

Return: $\mathcal{D}$

Appendix G. Parameters for data generation

We provide parameters used in the data generation procedure for each type of treatment. Exact details can be found in the code base.

Binary treatment

1. Dependency β_{XY}: $[0.0, 0.02, 0.04, 0.06, 0.08, 0.1]$
2. Variance τ: 1.0

Continuous treatment

1. β_{XY}: $[0.0, 0.001, 0.002, 0.003, 0.004, 0.005, 0.008, 0.012, 0.016, 0.02]$
2. β_{XZ}: $d_Z = 1 : 0.75, d_Z = 3, 15, 50 : 0.25$
3. β_{YZ}: $[0.5, 0.0]$
4. θ, ϕ: $d_Z = 1 : (2, 2)$, $d_Z = 3 : (4, 2)$, $d_Z = 15 : (8, 2)$, $d_Z = 50 : (16, 2)$
5. $\Sigma_{d_y+d_z}$: See code base for details

Mixed treatment

1. β_{XY}: $[0.0, 0.002, 0.004, 0.006, 0.008, 0.01, 0.015, 0.02, 0.025, 0.03, 0.04, 0.05, 0.1]$
2. β_{XZ}: 0.05
3. β_{YZ}: $[0.5, 0.0]$
4. θ, ϕ: $d_Z = 2 : (2, 2)$, $d_Z = 15 : (16, 2)$, $d_Z = 50 : (16, 2)$
5. $\Sigma_{d_y+d_z}$: See code base for details

$X \not\perp Y$ data

1. β_{XY}: $[0.0]$
2. β_{XZ}: 1.0
3. β_{YZ}: $[0.5, 0.0]$
4. θ, ϕ: $d_Z = 1 : (0.1, 1.5)$
5. $\Sigma_{d_y+d_z}$: See code base for details

$X \not\perp Y \mid Z$ data

1. β_{XY}: $[0.0]$
2. β_{XZ}: 0.0.
3. β_{YZ}: $[-0.5, 4.0]$
4. θ, ϕ: $d_Z = 1 : (1.0, 2.0)$
5. $\Sigma_{d_y+d_z}$: See code base for details

References

Alexandre Belloni, Victor Chernozhukov, and Christian Hansen. Inference on Treatment Effects after Selection among High-Dimensional Controls. *The Review of Economic Studies*, 81(2):608–650, 04 2014. ISSN 0034-6527. doi: 10.1093/restud/rdt044. URL https://doi.org/10.1093/restud/rdt044.

Z. I. Botev, J. F. Grotowski, and D. P. Kroese. Kernel density estimation via diffusion. *The Annals of Statistics*, 38(5):2916 – 2957, 2010. doi: 10.1214/10-AOS799. URL https://doi.org/10.1214/10-AOS799.

Daniel C. Castro, Ian Walker, and Ben Glocker. Causality matters in medical imaging. *Nature Communications*, 11(1), Jul 2020. ISSN 2041-1723. doi: 10.1038/s41467-020-17478-w. URL http://dx.doi.org/10.1038/s41467-020-17478-w.

Victor Chernozhukov, Denis Chetverikov, Mert Demirer, Esther Duflo, Christian Hansen, Whitney Newey, and James Robins. Double/debiased machine learning for treatment and structural parameters. *The Econometrics Journal*, 21(1):C1–C68, 01 2018. ISSN 1368-4221. doi: 10.1111/ectj.12097. URL https://doi.org/10.1111/ectj.12097.

Rhian M Daniel, SN Cousens, BL De Stavola, Michael G Kenward, and JAC Sterne. Methods for dealing with time-dependent confounding. *Statistics in medicine*, 32(9):1584–1618, 2013.

Laurent Charlin Dawen Liang and David M. Blei. Causal inference for recommendation. In *RecSys '20: Fourteenth ACM Conference on Recommender Systems*, New York, NY, USA, 2020. Association for Computing Machinery. ISBN 9781450375832.

Gary Doran, Krikamol Muandet, Kun Zhang, and Bernhard Schölkopf. A permutation-based kernel conditional independence test. In *UAI*, pages 132–141. Citeseer, 2014.

Tim Ensor, Stephanie L. Cooper, Lisa Davidson, Ann E. Fitzmaurice, and Wendy Jane Graham. The impact of economic recession on maternal and infant mortality: lessons from history. *BMC Public Health*, 10:727 – 727, 2010.

Robin J Evans and Vanessa Didelez. Parameterizing and simulating from causal models, 2021. arXiv preprint arXiv:2109.03694.

Jake Fawkes, Robert Hu, Robin J. Evans, and Dino Sejdinovic. Doubly robust kernel statistics for testing distributional treatment effects even under one sided overlap, 2022. URL https://arxiv.org/abs/2212.04922.

Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative adversarial nets. In Z. Ghahramani, M. Welling, C. Cortes, N. Lawrence, and K. Q. Weinberger, editors, *Advances in Neural Information Processing Systems*, volume 27. Curran Associates, Inc., 2014. URL https://proceedings.neurips.cc/paper/2014/file/5ca3e9b122f61f8f06494c97b1afccf3-Paper.pdf.

Arthur Gretton, Olivier Bousquet, Alex Smola, and Bernhard Schölkopf. Measuring statistical dependence with Hilbert-Schmidt norms. In *Proceedings of the 16th International Conference on Algorithmic Learning Theory*, ALT'05, page 63–77, Berlin, Heidelberg, 2005. Springer-Verlag. ISBN 354029242X. doi: 10.1007/11564089_7. URL https://doi.org/10.1007/11564089_7.

Michael U. Gutmann and Aapo Hyvärinen. Noise-contrastive estimation of unnormalized statistical models, with applications to natural image statistics. *Journal of Machine Learning Research*, 13(11):307–361, 2012. URL http://jmlr.org/papers/v13/gutmann12a.html.

Vasant Honavar and Sanghack Lee. A kernel conditional independence test for relational data. *Conference on Uncertainty in Artificial Intelligence*, 01 2017.

Alexander P Keil, Jessie P Buckley, Katie M O'Brien, Kelly K Ferguson, Shanshan Zhao, and Alexandra J White. A quantile-based g-computation approach to addressing the effects of exposure mixtures. *Environmental health perspectives*, 128(4):047004, 2020.

Scott Kostyshak. Non-parametric testing of U-shaped relationships. *Econometrics: Econometric & Statistical Methods - General eJournal*, 2017.

Christos Louizos, Uri Shalit, Joris M Mooij, David Sontag, Richard Zemel, and Max Welling. Causal effect inference with deep latent-variable models. In I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, editors, *Advances in Neural Information Processing Systems*, volume 30. Curran Associates, Inc., 2017. URL https://proceedings.neurips.cc/paper/2017/file/94b5bde6de888ddf9cde6748ad2523d1-Paper.pdf.

J. B. MacQueen. Some methods for classification and analysis of multivariate observations. In L. M. Le Cam and J. Neyman, editors, *Proc. of the fifth Berkeley Symposium on Mathematical Statistics and Probability*, volume 1, pages 281–297. University of California Press, 1967.

S McGrath, J G Young, and MA Hernán. Revisiting the g-null paradox. *Epidemiology*, 2021.

Lars Mescheder, Andreas Geiger, and Sebastian Nowozin. Which training methods for gans do actually converge? In *International conference on machine learning*, pages 3481–3490. PMLR, 2018.

Taku Moriyama and Masashi Kuwano. Causal inference for contemporaneous effects and its application to tourism product sales data. *Journal of Marketing Analytics*, 08 2021. doi: 10.1057/s41270-021-00130-x.

Krikamol Muandet, Kenji Fukumizu, Bharath Sriperumbudur, and Bernhard Schölkopf. Kernel mean embedding of distributions: A review and beyond. *Foundations and Trends® in Machine Learning*, 10(1-2):1–141, 2017. ISSN 1935-8245. doi: 10.1561/2200000060. URL http://dx.doi.org/10.1561/2200000060.

Krikamol Muandet, Motonobu Kanagawa, Sorawit Saengkyongam, and Sanparith Marukatat. Counterfactual mean embeddings. *J. Mach. Learn. Res.*, 22(1), jan 2021. ISSN 1532-4435.

Judea Pearl. *Causality*. Cambridge University Press, 2 edition, 2009. doi: 10.1017/CBO9780511803161.

Danilo Rezende and Shakir Mohamed. Variational inference with normalizing flows. In Francis Bach and David Blei, editors, *Proceedings of the 32nd International Conference on Machine Learning*, volume 37 of *Proceedings of Machine Learning Research*, pages 1530–1538, Lille, France, 07–09 Jul 2015. PMLR. URL http://proceedings.mlr.press/v37/rezende15.html.

Benjamin Rhodes, Kai Xu, and Michael U. Gutmann. Telescoping density-ratio estimation. In H. Larochelle, M. Ranzato, R. Hadsell, M. F. Balcan, and H. Lin, editors, *Advances in Neural Information Processing Systems*, volume 33, pages 4905–4916. Curran Associates, Inc., 2020. URL https://proceedings.neurips.cc/paper/2020/file/33d3b157ddc0896addfb22fa2a519097-Paper.pdf.

Thomas S Richardson, Robin J Evans, James M Robins, and Ilya Shpitser. Nested Markov properties for acyclic directed mixed graphs. arXiv preprint arXiv:1701.06686, 2017.

David Rindt, Dino Sejdinovic, and David Steinsaltz. A kernel- and optimal transport- based test of independence between covariates and right-censored lifetimes. *The International Journal of Biostatistics*, page 20200022, 2020. doi: doi:10.1515/ijb-2020-0022. URL https://doi.org/10.1515/ijb-2020-0022.

David Rindt, Dino Sejdinovic, and David Steinsaltz. Consistency of permutation tests of independence using distance covariance, hsic and dhsic. *Stat*, 10(1):e364, 2021. doi: https://doi.org/10.1002/sta4.364. URL https://onlinelibrary.wiley.com/doi/abs/10.1002/sta4.364. e364 sta4.364.

James M. Robins. A new approach to causal inference in mortality studies with a sustained exposure period—application to control of the healthy worker survivor effect. *Mathematical Modelling*, 7(9-12):1393–1512, 1986.

James M Robins and Larry A Wasserman. Estimation of effects of sequential treatments by reparameterizing directed acyclic graphs. In *Proceedings of the 13th Conference on Uncertainty in Artificial Intelligence*, 1997.

Paul R Rosenbaum. Conditional permutation tests and the propensity score in observational studies. *Journal of the American Statistical Association*, 79(387):565–574, 1984.

Paul R. Rosenbaum and Donald B. Rubin. The central role of the propensity score in observational studies for causal effects. *Biometrika*, 70(1):41–55, 1983. ISSN 00063444. URL http://www.jstor.org/stable/2335942.

Kenneth J. Rothman and Sander Greenland. Causation and causal inference in epidemiology. *American Journal of Public Health*, 95(S1):S144–S150, 2005. doi: 10.2105/AJPH.2004.059204. URL https://doi.org/10.2105/AJPH.2004.059204. PMID: 16030331.

Peter J. Rousseeuw. Silhouettes: A graphical aid to the interpretation and validation of cluster analysis. *Journal of Computational and Applied Mathematics*, 20:53–65, 1987. ISSN 0377-0427. doi: https://doi.org/10.1016/0377-0427(87)90125-7. URL https://www.sciencedirect.com/science/article/pii/0377042787901257.

Bernhard Scholköpf and Alexander J. Smola. *Learning with Kernels: Support Vector Machines, Regularization, Optimization, and Beyond*. MIT Press, Cambridge, MA, USA, 2001. ISBN 0262194759.

Rahul Singh, Liyuan Xu, and Arthur Gretton. Kernel methods for causal functions: Dose response curves and heterogeneous treatment effects, 2020. URL https://arxiv.org/abs/2010.04855.

Le Song, Alex Smola, Arthur Gretton, Karsten M. Borgwardt, and Justin Bedo. Supervised feature selection via dependence estimation. In *Proceedings of the 24th International Conference on Machine Learning*, ICML '07, page 823–830, New York, NY, USA, 2007. Association for Computing Machinery. ISBN 9781595937933. doi: 10.1145/1273496.1273600. URL https://doi.org/10.1145/1273496.1273600.

Bharath K. Sriperumbudur, Kenji Fukumizu, and Gert R.G. Lanckriet. Universality, characteristic kernels and RKHS embedding of measures. *Journal of Machine Learning Research*, 12(70):2389–2410, 2011. URL http://jmlr.org/papers/v12/sriperumbudur11a.html.

Eric V. Strobl, Kun Zhang, and Shyam Visweswaran. Approximate kernel-based conditional independence tests for fast non-parametric causal discovery. *Journal of Causal Inference*, 7(1):20180017, 2019. doi: doi:10.1515/jci-2018-0017. URL https://doi.org/10.1515/jci-2018-0017.

Daiki Watanabe, Tsukasa Yoshida, Yuya Watanabe, Yosuke Yamada, and Misaka Kimura. A U-shaped relationship between the prevalence of frailty and body mass index in community-dwelling japanese older adults: The kyoto–kameoka study. *Journal of Clinical Medicine*, 9, 2020.

Makoto Yamada, Taiji Suzuki, Takafumi Kanamori, Hirotaka Hachiya, and Masashi Sugiyama. Relative density-ratio estimation for robust distribution comparison. In J. Shawe-Taylor, R. Zemel, P. Bartlett, F. Pereira, and K. Q. Weinberger, editors, *Advances in Neural Information Processing Systems*, volume 24. Curran Associates, Inc., 2011. URL https://proceedings.neurips.cc/paper/2011/file/d1f255a373a3cef72e03aa9d980c7eca-Paper.pdf.

Kun Zhang, Jonas Peters, Dominik Janzing, and Bernhard 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, Arlington, Virginia, USA, 2011. AUAI Press. ISBN 9780974903972.

Qinyi Zhang, Sarah Filippi, Arthur Gretton, and Dino Sejdinovic. Large-scale kernel methods for independence testing. *Statistics and Computing*, 28(1):113–130, Jan 2017. ISSN 1573-1375. doi: 10.1007/s11222-016-9721-7. URL http://dx.doi.org/10.1007/s11222-016-9721-7.

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	
Publication Status	☐Published ☐ Accepted for Publication ☒Submitted for Publication ☐Unpublished and unsubmitted work written in a manuscript style
Publication Details	Hu, R., Sejdinovic D. and Evans R.J. (2021). A Kernel Test for Causal Association via Noise Contrastive Backdoor Adjustment. arXiv:2111.13226

Student Confirmation

Student Name:	Robert Hu		
Contribution to the Paper	I conceived implementation, experiments and did theory for the paper. During the meetings my supervisors helped out with helpful suggestions on methodology and feedback on writing. They also contributed a lot to the writing of the draft, checking the proofs, and proof-reading.		
Signature		Date	29/8-2022

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: Dino Sejdinovic		
Supervisor comments I am in agreement with the description of the contributions.		
Signature 	Date	09/09/2022

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

4

Large Scale Tensor Regression using Kernels and Variational Inference

Machine Learning
https://doi.org/10.1007/s10994-021-06067-7

Large scale tensor regression using kernels and variational inference

Robert Hu[1] · **Geoff K. Nicholls**[1] · **Dino Sejdinovic**[1]

Received: 9 April 2020 / Revised: 23 June 2021 / Accepted: 6 September 2021
© The Author(s) 2021

Abstract
We outline an inherent flaw of tensor factorization models when latent factors are expressed as a function of side information and propose a novel method to mitigate this. We coin our methodology *kernel fried tensor* (KFT) and present it as a large-scale prediction and forecasting tool for high dimensional data. Our results show superior performance against *LightGBM* and *Field aware factorization machines* (FFM), two algorithms with proven track records, widely used in large-scale prediction. We also develop a variational inference framework for KFT which enables associating the predictions and forecasts with calibrated uncertainty estimates on several datasets.

Keywords Large scale prediction · Tensor · RKHS · Kernel methods · Variational inference · Bayesian · Uncertainty quantification

1 Introduction and related work

In recent times, industrial prediction problems (Caro and Gallien 2010; Seeger et al. 2016) are not only large scale but also high dimensional (Zhai et al. 2014). Problems of this nature are ubiquitous and the most common setting is, but not limited to, the recommendation systems (Bobadilla et al. 2013). Here we are tasked with predicting user preference (i.e. ratings, buying propensity, etc.) for a product (movies, clothing, etc). More often than not, the number of users and products far exceeds the practicality of data matrix formalism, which characterizes the perils of high dimensionality in modern prediction problems. The choice of models is often limited to boosting models such as *LightGBM* (Ke et al. 2017),

Editor: Shai Shalev-Shwartz.

✉ Robert Hu
robert.hu@stats.ox.ac.uk

Geoff K. Nicholls
nicholls@stats.ox.ac.uk

Dino Sejdinovic
dino.sejdinovic@stats.ox.ac.uk

[1] Department of Statistics, University of Oxford, Oxford, UK

Published online: 09 November 2021

factorization machines (Juan et al. 2016; Rendle 2010) and matrix (Bobadilla et al. 2013) and tensor factorization models (Kolda and Bader 2009; Oseledets 2011). In particular, matrix/tensor factorization models are often used due to their memory-efficient representation of high dimensional data (Kuang et al. 2014). An additional benefit of factorization machines and matrix/tensor factorization models is their relative ease of extension to a Bayesian formulation, making uncertainty quantification straightforward.

A recurring problem in recommendation systems is the *cold start* problem (Bobadilla et al. 2012), where new users yield inaccurate predictions due to an absence of historical purchasing behavior. A common technique to overcome the cold start problem is to incorporate *side information* (Agarwal and Chen 2009; Xu et al. 2015; Zhang et al. 2014), which means adding descriptive covariates about each individual user (or product) to the model. While side information is not immediately applicable to factorization machines, there is extensive literature (Kim et al. 2016; Liu et al. 2019; Narita et al. 2012) on utilizing side information with matrix/tensor factorization models. There are limited choices of models when simultaneously considering scalability, uncertainty quantification, the cold start problem, and, ideally, interpretability (Rudin 2018). We contextualize our contribution by reviewing related works.

Tensor factorization is a generalization of matrix factorization to n-dimensions, where any technique that applies to tensors is directly applicable to matrices but not vice versa. We focus on tensor factorization as it offers the most flexibility and generality. In the frequentist setting, Canonical Polyadic (CP) and Tucker decomposition (Kolda and Bader 2009) are the most common factorization methods for tensors while Tensor-Train (TT) (Oseledets 2011) and Tensor-Ring (TR) (Zhao et al. 2016) decomposition are newer additions focusing on scalability. While Tucker decomposition has admirable analytical properties, the $\mathcal{O}(r^d)$ memory storage of the core tensor is infeasible in any large-scale application. CP decomposition is superseded by TT-decomposition, which in turn is extended by TR-decomposition. Due to certain pathologies (Batselier 2018) exhibited by TR, only TT remains plausible for a large-scale model. The existing methods *LightGBM*(Ke et al. 2017) and *Field-Aware Factorization Machines*(FFM) (Juan et al. 2016) are considered the gold standard[1,2] of large scale prediction, where an overall objective of this paper is to challenge this duopoly. To enhance the performance of tensor models, Du et al. (2016), Wu et al. (2019) and Zhang et al. (2019) applied neural methods for matrix and tensor factorization, where (Du et al. 2016; Zhang et al. 2019) is considered the state-of-the-art in performance. In the Bayesian domain, only FFM carries over with (Saha et al. 2017), which introduces a variational coordinate ascent method for factorization machines. There is rich literature in Bayesian matrix/tensor factorization ranging from Monte Carlo methods in the pioneering works of Salakhutdinov and Mnih (2007) to variational matrix factorization (Gönen et al. 2013; Kim and Choi 2014) and tensor factorization in Hawkins and Zhang (2018). Kim and Choi (2014) is of particular interest as they present a large-scale variational model incorporating side information. While coordinate ascent approaches proposed in Hawkins and Zhang (2018), Kim and Choi (2014) and Saha et al. (2017) are useful, we believe that the cyclical update scheme can be constraining for large scale scenarios that rely on parallelism. Further, the updating rules are hard to maintain with changes in model architecture. To mitigate maintenance of complicated gradient updates, we consider

[1] https://www.kaggle.com/c/avazu-ctr-prediction/discussion/12608.
[2] https://bit.ly/3bGGzpf.

Table 1 Where our contribution places

	Interpretable	Scalable	Tensor	Bayesian	Side information	Performant
CF-NADE (Du et al. 2016)		✓				✓
FFM (Juan et al. 2016)		✓		✓		✓
LightGBM (Ke et al. 2017)		✓			✓	✓
Tensor Factorization with Auxiliary side information (Narita et al. 2012)			✓		✓	
NTF (Wu et al. 2019)			✓			✓
KPMF (Pal and Jenamani 2018)					✓	✓
VBMF (Kim and Choi 2014)	✓	✓		✓	✓	
Kernel fried tensor	✓	✓	✓	✓	✓	✓

automatic differentiation (Paszke et al. 2017b) instead. In the context of maintainable variational inference, it would then suffice to find an analytical expression of the Evidence Lower BOund (ELBO) that is general for a family of models (Hoffman et al. 2013).

Side information is applied in two ways for matrix/tensor factorization models: implicitly through regularization schemes in He et al. (2017), Narita et al. (2012) and Pal and Jenamani (2018) Zhao et al. (2013) and directly as covariates in Agarwal and Chen (2009), Kim et al. (2016), Kim and Choi (2014) and Zhang et al. (2014). In terms of interpretability, the latter is more desirable as predictions are now an explicit expression of covariates, allowing for direct attribution analysis. A concerning observation of the results in Agarwal and Chen (2009), Kim et al. (2016) and Kim and Choi (2014) suggests that using side information in a covariate format barely improves performance and even worsens it in Agarwal and Chen (2009) using the "features only" model.

In this paper we develop a novel all-purpose large-scale prediction model that strives for a new level of versatility existing models lack. Our contribution *Kernel Fried Tensor*(KFT) aims to bridge the gap in the literature and answer the following questions:

1. Is there an interpretable tensor model that avoids constraints and complex global dependencies arising from the addition of side information but still makes full use of side information?
2. Can we formulate and characterize this model class in both primal and dual (Reproducing Kernel Hilbert Space) space?
3. Do models in this class compare favourably, for large scale prediction, to state-of-the-art models such as *LightGBM* (Ke et al. 2017), FFM (Juan et al. 2016) and existing factorization models?
4. Can we work with these new models in a scalable Bayesian context with calibrated uncertainty estimates?
5. What are the potential pitfalls of using these models? When are they appropriate to use?

Following our introduction, we were tasked with the multifaceted problem of developing a tensor model that scales, is interpretable, is Bayesian, handles side information, and provides on-par performance with the existing gold-standard.

The rest of the paper is organized as follows: Sect. 2 illustrates an important limitation of side information applied as covariates to tensors, Sect. 3 introduces and characterises KFT in both the frequentist and Bayesian setting, Sect. 4 are experiments and Sect. 5 provides an ablation study related to Question 5 (Table 1).

2 Background and problem

Tensor models are used in large-scale prediction tasks ranging from bioinformatics to industrial prediction; we work in the latter setting. While existing tensor models are versatile and scalable, they have a flaw: When we build a model of the latent factors in tensor factorization as a function of covariates (Agarwal and Chen 2009; Kim et al. 2016; Kim and Choi 2014; Zhang et al. 2014), the model may be restricted by global parameter couplings that are generated. These couplings lead to a reduction of tractability at scale. We provide a new family of tensor models which admit side information without model restriction or loss of tractability.

Tensor Train decomposition Before we explain how side information restricts tensor model expressiveness, we set out the background. Consider the task of reconstructing the tensor $\mathbf{Y} \in \mathbb{R}^{n_1 \times n_2 \cdots \times n_P}$. Many existing decomposition techniques (Kolda and Bader 2009) treat this problem. We focus on the Tensor Train (TT) decomposition (Oseledets 2011) as this generalises more readily than existing alternatives.

The n-mode (matrix) product of a tensor $\mathbf{X} \in \mathbb{R}^{I_1 \times I_2 \times \cdots \times I_N}$ with a matrix $\mathbf{U} \in \mathbb{R}^{J \times I_n}$ is $\mathbf{X} \times_n \mathbf{U}$. This product is of size $I_1 \times \ldots I_{n-1} \times J \times I_{n+1} \times \ldots \times I_N$. Elementwise, the product is

$$\left(\mathbf{X} \times_n \mathbf{U}\right)_{i_1 \ldots i_{n-1}\, j\, i_{n+1} \,\ldots\, i_N} = \sum_{i_n=1}^{I_n} x_{i_1 i_2 \ldots i_N}\, u_{j i_n}.$$

We will consider the following notion of a mode product between tensor $\mathbf{X} \in \mathbb{R}^{I_1 \times \cdots I_{N-1} \times I_N}$, applying the N-th mode product $\times_N$ with a tensor $\mathbf{U} \in \mathbb{R}^{I_N \times K_1 \times K_2}$ gives $\mathbf{X} \times_N \mathbf{U} \in \mathbb{R}^{I_1 \times \cdots I_{N-1} \times K_1 \times K_2}$.

In TT, $\mathbf{Y}$ is decomposed into P latent tensors $\mathbf{V}_p \in \mathbb{R}^{R_p \times n_p \times R_{p-1}}, p = 1, \ldots P$, with $\mathbf{V}_1 \in \mathbb{R}^{R_1 \times n_1 \times 1}$ and $\mathbf{V}_P \in \mathbb{R}^{1 \times n_P \times R_{P-1}}$. Here R_p is the latent dimensionality for each factor in the decomposition, with $R_1 = R_P = 1$. Let $\times_{-1} \mathbf{V}_p$ be the operation of applying the mode product to the last dimension of $\mathbf{V}_p$. We seek $\mathbf{V}_1 \ldots \mathbf{V}_P$ so that

$$\begin{aligned} \mathbf{Y} &\approx \prod_{p=1}^{P} \times_{-1} \mathbf{V}_p \\ y_{i_1 \ldots i_P} &\approx \sum_{r_0 \ldots r_P} \prod_{p=1}^{P} v_{r_p i_p r_{p-1}}. \end{aligned} \tag{1}$$

Suppose that, associated with dimension p, we have c_p-dimensional side information denoted $\mathbf{D}_p \in \mathbb{R}^{n_p \times c_p}$. For example, if p is the dimension representing $n_p = 10{,}000$ different books, then the columns of $\mathbf{D}_p \in \mathbb{R}^{10{,}000 \times c_p}$ might contain the author of the book, page count etc. Similar to Kim et al. (2016) and Kim and Choi (2014), side information is built into the second dimension of the latent tensor $\mathbf{V}_p \in \mathbb{R}^{R_p \times c_p \times R_{p-1}}$ using the mode product $\mathbf{V}_p \times_2 \mathbf{D}_p$. It should be noted that the middle dimension of $\mathbf{V}_p$ changed from n_p to c_p to accommodate the dimensionality of the side information. For TT decomposition our approximation becomes

$$\mathbf{Y} \approx \prod_{p=1}^{P} \times_{-1} (\mathbf{V}_p \times_2 \mathbf{D}_p)$$
$$y_{i_1 \dots i_P} \approx \sum_{\substack{r_0 \dots r_P \\ i'_1 \dots i'_P}} \prod_{p=1}^{P} v_{r_p i'_p r_{p-1}} d_{i_p, i'_p} \tag{2}$$

The above example illustrates the *primal* setting where side information is applied directly. Similarly to Kim et al. (2016), we also consider kernelized side information in the reproducing kernel hilbert space (RKHS) which we will refer to as the *dual* setting.

2.1 The problem with adding side information to tensor factorization

Consider a matrix factorization problem for $\mathbf{Y} \in \mathbb{R}^{n_1 \times n_2}$ with unknown latent factors $\mathbf{U} \in \mathbb{R}^{n_1 \times R}, \mathbf{V} \in \mathbb{R}^{n_2 \times R}$. We are approximating

$$\mathbf{Y} = \begin{bmatrix} y_{11} & \cdots & y_{1n_2} \\ \vdots & \ddots & \\ y_{n_1 1} & & y_{n_1 n_2} \end{bmatrix} \approx \mathbf{U} \cdot \mathbf{V}^\top = \begin{bmatrix} \sum_{r=1}^{R} u_{1r} v_{1r} & \cdots & \sum_{r=1}^{R} u_{1r} v_{n_2 r} \\ \vdots & \ddots & \\ \sum_{r=1}^{R} u_{n_1 r} v_{1r} & & \sum_{r=1}^{R} u_{n_1 r} v_{n_2 r} \end{bmatrix}. \tag{3}$$

If we update u_{1r} in the approximation $y_{11} \approx \sum_{r=1}^{R} u_{1r} v_{1r}$, we change the approximation $y_{1,2} \approx \sum_{r=1}^{R} u_{1r} v_{2r}$ since they share the parameter u_{1r}. However, $y_{2,1}$ and $y_{2,2}$ remain unchanged. Parameters are coupled across rows and columns but not globally. This is the standard setup in latent factorization.

Now consider the case where we have $\mathbf{D}_1 = \mathbf{D}_2 = \mathbf{1}_{n_1 \times n_1}$. We take our latent factors to be a linear function of available side information which leads $\mathbf{U}, \mathbf{V}$ to form $\mathbf{D}_1 \mathbf{U} = \begin{bmatrix} \sum_{i=1}^{n_1} u_{i1} & \cdots & \sum_{i=1}^{n_1} u_{iR} \\ \vdots & \ddots & \\ \sum_{i=1}^{n_1} u_{i1} & & \sum_{i=1}^{n_1} u_{iR} \end{bmatrix}$ and $\mathbf{D}_2 \mathbf{V}$ (similar form). It follows that

$$(\mathbf{D}_1 \mathbf{U}) \cdot (\mathbf{D}_2 \mathbf{V})^\top = \begin{bmatrix} \sum_{rij} u_{ir} v_{jr} & \cdots & \sum_{rij} u_{ir} v_{jr} \\ \vdots & \ddots & \\ \sum_{rij} u_{ir} v_{jr} & & \sum_{rij} u_{ir} v_{jr} \end{bmatrix}, \tag{4}$$

is a constant matrix! We have lost all model flexibility as we are approximating $\mathbf{Y}$ with a constant. Now consider a more realistic example with $\mathbf{D}_1 = \begin{bmatrix} d_{11} & \cdots & d_{1n_1} \\ \vdots & \ddots & \\ d_{n_1 1} & & d_{n_1 n_1} \end{bmatrix}$ and $\mathbf{D}_2 = \begin{bmatrix} z_{11} & \cdots & z_{1n_2} \\ \vdots & \ddots & \\ z_{n_2 1} & & z_{n_2 n_2} \end{bmatrix}$. In this case

$$(\mathbf{D}_1 \mathbf{U}) \cdot (\mathbf{D}_2 \mathbf{V})^\top = \begin{bmatrix} \sum_{rij} d_{1i} z_{1j} u_{ir} v_{jr} & \cdots & \sum_{rij} d_{1i} z_{n_2 j} u_{ir} v_{jr} \\ \vdots & \ddots & \\ \sum_{rij} d_{n_1 i} z_{1j} u_{ir} v_{jr} & & \sum_{rij} d_{n_1 i} z_{n_2 j} u_{ir} v_{jr} \end{bmatrix}. \tag{5}$$

Again, u_{ir} appears in all entries in our matrix approximation for $\mathbf{Y}$. However, this time changing u_{ir} will not change all entries by the same amount but rather differently across *all* entries depending on the entries of $\mathbf{D}_1$ and $\mathbf{D}_2$. This connects all entries in the

approximating matrix, introduces complex global variable dependence, and makes fitting infeasible at a large scale as the optimization updates globally. The observation applies in both primal and dual representations. In the primal representation, there is a restriction in expressiveness as the rank of our approximation falls off with the rank of the side information. In this setting, near-colinearity is also a problem as it leads to unstable optimization and factorizations which are very sensitive to noise.

We see that when we add side information we may inadvertently restrict the expressiveness of our model. We formulate a new tensor model with a range and dependence structure unaffected by the addition of side information in either the primal or dual setting.

3 Proposed approach

3.1 Tensors and side information

We seek a tensor model that benefits from additional side information while not forfeiting model flexibility. We introduce two strategies, which we call *weighted latent regression* (WLR) and *latent scaling* (LS).

3.1.1 Weighted latent regression

We now return to the previous setting but with additional latent tensors $\mathbf{U}' \in \mathbb{R}^{n_1 \times R}$ and $\mathbf{V}' \in \mathbb{R}^{n_2 \times R}$. We approximate $\mathbf{Y}$ as:

$$\begin{aligned}\mathbf{Y} &\approx ((\mathbf{D}_1\mathbf{U}) \circ \mathbf{U}') \cdot ((\mathbf{D}_2\mathbf{V}) \circ \mathbf{V}')^\top \\ &= \begin{bmatrix} \sum_{rij} u'_{1r} v'_{1r} d_{1i} z_{1j} u_{ir} v_{jr} & \cdots & \sum_{rij} u'_{1r} v'_{n_2 r} d_{1i} z_{n_2 j} u_{ir} v_{jr} \\ \vdots & \ddots & \\ \sum_{rij} u'_{n_1 r} v'_{1r} d_{1i} z_{1j} u_{ir} v_{jr} & & \sum_{rij} u'_{n_1 r} v'_{n_2 r} d_{1i} z_{1j} u_{ir} v_{jr} \end{bmatrix}\end{aligned} \tag{6}$$

By taking the Hadamard product ($\circ$) with additional tensors $\mathbf{U}'$, $\mathbf{V}'$ we recover the model flexibility and dependence structure of vanilla matrix factorization, as $\mathbf{U}'$ and $\mathbf{V}'$ are independent of $\mathbf{U}$ and $\mathbf{V}$. Here, changing u_{ir} would still imply a change for all entries by magnitudes defined by the side information, however we can calibrate these changes on a latent entrywise level by scaling each entry with $u'_{ir} v'_{jr}$. For any TT decomposition with an additional tensor $\mathbf{V}'_p$, the factorization becomes

$$\begin{aligned}\mathbf{Y} &\approx \prod_{p=1}^{P} \times_{-1} (\mathbf{V}'_p \circ (\mathbf{V}_p \times_2 \mathbf{D}_p)) \\ y_{i_1 \ldots i_P} &\approx \sum_{\substack{r_1 \ldots r_P \\ i'_1 \ldots i'_P \\ i''_1 \ldots i''_P}} \prod_{p=1}^{P} v'_{r_p i''_p r_{p-1}} v_{r_p i'_p r_{p-1}} d_{i_p, i'_p} \delta_{i_p}(i''_p)\end{aligned} \tag{7}$$

where $\delta(\cdot)$ denotes Kronecker delta. We interpret this as weighting the regression terms $\sum_{i,j} d_{pi} z_{qj} u_{ir} v_{jr}$ over indices p, q with $u_{pr} v_{qr}$ and then summing over latent indices r.

Interpretability We can decompose the estimate into weights of side information illustrated in Fig. 1. A guiding example on analyzing the latent factors is provided in Appendix C

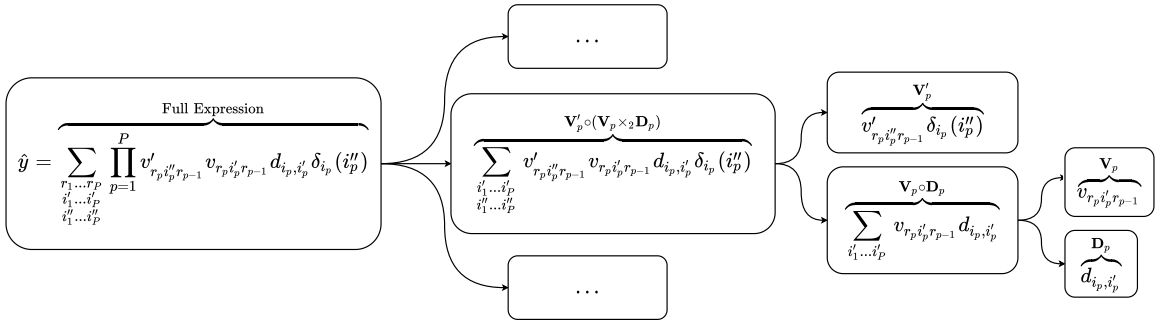


Fig. 1 Decomposing the estimate into interpretable weights

3.1.2 Latent scaling

An alternative computationally cheaper procedure would be to consider additional latent tensors $\mathbf{U}_1 \in \mathbb{R}^{n_1 \times K}, \mathbf{U}_2 \in \mathbb{R}^{n_1 \times L}, \mathbf{V}_1 \in \mathbb{R}^{n_2 \times K}, \mathbf{V}_2 \in \mathbb{R}^{n_2 \times L}$. We would approximate

$$\begin{aligned} \mathbf{Y} &\approx (\mathbf{U}_1 \mathbf{V}_1^\top) \circ ((\mathbf{D}_1 \mathbf{U}) \cdot (\mathbf{D}_2 \mathbf{V})^\top) + (\mathbf{U}_2 \mathbf{V}_2^\top) \\ &= \begin{bmatrix} \sum_q u^1_{1k} v^1_{1k} \sum_{rij} d_{1i} z_{1j} u_{ir} v_{jr} + \sum_l u^2_{1l} v^2_{1l} & \cdots \\ \vdots & \ddots \\ \sum_q u^1_{n_1 k} v^1_{1k} \sum_{rij} d_{1i} z_{1j} u_{ir} v_{jr} + \sum_l u^2_{n_1 l} v^2_{1l} & \end{bmatrix} \end{aligned} \tag{8}$$

We have similarly regained our original model flexibility where have introduced back independence for each term by scaling ($\mathbf{V}^s$) and adding a constant ($\mathbf{V}^b$) for each regression term with a 'latent scale and bias' term. We generalize this to

$$\begin{aligned} \mathbf{Y} &\approx \left(\prod_{p=1}^{P} \times_{-1} \mathbf{V}^s_p \right) \circ \left(\prod_{p=1}^{P} \times_{-1} (\mathbf{V}_p \times_2 \mathbf{D}_p) \right) + \left(\prod_{p=1}^{P} \times_{-1} \mathbf{V}^b_p \right) \\ y_{i_1 \dots i_P} &\approx \sum_{r^s_1 \dots r^s_P} \prod_{p=1}^{P} v^s_{r_p i_p r_{p-1}} \sum_{\substack{r_1 \dots r_P \\ i'_1 \dots i'_P}} \prod_{p=1}^{P} v_{r_p i'_p r_{p-1}} d_{i_p, i'_p} + \sum_{r^b_1 \dots r^b_P} \prod_{p=1}^{P} v^b_{r_p i_p r_{p-1}} \end{aligned} \tag{9}$$

One may conjecture that adding side information to tensors through a linear operation is counterproductive due to the restrictions it imposes on the approximation, and dispute that our proposal of introducing additional tensors to increase model flexibility is futile when side information is likely to be marginally informative or potentially uninformative. As an example, return to the case of completely non-informative constant side information, $\mathbf{D}_1 = \mathbf{1}, \mathbf{D}_2 = \mathbf{1}$. In this corner case, both our proposed models reduce to regular matrix factorization: the side information regression term collapses to a constant, which in conjunction with the added terms reduces to regular tensor factorization without side information.

Interpretability We can decompose the estimate into weights of side information illustrated in Fig. 2.

A comment on identifiability It should be noted that the proposed models need not be indentifiable.

To see this, return to the scenario where side information is constant. The term $\mathbf{V}_p \times_2 \mathbf{D}_p$ has constant rows equal to row sums of $\mathbf{V}_p$. Any transformation of $\mathbf{V}_p$ which preserves these row sums leaves the fit unchanged. However, in large-scale industrial

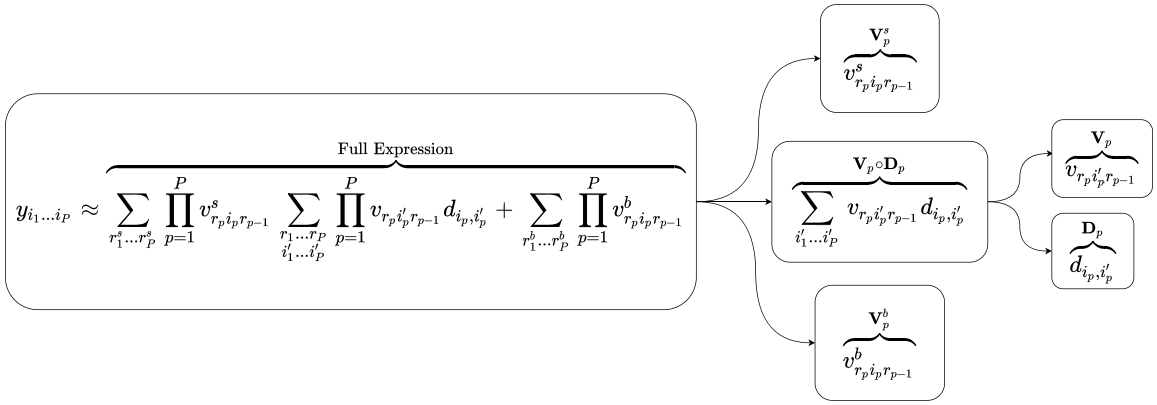


Fig. 2 Decomposing the estimate into interpretable weights

prediction, we draw utility from good generalization performance in prediction, and parameter identifiability is secondary.

3.2 Primal regularization terms

In weight space regression, the regularization terms are given by the squared Frobenius norm. The total regularization term would be written as

$$\Lambda = \sum_{p} \lambda_p \|\mathbf{V}_p\|_F^2 + \lambda'_p \|\mathbf{V}'_p\|_F^2 \tag{10}$$

3.3 RKHS and the representer theorem

We extend our framework to the RKHS dual space formalism, where our extension can intuitively be viewed as a tensorized version of kernel ridge regression (Vovk 2013). The merit of this is to enhance the performance by providing an implicit non-linear feature map of side information using kernels.

Firstly consider side information $\mathbf{D}_p = \{\mathbf{x}_i^p\}_{i=1}^{n_p}, \mathbf{x}_i^p \in \mathbb{R}^{c_p}$ which are kernelized using a kernel function $k : \mathbb{R}^{c_p} \times \mathbb{R}^{c_p} \to \mathbb{R}$. Denote $k_{ij}^p = k(\mathbf{x}_i^p, \mathbf{x}_j^p) \in \mathbb{R}$ and $\mathbf{K}_p = k(\mathbf{D}_p, \mathbf{D}_p) \in \mathbb{R}^{n_p \times n_p}$. Consider a $\mathbf{V} \in \mathbb{R}^{R_1 \times n_1 \cdots \times n_Q \times R_2}$, where $Q < P$. Using the Representer theorem (Schölkopf et al. 2001) we can express $\prod_{q=1}^{Q} \mathbf{V} \times_{q+1} \mathbf{K}_q$ as a function in RKHS

$$\mathsf{v}_{r_1, r_2} = \sum_{n_1 \ldots n_q} v_{r_1 n_1 \ldots n_q r_2} \prod_{q=1}^{Q} k_q(\cdot, \mathbf{x}_{n_q}^q). \tag{11}$$

where $\mathsf{v}_{r_1, r_2} : \mathbb{R}^{n_1} \times \cdots \times \mathbb{R}^{n_Q} \to \mathbb{R}$ and $\mathsf{v}_{r_1, r_2} \in \mathcal{H}$, which denotes the RKHS with respect to the kernels $\prod_q^Q \times k_q$. We use mode dot notation $\times_{q+1}$ here to apply kernelized side information $\mathbf{K}_q$ to each dimension of size $n_1 \ldots n_Q$, where $q + 1$ is used to account for the first dimension consisting of R_1.

3.4 Dual space regularization term for WLR

Consider applying another tensor with the same shape as $\mathbf{V}'$ through element wise product to robustify $\mathbf{V}$, i.e. $\mathbf{V}' \circ \prod_{q=1}^{Q} \mathbf{V} \times_{q+1} \mathbf{K}_q$. Then we have

$$\begin{aligned} \mathsf{v}_{r_1 r_2} &= \left(\sum_{n'_1 \dots n'_q} v'_{r_1 n'_1 \dots n'_q r_2} \prod_{q=1}^{Q} \delta_{\mathbf{x}^q_{n'_q}}(\cdot) \right) \cdot \left(\sum_{n_1 \dots n_q} v_{r_1 n_1 \dots n_q r_2} \prod_{q=1}^{Q} k(\cdot, \mathbf{x}^q_{n_q}) \right) \\ &= \sum_{\substack{n_1 \dots n_q \\ n'_1 \dots n_q}} v_{r_1 n_1 \dots n_q r_2} v'_{r_1 n'_1 \dots n'_q r_2} \prod_{q=1}^{Q} \delta_{\mathbf{x}^Q_{n'_q}}(\cdot) \prod_{q=1}^{Q} k(\cdot, \mathbf{x}^q_{n_q}) \end{aligned} \tag{12}$$

where the regularization term for v_{r_1,r_2} is given by:

$$\begin{aligned} \Lambda &= \sum_p \lambda_p \sum_{r_1,r_2} \langle \mathsf{v}_{r_1,r_2}, \mathsf{v}_{r_1,r_2} \rangle_{\mathcal{H}} \\ &= \sum_p \lambda_p \left(\left(\prod_{q=1}^{Q} \mathbf{V} \times_{q+1} \mathbf{K}_q \right) \circ \left((\prod_{q=1}^{Q} (\mathbf{V}' \circ \mathbf{V}') \times_{q+1} \mathbf{1}_{n_q \times n_q}) \circ \mathbf{V} \right) \right)_{++} \end{aligned} \tag{13}$$

and $(\cdot)_{++}$ means summing all elements.

3.5 Dual space regularization term for LS

For LS models, the regularization term is calculated as

$$\Lambda = \sum_p \lambda_p \left[\|\mathbf{V}^s_p\|_F^2 + \sum_r \langle \mathsf{v}_r, \mathsf{v}_r \rangle_{\mathcal{H}} + \|\mathbf{V}^b_p\|_F^2 \right] \tag{14}$$

where $\mathsf{v}_{r_1,r_2} = \sum_{n_1 \dots n_q} v_{r_p n_1 \dots n_q r_{p-1}} \prod_{q=1}^{Q} k(\cdot, \mathbf{x}^Q_{n_q})$ and $\sum_r \langle \mathsf{v}_r, \mathsf{v}_r \rangle_{\mathcal{H}} = \left(\left(\prod_{q=1}^{Q} \mathbf{V} \times_{q+1} \mathbf{K}_q \right) \circ \mathbf{V} \right)_{++}$.

3.6 Scaling with random Fourier features

To make tensors with kernelized side information scalable, we rely on a random fourier feature (Rahimi and Recht 2007) (RFFs) approximation of the true kernels. RFFs approximate a translation-invariant kernel function k using Monte Carlo:

$$\hat{k}(\mathbf{x}, \mathbf{y}) = \frac{2}{M} \sum_{i=1}^{M/2} \left[\cos(\omega_i^T \mathbf{x}) \cos(\omega_i^T \mathbf{y}) + \sin(\omega_i^T \mathbf{x}) \sin(\omega_i^T \mathbf{y}) \right] \tag{15}$$

where ω_i are frequencies drawn from a normalized non-negative spectral measure Λ of kernel k. Our primary goal in using RFFs is to create a memory efficient, yet expressive method. Thus, we write

$$\hat{k}\left(\mathbf{x}^p_i, \mathbf{x}^p_j\right) \approx \phi\left(\mathbf{x}^p_i\right)^{\top} \phi\left(\mathbf{x}^p_j\right) \tag{16}$$

with explicit feature map $\phi : \mathbb{R}^{D_p} \to \mathbb{R}^M$, and $M \ll N_p$. In the case of RFFs,

$$\phi(\cdot)^\top = [\cos(\omega_1^T \cdot), \dots, \cos(\omega_{M/2}^T \cdot), \sin(\omega_1^T \cdot), \dots, \sin(\omega_{M/2}^T \cdot)].$$

This feature map can be applied in the primal space setting as a computationally cheap alternative to the RKHS dual setting.

A drawback of tensors with kernelized side information is the $\mathcal{O}(N_p^2)$ memory growth of kernel matrices. If one of the dimensions has a large N_p in the dual space setting, we approximate large kernels $\mathbf{K}_p$ with

$$\mathbf{V} \times_{p+1} \boldsymbol{\Phi} \times_{p+1} \boldsymbol{\Phi}^\top \approx \mathbf{V} \times_{p+1} \mathbf{K}_p, \tag{17}$$

where $\boldsymbol{\Phi} = \phi(\mathbf{D}_p) \in \mathbb{R}^{N_p \times M}$. To see that this is a valid approximation, an element $\bar{v}_{i_1 \dots i_p}$ in $\mathbf{V} \times_p \mathbf{K}_p$ is given by $\bar{v}_{i_1 \dots i_p} = \sum_{i'_p=1}^{N_p} v_{i_1 \dots i'_p} k(\mathbf{x}_{i_p}^p, \mathbf{x}_{i'_p}^p)$. Using RFFs we have

$$\bar{v}_{i_1 \dots i_p} = \sum_{i'_p=1}^{N_p} \sum_{c=1}^{M} v_{i_1 \dots i'_p} \boldsymbol{\Phi}_{i'_p c} \boldsymbol{\Phi}_{i_p c} = \sum_{i'_p=1}^{N_p} v_{i_1 \dots i'_p} \phi(\mathbf{x}_{i'_p}^p)^\top \phi(\mathbf{x}_{i_p}^p). \tag{18}$$

KFT with RFFs now becomes:

$$\mathsf{v}_{r_p} = \sum_{\substack{n_1 \dots n_q \\ n'_1 \dots n_q}} v_{r_p n_1 \dots n_q r_{p-1}} v'_{r_p n'_1 \dots n'_q r_{p-1}} \prod_{q=1}^{Q} \delta_{\mathbf{x}_{n'_q}^Q}(\cdot) \prod_{q=1}^{Q} \phi(\mathbf{x}_{n_q}^Q) \phi(\cdot) \tag{19}$$

with the regularization term

$$\sum_r \langle \mathsf{v}_r, \mathsf{v}_r \rangle_{\mathcal{H}} = \left(\left(\prod_{q=1}^{Q} \mathbf{V} \times_{q+1} \boldsymbol{\Phi}_q \times_q \boldsymbol{\Phi}_q^\top \right) \circ \left(\left(\prod_{q=1}^{Q} (\mathbf{V}' \circ \mathbf{V}') \times_q \mathbf{1}_{n_q \times n_q} \right) \circ \mathbf{V} \right) \right)_{++}. \tag{20}$$

For a derivation, please refer to the Appendix B.

3.7 Kernel fried tensor

Having established our new model, we coin it kernel fried tensor (KFT). Given some loss $\mathcal{L}(\mathbf{Y}, \tilde{\mathbf{Y}})$ for predictions $\tilde{\mathbf{Y}}$ the full objective is

$$\min_{\text{wrt } \mathbf{V}_p, \mathbf{V}'_p, \Theta_p} \mathcal{L}(\mathbf{Y}, \tilde{\mathbf{Y}}) + \Lambda(\mathbf{V}_p, \mathbf{V}'_p, \Theta_p) \tag{21}$$

where $\mathbf{V}_p, \mathbf{V}'_p, \Theta_p$ are parameters of the model and Θ_p are kernel parameters if we use the RKHS dual formulation. As our proposed model involves mutually dependent components with non-zero mixed partial derivatives, optimizing them jointly with a first order solver is inappropriate as mixed partial derivatives will not be considered during each gradient step. Inspired by the EM-algorithm (Dempster et al. 1977), we summarize our training procedure in Algorithm 1. By updating each parameter group sequentially and independently, we eliminate the effects of mixed partials leading to accurate gradient updates. For further details, we refer to the Appendix E.1.

Algorithm 1 Tensor training procedure

Require: $\{\mathbf{V}_p\}_{p=1}^{P}, \{\mathbf{V}'_p\}_{p=1}^{P}, \{\Theta_p\}_{p=1}^{P}$, Data, learning rate α_{lr}
  for epochs **do**
    for $p = 1, \ldots, P$ **do**
      Fix all parameters except $\mathcal{G}_p = \{\mathbf{V}_p, \mathbf{V}'_p, \Theta_p\}$
      for batch in data **do**
        Calculate $\mathcal{L}(\mathbf{Y}, \tilde{\mathbf{Y}}) + \Lambda$
        Update $\mathcal{G}_p$ as $\mathcal{G}_p = \mathcal{G}_p - \alpha_{\text{lr}} \frac{\partial(\mathcal{L}+\Lambda)}{\partial \mathcal{G}_p}$
      end for
    end for
  end for

3.7.1 Joint features

From a statistical point of view, we are assuming that each of our latent tensors $\mathbf{V}_p$ factorizes $\mathbf{Y}$ into P independent components with prior distribution corresponding to $\mathcal{N}(\mathrm{v}^p_{r_{p-1}r_p}|0, \mathbf{K}_p)$, where $\mathrm{v}^p_{r_{p-1}r_p} \in \mathbb{R}^{n_p}$ is the r_{p-1}, r_p cell selected from $\mathbf{V}_p$. We can enrich our approximation by jointly modelling some dimensions p by choosing some $\mathbf{V}_p \in \mathbb{R}^{R_{p+1} \times n_{p+1} \times n_p \times R_{p-1}}$. If we denote this dimension p by p' we have that

$$
\begin{aligned}
\mathbf{Y} &\approx \prod_{p=1}^{p'-1} \times_{-1} (\mathbf{V}_p \times_2 \mathbf{K}_p) \times_{-1} (\mathbf{V}_{p'} \times_3 \mathbf{K}_{p'} \times_2 \mathbf{K}_{p'+1}) \prod_{p=p'+2}^{P} \times_{-1} (\mathbf{V}_p \times_2 \mathbf{K}_p) \\
y_{i_1 \ldots i_P} &\approx \sum_{\substack{r_1 \ldots r_P \\ i'_1 \ldots i'_P}} \prod_{p=1}^{p'-1} v_{r_p i'_p r_{p-1}} k(\mathbf{x}^p_{i_p}, \mathbf{x}^p_{i'_p}) \cdot v_{r_p i'_{p'} i'_{p'+1} r_{p-1}} \\
& k(\mathbf{x}^{p'}_{i_{p'}}, \mathbf{x}^{p'}_{i'_{p'}}) k\left(\mathbf{x}^{p'+1}_{i_{p'+1}}, \mathbf{x}^{p'+1}_{i'_{p'+1}}\right) \prod_{p=1}^{p'+2} v_{r_p i'_p r_{p-1}} k(\mathbf{x}^p_{i_p}, \mathbf{x}^p_{i'_p})
\end{aligned}
\tag{22}
$$

and the prior would instead be given as $\mathcal{N}(\mathrm{vec}(\mathrm{v}^{p'}_{r_{p'-1}r_{p'+1}})|0, \mathbf{K}_{p'} \otimes \mathbf{K}_{p'+1})$. Here $\mathrm{vec}(\mathrm{v}^{p'}_{r_{p'-1}r_{p'+1}}) \in \mathbb{R}^{n_{p'} n_{p'+1}}$ and $\mathrm{vec}(\cdot)$ means flattening a tensor to a vector. The cell selected from $\mathbf{V}_{p'}$ now has a dependency between dimensions p' and $p'+1$. We refer to a one dimensional factorization component of TT as a *TT-core* and a multi dimensional factorization component as a *joint TT-core*.

3.8 Bayesian inference

We turn to Bayesian inference for uncertainty quantification with KFT. Assume a Gaussian conditional likelihood for an observation $y_{i_1 \ldots i_P}$ with inspiration from Gönen et al. (2013) Kim and Choi (2014). For KFT-WLR we have that

$$p(y_{i_1 \dots i_P} | \mathbf{V}_1 \dots \mathbf{V}_P, \mathbf{V}'_1 \dots \mathbf{V}'_P) = \mathcal{N}\left(y_{i_1 \dots i_P} \middle| \sum_{\substack{r_1 \dots r_P \\ i'_1 \dots i'_P \\ i''_1 \dots i''_P}} \prod_{p=1}^{P} v'_{r_p i''_p r_{p-1}} v_{r_p i'_p r_{p-1}} k(\mathbf{x}^p_{i_p}, \mathbf{x}^p_{i'_p}) \delta_{\mathbf{x}^p_{i_p}}(\mathbf{x}^p_{i''_p}), \sigma_y^2 \right). \tag{23}$$

The corresponding objective for KFT-LS is

$$p(y_{i_1 \dots i_P} | \mathbf{V}_1 \dots \mathbf{V}_P, \mathbf{V}'_1 \dots \mathbf{V}'_P) = \mathcal{N}\left(y_{i_1 \dots i_P} \middle| \sum_{r^s_1 \dots r^s_P} \prod_{p=1}^{P} v^s_{r_p i_p r_{p-1}} \sum_{\substack{r_1 \dots r_P \\ i'_1 \dots i'_P}} \prod_{p=1}^{P} v_{r_p i'_p r_{p-1}} k(\mathbf{x}^p_{i_p}, \mathbf{x}^p_{i'_p}) + \sum_{r^b_1 \dots r^b_P} \prod_{p=1}^{P} v^b_{r_p i_p r_{p-1}}, \sigma_y^2 \right) \tag{24}$$

where σ_y^2 is a scalar hyperparameter.

3.9 Variational approximation

Our goal is to maximize the posterior distribution $p(\mathbf{V}_p, \mathbf{V}'_p | \mathbf{Y})$ which is intractable as the likelihood $p(\mathbf{Y}) = \int p(\mathbf{Y} | \mathbf{V}_1 \dots \mathbf{V}_P, \mathbf{V}'_1 \dots \mathbf{V}'_P) p(\mathbf{V}_1) \dots p(\mathbf{V}_P) p(\mathbf{V}'_1) \dots p(\mathbf{V}'_P) d\mathbf{V}_{1 \dots P} d\mathbf{V}'_{1 \dots P}$ does not have a closed form solution due to the product of Gaussians. Instead we use variational approximations for $\mathbf{V}_p, \mathbf{V}'_p$ by parametrizing distributions of the Gaussian family and optimize the evidence lower bound (ELBO)

$$\mathcal{L}(y_{i_1 \dots i_P}, \mathbf{V}_1 \dots \mathbf{V}_P, \mathbf{V}'_1 \dots \mathbf{V}'_P) = \mathbb{E}_q[\log p(y_{i_1 \dots i_P} | \mathbf{V}_1 \dots \mathbf{V}_P, \mathbf{V}'_1 \dots \mathbf{V}'_P)] - \left(\sum_{p=1}^{P} D_{\mathrm{KL}}(q(\mathbf{V}_p) \| p(\mathbf{V}_p)) + D_{\mathrm{KL}}(q(\mathbf{V}'_p) \| p(\mathbf{V}'_p)) \right) \tag{25}$$

In our framework, we consider the univariate Gaussian and multivariate Gaussian as variational approximations with corresponding priors where σ_y^2 is interpreted to control the weight of the reconstruction term against the KL-term.

3.9.1 Univariate VI

Univariate KL For the case of univariate normal priors, we calculate the KL divergence as

$$D_{\mathrm{KL}}(\mathcal{N}_q \,\|\, \mathcal{N}_p) = \frac{(\mu_q - \mu_p)^2}{2\sigma_p^2} + \frac{1}{2}\left(\frac{\sigma_q^2}{\sigma_p^2} - 1 - \ln\frac{\sigma_q^2}{\sigma_p^2}\right) \tag{26}$$

where μ_q, μ_p and σ_q^2, σ_p^2 are the mean and variance for the variational approximation and prior respectively, where μ_p, σ_p^2 are chosen a priori.

Model For a univariate Gaussian variational approximation we assume the following prior structure

$$p(\mathbf{V}'_p) = \prod_{r_p, i_p} \mathcal{N}(v'_{r_p i_p r_{p-1}} | \mu'_p, \sigma'^2_p), \quad p(\mathbf{V}_p) = \prod_{r_p, i_p} \mathcal{N}(v_{r_p i_p r_{p-1}} | \mu_p, \sigma_p^2) \tag{27}$$

with corresponding univariate meanfield approximation

$$\begin{aligned} q(\mathbf{V}'_p) &= \prod_{r_p, i_p} \mathcal{N}(v'_{r_p i_p r_{p-1}} | \mu'_{r_p i_p r_{p-1}}, \sigma'^2_{r_p i_p r'_{p-1}}), \\ q(\mathbf{V}_p) &= \prod_{r_p, i_p} \mathcal{N}(v_{r_p i_p r_{p-1}} | \mu_{r_p i_p r_{p-1}}, \sigma^2_{r_p i_p r_{p-1}}). \end{aligned} \tag{28}$$

We take $\mu'_p, \sigma'^2_p, \mu_p, \sigma_p^2$ to be hyperparameters.

Weighted latent regression reconstruction term For Weighted latent regression, we express the reconstruction term as

$$\begin{aligned} &\mathbb{E}_q[\log p(y_{i_1 \dots i_P} | \mathbf{V}_1 \dots \mathbf{V}_P, \mathbf{V}'_1 \dots \mathbf{V}'_P)] \\ &\propto \frac{1}{\sigma_y^2}\left[\mathbf{Y}^2 - 2\mathbf{Y} \circ \left(\prod_{p=1}^{P} \times_{-1} (\mathbf{M}'_p \circ (\mathbf{M}_p \times_2 \mathbf{K}_p))\right) + \left(\prod_{p}^{P} \times_{-1} \mathbf{M}'_p \circ (\mathbf{M}_p \times_2 \mathbf{K}_p)\right)^2 \right. \\ &\quad + \left(\prod_{p}^{P} \times_{-1} \Sigma'_p \circ (\Sigma_p \times_2 (\mathbf{K}_p)^2)\right) + \left(\prod_{p}^{P} \times_{-1} (\Sigma'_p \circ (\mathbf{M}_p \times_2 \mathbf{K}_p)^2)\right) \\ &\quad \left. + \left(\prod_{p}^{P} \times_{-1} \mathbf{M}'^2_p \circ (\Sigma_p \times_2 (\mathbf{K}_p)^2)\right)\right] \end{aligned} \tag{29}$$

where $\mathbf{M}'_p, \mathbf{M}_p$ and Σ'_p, Σ_p correspond to the tensors containing the variational parameters $\mu'_{r_p i_p r_{p-1}}$, $\mu_{r_p i_p r_{p-1}}$ and $\sigma'^2_{r_p i_p r_{p-1}}$, $\sigma^2_{r_p i_p r_{p-1}}$ respectively. For the case of RFF's, we approximate $\Sigma_p \times_2 (\mathbf{K}_p)^2 \approx \Sigma_p \times_2 (\boldsymbol{\Phi}_p \bullet \boldsymbol{\Phi}_p)^\top \times_2 (\boldsymbol{\Phi}_p \bullet \boldsymbol{\Phi}_p)$, where $\bullet$ is the transposed Khatri–Rao product. It should further be noted that any square term means element wise squaring. We provide a derivation in the Appendix A.1.

Latent scaling reconstruction term For Latent Scaling, we express the reconstruction term as

$$
\begin{aligned}
&\mathbb{E}_q[\log p(y_{i_1 \ldots i_P} | \mathbf{V}_1 \ldots \mathbf{V}_P, \mathbf{V}'_1 \ldots \mathbf{V}'_P)] \\
&\propto \frac{1}{\sigma_y^2}\Bigg[\mathbf{Y}^2 - 2\mathbf{Y} \circ \Bigg(\prod_{p=1}^{P} (\mathbf{M}_p^s \circ (\mathbf{M}_p \times_2 \mathbf{K}_p) + \mathbf{M}_p^b)\Bigg) \\
&\quad + \Bigg(\Bigg(\prod_{p=1}^{P} \times_{-1} \mathbf{M}_p^s\Bigg)^2 + \Bigg(\prod_{p=1}^{P} \times_{-1} \Sigma_p^s\Bigg)\Bigg) \\
&\quad \circ \Bigg(\Bigg(\prod_{p}^{P} \times_{-1} \mathbf{M}_p \times_2 \mathbf{K}_p\Bigg)^2 + \prod_{p=1}^{P} \times_{-1} \Sigma_p \times_2 (\mathbf{K}_p)^2\Bigg) \\
&\quad + 2\Bigg(\prod_{p=1}^{P} \times_{-1} \mathbf{M}_p^s\Bigg) \circ \Bigg(\prod_{p=1}^{P} \times_{-1} \mathbf{M}_p^b\Bigg) \circ \Bigg(\prod_{p=1}^{P} \times_{-1} \mathbf{M}_p \times_2 \mathbf{K}_p\Bigg) \\
&\quad + \Bigg(\Bigg(\prod_{p=1}^{P} \times_{-1} \mathbf{M}_p^b\Bigg)^2 + \Bigg(\prod_{p=1}^{P} \times_{-1} \Sigma_p^b\Bigg)\Bigg)\Bigg].
\end{aligned}
\tag{30}
$$

For details, see the Appendix A.2.

3.9.2 Multivariate VI

Multivariate KL The KL divergence between a multivariate normal prior p and a variational approximation given by q:

$$
D_{\mathrm{KL}}(\mathcal{N}_q \parallel \mathcal{N}_p) = \frac{1}{2}\left[\mathrm{tr}\left(\Sigma_p^{-1}\Sigma_q\right) + (\mu_p - \mu_q)^{\mathsf{T}}\Sigma_p^{-1}(\mu_p - \mu_q) - k + \ln\left(\frac{\det \Sigma_p}{\det \Sigma_q}\right)\right] \tag{31}
$$

Where μ_p, μ_q and Σ_p, Σ_q are the mean and covariance for the prior and variational respectively. Inspired by g-prior (Zellner 1986), we take $\Sigma_p = \mathbf{K}_p^{-1}$, where $\mathbf{K}_p^{-1}$ is the inverse kernel covariance matrix of side information for mode p. When side information is absent, we take $\Sigma_p = \mathbf{I}$. Another benefit of using the inverse is that is simplifies calculations, since we now avoid inverting a dense square matrix in the KL-term. Similar to the univariate case we choose μ_p a priori, although here it becomes a constant tensor rather than a constant scalar.

Model For the multivariate case, we consider the following priors

$$
\begin{aligned}
p(\mathbf{V}'_p) &= \prod_{r_p, i_p} \mathcal{N}(v'_{r_p i_p r_{p-1}} | \mu'_p, \sigma'^2_p) \\
p(\mathbf{V}_p) &= \prod_{r_p} \mathcal{N}(\mathrm{vec}(\mathsf{v}_{r_p}) | \mu_p, \prod_{q=1}^{Q_p} \otimes (\mathbf{K}_q)^{-1})
\end{aligned}
\tag{32}
$$

Where Q_p is the number of dimensions jointly modeled in each TT-core. For the variational approximations, we have

$$
\begin{aligned}
q(\mathbf{V}'_p) &= \prod_{r_p, i_p} \mathcal{N}(v'_{r_p i_p r_{p-1}} | \mu'_{r_p i_p r_{p-1}}, \sigma'^2_{r_p i_p r_{p-1}}) \\
q(\mathbf{V}_p) &= \prod_{r_p, i_p} \mathcal{N}(\text{vec}(\mathbf{v}_{r_p}) | \mu_{r_p i_p r_{p-1}}, \prod_{q=1}^{Q_p} \otimes(\mathbf{B}_q \mathbf{B}_q^\top))
\end{aligned}
\tag{33}
$$

We take $\mu'_p, \sigma'^2_p, \mu_p$ to be hyperparameters and $\Sigma_q = \mathbf{B}_q \mathbf{B}_q^\top$.

Sampling and parametrization Calculating $\prod_{q=1}^{Q_i} \otimes \mathbf{B}_q \mathbf{B}_q^\top$ directly will yield a covariance matrix that is prohibitively large. To sample from $q(\mathbf{V}_p)$ we exploit that positive definite matrices A and B with their Cholesky decompositions L_A and L_B have the following property

$$
A \otimes B = (L_A L_A^\top) \otimes (L_B L_B^\top) = (L_A \otimes L_B)(L_A \otimes L_B)^\top. \tag{34}
$$

together with the fact that

$$
\prod_{i=1}^{N} \mathbf{X} \times_{i+1} A_i = \left(\prod_{i=1}^{N} \otimes A_i \right) \cdot \text{vec}(\mathbf{X}), \tag{35}
$$

where $\text{vec}(\mathbf{X}) \in \mathbb{R}^{\prod_i^N I_i \times R}$. We would then draw a sample $\mathbf{b} \sim q(\mathbf{V})$ as

$$
\mathbf{b} = \mu_{r_p} + \prod_{q=1}^{Q_i} \tilde{\mathbf{z}} \times_{q+1} \mathbf{B}_q \tag{36}
$$

where $\tilde{\mathbf{z}} \sim \mathcal{N}(0, \mathbf{I}_{\prod_{p=1}^{P} n_p})$ is reshaped into $\tilde{\mathbf{z}} \in \mathbb{R}^{\prod_{q=1} \times n_q}$. We take $\mathbf{B}_q = \mathbf{ltri}(B_q B_q^\top) + D_q$ (Ong et al. 2017), where $B_q \in \mathbb{R}^{n_q \times r}$, D_q to be a diagonal matrix and **ltri** denotes taking the lower triangular component of a square matrix including the diagonal. We choose this parametrization for a linear time-complexity calculation of the determinant in the KL-term by exploiting that $\det\left(\Sigma_q\right) = \det\left(\mathbf{B}_q \mathbf{B}_q^\top\right) = (\det\left(\mathbf{B}_q\right))^2$. In the RFF case, we take $\mathbf{B}_q = B_q$ and estimate the covariance as $\mathbf{B}_q \mathbf{B}_q^\top + D_q^2$

Weighted latent regression reconstruction term We similarly to the univariate case express the reconstruction term as

$$
\begin{aligned}
&\mathbb{E}_q[\log p(y_{i_1 \ldots i_P} | \mathbf{V}_1 \ldots \mathbf{V}_P, \mathbf{V}'_1 \ldots \mathbf{V}'_P)] \\
&\propto \frac{1}{\sigma_y^2} \Bigg[\mathbf{Y}^2 - 2\mathbf{Y} \circ \left(\prod_{p=1}^{P} \times_{-1} (\mathbf{M}'_p \circ (\mathbf{M}_p \times_2 \mathbf{K}_p)) \right) + \left(\prod_{p}^{P} \times_{-1} \mathbf{M}'_p \circ (\mathbf{M}_p \times_2 \mathbf{K}_p) \right)^2 \\
&\quad + \left(\prod_{p}^{P} \times_{-1} \Sigma'_p \circ (\mathbf{1} \times_2 \left(\text{diag}((\mathbf{K}_p \cdot \mathbf{B}_p)^2 \cdot \bar{1}) \right) \right) + \left(\prod_{p}^{P} \times_{-1} (\Sigma'_p \circ (\mathbf{M}_p \times_2 \mathbf{K}_p)^2) \right) \\
&\quad + \left(\prod_{p}^{P} \times_{-1} \mathbf{M}'^2_p \circ (\mathbf{1} \times_2 \left(\text{diag}((\mathbf{K}_p \cdot \mathbf{B}_p)^2 \cdot \bar{1}) \right) \right) \Bigg]
\end{aligned}
\tag{37}
$$

where $\Sigma_p = \mathbf{B}_p \mathbf{B}_p^T$, $\mathbf{1}$ denotes a constant one tensor with the same dimensions as Σ'_p, $\bar{1} \in \mathbb{R}^{R \times 1}$ where R is the column dimension of $\mathbf{B}_p$ and Σ'_p is the same as in the univariate case. For RFF's we have that

$$
\begin{aligned}
(\mathbf{K}_p \cdot \mathbf{B}_p)^2 \cdot \bar{1} \approx & ((\boldsymbol{\Phi}_p \cdot \boldsymbol{\Phi}_p^\top) \cdot \mathbf{B}_p)^2 \cdot \bar{1} + \text{vec}(D_p^2) \\
= & (\boldsymbol{\Phi}_p \cdot (\boldsymbol{\Phi}_p^\top \cdot \mathbf{B}_p))^2 \cdot \bar{1} + \text{vec}(D_p^2).
\end{aligned} \tag{38}
$$

Latent scaling reconstruction term The latent scaling version has the following expression

$$
\begin{aligned}
& \mathbb{E}_q[\log p(y_{i_1 \dots i_p} | \mathbf{V}_1 \dots \mathbf{V}_P, \mathbf{V}'_1 \dots \mathbf{V}'_P)] \\
& \propto \frac{1}{\sigma_y^2} \Bigg[\mathbf{Y}^2 - 2\mathbf{Y} \circ \Bigg(\prod_{p=1}^{P} (\mathbf{M}_p^s \circ (\mathbf{M}_p \times_2 \mathbf{K}_p) + \mathbf{M}_p^b) \Bigg) \\
& \quad + \Bigg(\Bigg(\prod_{p=1}^{P} \times_{-1} \mathbf{M}_p^s \Bigg)^2 + \Bigg(\prod_{p=1}^{P} \times_{-1} \Sigma_p^s \Bigg) \Bigg) \\
& \quad \circ \Bigg(\Bigg(\prod_{p}^{P} \times_{-1} \mathbf{M}_p \times_2 \mathbf{K}_p \Bigg)^2 + \prod_{p=1}^{P} \times_{-1} (\mathbf{1} \times_2 (\text{diag}((\mathbf{K}_p \cdot \mathbf{B}_p)^2 \cdot \bar{1}))) \Bigg) \\
& \quad + 2 \Bigg(\prod_{p=1}^{P} \times_{-1} \mathbf{M}_p^s \Bigg) \circ \Bigg(\prod_{p=1}^{P} \times_{-1} \mathbf{M}_p^b \Bigg) \circ \Bigg(\prod_{p=1}^{P} \times_{-1} \mathbf{M}_p \times_2 \mathbf{K}_p \Bigg) \\
& \quad + \Bigg(\Bigg(\prod_{p=1}^{P} \times_{-1} \mathbf{M}_p^b \Bigg)^2 + \Bigg(\prod_{p=1}^{P} \times_{-1} \Sigma_p^b \Bigg) \Bigg) \Bigg].
\end{aligned} \tag{39}
$$

For details, see the Appendix A.2.

RFFs and KL divergence Using $(\boldsymbol{\Phi}_p \boldsymbol{\Phi}_p^\top)^{-1} \approx (\mathbf{K}_p)^{-1}$ as our prior covariance, we observe that the KL-term presents computational difficulties as a naive approach would require storing $(\mathbf{K}_p)^{-1} \in \mathbb{R}^{n_p \times n_p}$ in memory. Assuming we take $\Sigma_p = BB^\top, B \in \mathbb{R}^{n_p \times R}$, we can manage the first term by using the equivalence

$$
(A \bullet B) \cdot (A \bullet B)^\top = (AA^\top) \circ (BB^\top). \tag{40}
$$

Consequently, we have that

$$
\text{tr}(\Sigma_p^{-1} \Sigma_q) = \Bigg(\underbrace{(\boldsymbol{\Phi}\boldsymbol{\Phi}^\top)}_{\text{Using } (\mathbf{K}_p)^{-1}} \circ BB^\top \Bigg)_{++} = \Big((\boldsymbol{\Phi} \bullet B)^\top \cdot (\boldsymbol{\Phi} \bullet B) \Big)_{++}. \tag{41}
$$

We can calculate the second term using (34) and (35). For the third term, we remember Weinstein–Aronszajn's identity

$$
\det(I_m + AB) = \det(I_n + BA) \tag{42}
$$

where $A \in \mathbb{R}^{m \times n}, B \in \mathbb{R}^{n \times m}$ and AB is trace class. If we were to take our prior covariance matrix to be $\Sigma_p = (\boldsymbol{\Phi}_p \boldsymbol{\Phi}_p^\top + I_{n_p})^{-1} \approx (\mathbf{K}_p + I_{n_p})^{-1}$ and our posterior covariance matrix to be approximated as $\Sigma_q = BB^\top + I_{n_p}$, we could use Weinstein–Aronszajn's identity to calculate the third log term in a computationally efficient manner.

From a statistical perspective, adding a diagonal to the covariance matrix implies regularizing it by increasing the diagonal variance terms. Taking inspiration from Kim and Teh (2017), we can further choose the magnitude σ of the regularization

$$\begin{aligned}\det(\sigma^2 I_n + \boldsymbol{\Phi}\boldsymbol{\Phi}^\top) &= (\sigma^2)^n \det(I_n + \sigma^{-2}\boldsymbol{\Phi}\boldsymbol{\Phi}^\top) \\ &= (\sigma^2)^n \det(I_m + \sigma^{-2}\boldsymbol{\Phi}^\top\boldsymbol{\Phi}) \\ &= (\sigma^2)^{n-m} \det(\sigma^2 I_m + \boldsymbol{\Phi}^\top\boldsymbol{\Phi})\end{aligned}$$

The KL expression then becomes

$$\begin{aligned}D_{\mathrm{KL}}(\mathcal{N}_q \parallel \mathcal{N}_p) = \frac{1}{2}\Bigg[&\left((\boldsymbol{\Phi} \bullet B)^\top \cdot (\boldsymbol{\Phi} \bullet B)\right)_{++} + \sigma_p^2 (B \circ B)_{++} \\ &+ \sigma_q^2 (\boldsymbol{\Phi} \circ \boldsymbol{\Phi})_{++} + \sigma_q^2 \sigma_p^2 N + (\mu_p - \mu_q)^\top (\boldsymbol{\Phi}\boldsymbol{\Phi}^\top + \sigma_p^2 I)(\mu_p - \mu_q) \\ &- k + \ln\left(\frac{|(\sigma_p^2)^{N-I} \det(\boldsymbol{\Phi}^\top\boldsymbol{\Phi} + \sigma_p^2 I)|^{-1}}{|(\sigma_q^2)^{N-I} \det(B^\top B + \sigma_q^2 I)|}\right)\Bigg].\end{aligned} \tag{43}$$

3.9.3 Calibration metric

We evaluate the overall calibration of our variational model using the sum

$$\Xi = \sum_{\alpha \in \Omega} |\xi_{1-2\alpha} - (1 - 2\alpha)| \tag{44}$$

of the *calibration rate* $\xi_{1-2\alpha}$, which we define as

$$\xi_{1-2\alpha} = \frac{\text{number of } y_{i_1 \ldots i_P} \text{ within } 1 - 2\alpha \text{ confidence level}}{\text{total number of } y_{i_1 \ldots i_P}} \tag{45}$$

where we consider α to take values in $\{0.05, 0.15, 0.25, 0.35, 0.45\}$. This calibration rate can be understood as the true (frequentist) coverage probability and $1 - 2\alpha$ as the nominal coverage probability. The model is calibrated when the true coverage probability is close to the nominal coverage probability, that is when Ξ is small. To ensure that our model finds a meaningful variational approximation, we take our hyperparameter selection criteria to be:

$$\eta_{\text{criteria}} = \Xi - R^2 \tag{46}$$

where $R^2 := 1 - \frac{\sum_{i_1 \ldots i_n} (y_{i_1 \ldots i_n} - \hat{y}_{i_1 \ldots i_n})^2}{\mathrm{Var}(\mathbf{Y})}$ is the coefficient of determination (Draper and Smith 1966) calculated using the "mean terms" $\left(\prod_{p=1}^P \times_{-1} (\mathbf{M}'_p \circ (\mathbf{M}_p \times_2 \mathbf{K}_p))\right)$ or $\left(\prod_{p=1}^P (\mathbf{M}^s_p \circ (\mathbf{M}_p \times_2 \mathbf{K}_p) + \mathbf{M}^b_p)\right)$ as predictions $\hat{y}_{i_1 \ldots i_n}$. If we only use $\eta_{\text{criteria}} = \Xi$, we argue that there is an inductive bias in choosing α's which may lead to an approximation that is calibrated per se, but not meaningful as the modes are incorrect (low R^2 value). Similarly to the frequentist case, we use an EM-inspired optimization strategy in Algorithm 2. The main idea is to find the mode and variance parameters of our variational approximation in a mutually exclusive sequential order, starting with the modes. Similar to the frequentist case, the reconstruction term of the ELBO has terms that both contain Σ_p and $\mathbf{M}'_p$ which motivates the EM-inspired approach. For further details, please refer to the Appendix E.2.

Algorithm 2 Bayesian tensor training procedure

Require: $\{\mathbf{M}_p\}_{p=1}^P, \{\mathbf{M}'_p\}_{p=1}^P, \{\Sigma_p\}_{p=1}^P, \{\Sigma'_p\}_{p=1}^P, \{\Theta_p\}_{p=1}^P$, Data, learning rate α_{lr}

for epochs **do**

  for parameter group $\mathcal{Z}$ in $\left\{ \left(\{\mathbf{M}_p\}_{p=1}^P, \{\mathbf{M}'_p\}_{p=1}^P, \{\Theta_p\}_{p=1}^P\right), \left(\{\Sigma_p\}_{p=1}^P, \{\Sigma'_p\}_{p=1}^P\right) \right\}$ **do**

    for sub-parameter group $\mathcal{G}$ in $\mathcal{Z}$ **do**

      for $p = 1, \ldots, P$ **do**

        Fix all parameters except $\mathcal{G}_p \subset \mathcal{G}$

        for batch in data **do**

          Calculate ELBO $\mathcal{L}_{\text{ELBO}}$

          Update $\mathcal{G}_p$ as $\mathcal{G}_p = \mathcal{G}_p - \alpha_{\text{lr}} \frac{\partial \mathcal{L}_{\text{ELBO}}}{\partial \mathcal{G}_p}$

        end for

      end for

    end for

  end for

end for

3.10 Extension to forecasting

KFT in its current form is fundamentally unable to accommodate forecasting problems. To see this, we first consider the forecasting problem of predicting observation y_T from previous observations $\mathbf{y}_t = [y_0, \ldots, y_t], \quad 0 < t < T$ using model $f(\mathbf{y}_t)$. The model is then optimized by minimizing

$$\mathcal{L}(y_T, \mathbf{y}_t) = \|y_T - f(\mathbf{y}_t)\|^2 \tag{47}$$

for all T. Forecasting problems assume that the model does not have access to future y_{T+1} outside the training set and assumes it to learn y_{T+1} through an autoregressive assumption. Imposing this assumption on KFT would imply that latent factorizations for time-indexed $T + 1$ would remain untrained with the current training procedure, as we do not access to these indices during training. With untrained latent factorizations for $T + 1$, any forecast would at best be random.

However, we can easily extend KFT to be autoregressive by directly applying (Yu et al. 2016).

3.10.1 Frequentist setting

We consider the Temporal Regularized Matrix Factorization (TRMF) framework presented in Yu et al. (2016)

$$\mathcal{T}_{\text{AR}}(\mathbf{X} \mid \mathbb{L}, \mathcal{W}, \eta) := \frac{1}{2} \sum_{t=m}^{T} \left\| \boldsymbol{x}_t - \sum_{l \in \mathbb{L}} W^{(l)} \boldsymbol{x}_{t-l} \right\|^2 + \frac{\eta}{2} \sum_t \|\boldsymbol{x}_t\|^2 \tag{48}$$

where $\mathbf{X} \in \mathbb{R}^{T \times k}$ is the temporal factorization component in the matrix factorization $\mathbf{U} \cdot \mathbf{X}^\top$ and $\mathbf{x}_t$ denotes $\mathbf{X}$ sliced at time index t. Further we take $\mathcal{W} = \left\{ W^{(l)} \in \text{diag}(\mathbb{R}^k) \mid l \in \mathbb{L} \right\}$(i.e. set of diagonal matrices) and $\mathbb{L} = \{l_i < T \mid i = 1, \ldots, I\}$ as the set of time indices to lag.

Additionally, the regularization weight η is needed to ensure that $\mathbf{X}$ varies smoothly. However in KFT, such regularization already exists and it suffices to consider

$$\mathcal{T}_{\text{AR}}(\mathbf{X} \mid \mathbb{L}, \mathcal{W}) = \frac{1}{2} \sum_{t=m}^{T} \left\| \boldsymbol{x}_t - \sum_{l \in \mathbb{L}} W^{(l)} \boldsymbol{x}_{t-l} \right\|^2.$$

Forecasting for WLR TRMF can be extended to WLR by simply taking $\mathbf{X} = \mathbf{V}'_t \circ (\mathbf{V}_t \times_2 \mathbf{D}_t) \in \mathbb{R}^{r_{T+1} \times T \times r_{T-1}}$ and $\mathcal{W} = \left\{ W^{(l)} \in \mathbb{R}^{r_{T+1} \times r_{T-1}} \mid l \in \mathbb{L} \right\}$, which then yields

$$\mathcal{T}_{\text{AR}}(\mathbf{X} \mid \mathbb{L}, \mathcal{W}) = \frac{1}{2} \sum_{t=m}^{T} \left\| \boldsymbol{x}_t - \sum_{l \in \mathbb{L}} W^{(l)} \circ \boldsymbol{x}_{t-l} \right\|^2, \tag{49}$$

which we coin KFTRegularizer (KFTR).[3] We follow the same training strategy proposed in Yu et al. (2016) by sequentially updating $\mathcal{F} = \{\mathbf{V}_p, \mathbf{V}'_p \mid p = 1 \ldots, P\} \backslash \mathbf{X}$, $\mathbf{X}$ and $\mathcal{W}$.

Forecasting for LS KFTR can also be applied to the LS variant by applying the temporal regularization to all three components $\mathbf{X}^s = \mathbf{V}^s_t$, $\mathbf{X}^b = \mathbf{V}^b_t$ and $\mathbf{X} = \mathbf{V}_t \times \mathbf{D}_t$. We then apply Eq. (49) to each term.

3.10.2 Bayesian setting

KFTR is extended to the Bayesian setting by optimizing the quantity

$$\log p(\mathbf{X}_T, \ldots, \mathbf{X}_m) = \sum_{t=m}^{T} \log p(\mathbf{X}_t) \tag{50}$$

in addition to the ELBO. The probability distribution $p(\cdot)$ is assumed to be a univariate normal with fixed variance σ^2_{KFTR}. We define

$$\mathbf{X}_t = \left[\mathbf{V}'_{\text{time}} \circ (\mathbf{V}_{\text{time}} \times_2 \mathbf{D}_{\text{time}}) \right]_t \tag{51}$$

as functions of variational variables $\mathbf{V}, \mathbf{V}'$. Here $[\cdot]_t$ means slicing the tensor at index t. As an autoregressive dependency on $\mathbf{X}_t$ is required, we take

$$p(\mathbf{X}_t) = \int p(\mathbf{X}_t \mid \mathbf{X}_{l_1}, \ldots, \mathbf{X}_{l_K}) p(\mathbf{X}_{l_1}) \ldots p(\mathbf{X}_{l_K}) d\mathbf{X}_{l_1} \ldots \mathbf{X}_{l_K}. \tag{52}$$

However $\mathbf{X}_t$ is composed of variational variables $\mathbf{V}, \mathbf{V}'$ and thus we can write

$$\begin{aligned} \log p(\mathbf{X}_t) &= \log \int p(\mathbf{X}_t \mid \mathbf{X}_{l_1}, \ldots, \mathbf{X}_{l_K}) q(\mathbf{V}) q(\mathbf{V}') d\mathbf{V} d\mathbf{V}' \\ &= \log \mathbb{E}_q [p(\mathbf{X}_T \mid \mathbf{X}_{l_1}, \ldots, \mathbf{X}_{l_K})] \geq \mathbb{E}_q [\log p(\mathbf{X}_T \mid \mathbf{X}_{l_1}, \ldots, \mathbf{X}_{l_K})]. \end{aligned} \tag{53}$$

The log-expectation is intractable, so we use Jensen's inequality and optimize a lower bound instead. We then arrive at the following expression

[3] The recursive abbreviation reflects the recursive nature of the autoregressive regularization!

$$\begin{aligned}\mathbb{E}_q[\log p(\mathbf{X}_T \mid \mathbf{X}_{l_1}, \ldots, \mathbf{X}_{l_K})] &\propto \frac{1}{\sigma^2_{\text{TRMF}}} \mathbb{E}_q\left[\left(x_{r_p,i_t=T,r_{p+1}} - \sum_{k=1}^{K} w_k x_{r_p,i_t=l_k,r_{p+1}}\right)^2\right] \\ &= \frac{1}{\sigma^2_{\text{TRMF}}} \mathbb{E}_q\left[\left(x^2_{r_p,i_t=T,r_{p+1}} - 2x_{r_p,i_t=T,r_{p+1}} \sum_{k=1}^{K} w_k x_{r_p,i_t=l_k,r_{p+1}}\right.\right. \\ &\quad \left.\left. +(\sum_{k=1}^{K} w_k x_{r_p,i_t=l_k,r_{p+1}})^2\right)\right].\end{aligned} \tag{54}$$

We calculate the expression by taking expectations of the x-terms with respect to v, v' and arrive at

$$\begin{aligned}\mathbb{E}_q[\log p(\mathbf{X}_T \mid \mathbf{X}_{l_1}, \ldots, \mathbf{X}_{l_K})] &\propto \left[(\mathbf{M}'^2 + \Sigma') \circ \left[(\mathbf{M} \times_2 \mathbf{K})^2 + \Sigma \times_2 \mathbf{K}^2\right]\right]_T \\ &- 2\left[\mathbf{M}' \circ (\mathbf{M} \times_2 \mathbf{K})\right]_T \circ \left(\sum_{k=1}^{K} \mathbf{W}_k \circ \left[\mathbf{M}' \circ (\mathbf{M} \times_2 \mathbf{K})\right]_{l_k}\right) \\ &+ \sum_{k=1}^{K} \mathbf{W}_k^2 \circ \left[(\mathbf{M}'^2 + \Sigma') \circ \left[(\mathbf{M} \times_2 \mathbf{K})^2 + \Sigma \times_2 \mathbf{K}^2\right]\right]_{l_k} \\ &+ \left(\sum_{k=1}^{K} \mathbf{W}_k \circ \left[\mathbf{M}' \circ (\mathbf{M} \times_2 \mathbf{K})\right]_{l_k}\right)^2 - \sum_{k=1}^{K} \mathbf{W}_k^2 \circ \left[\mathbf{M}' \circ (\mathbf{M} \times_2 \mathbf{K})\right]^2_{l_k}.\end{aligned} \tag{55}$$

It should be noted that the above expression considers the WLR case for a univariate mean-field model. For a complete derivation and an expression for the multivariate meanfield model and the LS version, we refer to Appendix A.2.2.

3.11 Complexity analysis

We give a complexity analysis for all variations of KFT in the frequentist setting and Bayesian setting.

Theorem 1 *KFT has computational complexity*

$$\mathcal{O}\left(\max_p \left(n_p c_p r_p r_{p-1}\right) + \left(\max_p r_p\right)^P\right)$$

and memory footprint $\mathcal{O}\left(P \cdot \left(\max_p \left(n_p r_p r_{p-1}\right) + \max_p \left(n_p c_p\right)\right)\right)$ *for a gradient update on a batch of data. In the dual case, we take* $c_p = n_p$.

We provide proof in the appendix. It should be noted that one can reduce complexity and memory footprint by permuting the modes of the tensor such that the larger modes are on the edges, i.e. when $r_p = 1$ or $r_{p-1} = 1$. Then n_p will only scale with r_{p-1} or r_p.

KFTR complexity The additional complexity associated with adding an autoregressive regularization term is at worst $\mathcal{O}(r_p r_{p-1} KT)$, where K is the number of lags and T the size of the temporal mode. As this term scales linearly with K, it does not have an overall impact on the complexity of KFT.

4 Experiments

The experiments are divided into analyzing the frequentist version and bayesian version of KFT. Frequentist KFT is compared against competing methods on prediction and forecasting on various high dimensional datasets. Datasets we use are summarized in Table 2. For Bayesian KFT, we investigate the performance of the Bayesian version of KFT with a focus on the calibration of the obtained posterior distributions, by comparing nominal coverage probabilities to the true coverage probabilities.

4.1 Predictive performance of KFT

We compare KFT to the established FFM and LightGBM on the task of prediction on three different datasets, Retail Sales, Movielens-20M and Alcohol Sales (cf. Table 2). We compare KFT using squared loss. LightGBM is a challenging benchmark as it has continuously received development, engineering, and performance optimization since its inception in 2017. We execute our experiments by running 20 iterations of hyperopt

Table 2 Summary of datasets

Dataset	Mean	SD	N	Dimensionality
Alcohol Sales	11.567	37.678	2,816,946	$3476 \times 4542 \times 24$
Movielens-20M	3.529	1.051	19,800,443	$138493 \times 10,370 \times 24$
Fashion Retail Sales	2.430	3.968	24,291,539	$80 \times 400,000 \times 24$
CCDS	−0.000	0.997	331,500	$17 \times 125 \times 156$
Traffic	0.053	0.045	10,167,809	$10,560 \times 963$

(Bergstra et al. 2013) for all methods to find the optimal hyperparameter configuration constrained with a memory budget of 16GB (which is the memory limit of a high-end GPU) for 5 different seeds, where the seed controls how the data is split. We split our data into 60% training, 20% validation, and 20% testing. We report scores in R^2, since it provides a normalized goodness-of-fit score and measure the performance in terms of R^2-value on test data. For further details on hyperparameter range, data, and preprocessing, see Appendix F. The results are reported in Table 3, where the best results are boldfaced. We observe that KFT has configurations that can outperform the benchmarks with a good margin. Furthermore, the dual space models are generally doing better than their primal counterparts. We hypothesize that the enhanced expressiveness of kernelized side information is the reason for this.

The next experiment makes a direct comparison of KFT to recent Matrix Factorization methods tailored for the Movielens-1M and Movielens-10M datasets. The purpose of this experiment is to further demonstrate the competitiveness of KFT on recommendation tasks. Table 4 gives comparison of KFT RMSE on Movielens-1M and Movielens-10M to RMSEs of existing methods which are reported in respective paper. KFT outperforms existing non-neural models and marginally underperforms compared to neural-based state-of-the-art models for matrix factorization. In comparison to tensor factorization, KFT outperforms NTF despite NTF being a neural model. The RMSE is generally higher for tensor factorization, as data becomes sparser with each additional dimension.

Table 3 KFT results

Method/dataset	Retail sales	Movielens-20M	Alcohol sales
P-way, WLR, Primal	0.108 ± 0.435	0.38 ± 0.012	0.69 ± 0.015
P-way, WLR, Dual	0.571 ± 0.007	**0.390 ± 0.005**	**0.704 ± 0.013**
2-way, time only , WLR, Primal	0.466 ± 0.034	0.168 ± 0.013	0.472 ± 0.025
2-way, time only , WLR, Dual	**0.594 ± 0.005**	0.349 ± 0.01	0.692 ± 0.013
P-way, LS, Primal	0.152 ± 0.033	0.272 ± 0.032	0.542 ± 0.026
P-way, LS, Dual	0.027 ± 0.006	0.281 ± 0.027	0.562 ± 0.022
2-way, time only, LS, Primal	0.505 ± 0.008	0.065 ± 0.042	0.466 ± 0.022
2-way, time only,LS, Dual	0.458 ± 0.015	0.326 ± 0.024	0.699 ± 0.012
Naive methods			
P-way Vanilla, Primal	−0.374± 0.007	−2.197± 0.007	0.025 ± 0.012
P-way Vanilla, Dual	-0.087 ± 0.017	−4.837± 0.028	0.196 ± 0.008
P-way Vanilla, No side information	0.385 ± 0.002	0.147 ± 0.038	0.477 ± 0.015
Benchmark models			
LightGBM	0.581 ± 0.009	0.303 ± 0.003	0.646 ± 0.015
FFM	0.48 ± 0.032	0.293 ± 0.021	0.64 ± 0.019
Linear	0.058 ± 0.001	0.185 ± 0	0.03 ± 0.002

Performance measured in R^2

Table 4 Frequentist results in RMSE compared to reported results

Method	Movielens-1M (64/16/20%)	Movielens-1M (85.5/4.5/10%)	Movielens-10M (64/16/20%)	Movielens-10M (85.5/4.5/10%)	Neural	Claims SOTA
CF-NADE (Du et al. 2016)		**0.829**		0.771	✓	✓
NTF (Wu et al. 2019)	0.8829[a]				✓	✓
KPMF (Pal and Jenamani 2018)	0.8554					✓
STAR-GCN (Zhang et al. 2019)		0.832 ± 0.0016		**0.770 ± 0.0001**	✓	✓
HIRE (Liu et al. 2019)	0.8607					✓
KFT, P-way, WLR, Dual, movie-genome side info	**0.85 ± 0.002/0.872**[a] **± 0.003**	0.835 ± 0.002	0.796 ± 0.002	0.784 ± 0.002		

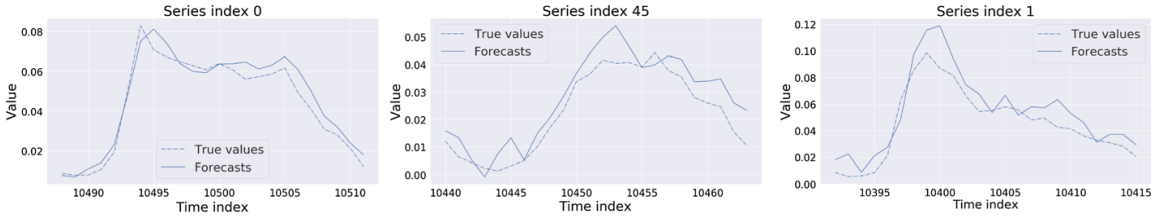
[a]Indicates a three dimensional tensor factorization for exact comparison with NTF

Table 5 Traffic results

	NRSME	ND
KFT, P-way, WLR, dual	**0.405**	0.181
TRMF (Yu et al. 2016)	0.423	0.187
DeepAR (Salinas et al. 2019)	0.420	**0.17**

Table 6 CCDS results

	RSME
KFT, P-way, WLR, dual	**0.8143**
Orthogonal (Yu et al. 2014)	0.8325



(a) Fold 5, Series index 0 **(b)** Fold 3, Series index 45 **(c)** Fold 1, Series index 1

Fig. 3 Forecasts on the Traffic dataset using KFTR.

For forecasting problems, we replicate the experiments in Yu et al. (2016) and Salinas et al. (2019) for the traffic dataset and Yu et al. (2014) for the CCDS data and compare against KFT in Tables 5 and 6.

We plot forecasts for different plots and series in Fig. 3.

4.2 Calibration study of Bayesian KFT

We use the same setup as in the frequentist case, modulo the hyperparameter evaluation objective which instead is (46). For details on hyperparameter choices and data preparation, please refer to the Appendix F. We run a regular tensor factorization without side information as a benchmark for performance, which is intended to mimic (Hawkins and Zhang 2018) as a comparison. We summarize the results in Table 7. We obtain calibrated variational approximations and observe that models using side information yield better predictive performance but that their calibration becomes slightly worse. By using a Bayesian framework we seem to generally lose some predictive performance compared to the corresponding frequentist methods, except in the case of Movielens-20M. We provide a visualization of the calibration ratios for all datasets in Fig. 4.

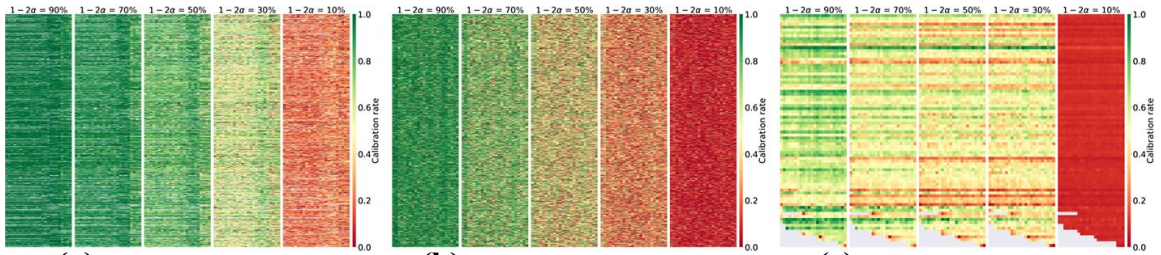
Table 7 Table of results for Bayesian KFT. Here we denote $\Xi_\alpha = |\xi_{1-2\alpha} - (1-2\alpha)|$, see Sect. 3.9.3 for more details

Model	Dataset	$\Xi_{0.05}$	$\Xi_{0.15}$	$\Xi_{0.25}$	$\Xi_{0.35}$	$\Xi_{0.45}$	Ξ	R^2	η_{criteria}	ELBO	log-likelihood
P-way, WLR Dual, Univ.	Alcohol	0.08 ±0.011	0.236 ±0.029	0.346 ±0.054	0.349 ±0.072	0.161 ±0.04	1.171 ±0.206	**0**.677 ±**0**.013	0.494 ±0.219	3219.538 ±776.927	14403.153 ±6827.637
P-way, WLR Dual, Multv.	Alcohol	0.065 ±0.045	0.217 ±0.101	0.345 ±0.171	0.411 ±0.246	0.245 ±0.18	1.282 ±0.739	0.559 ±0.024	0.724 ±0.719	517363.686 ±768293.97	628416.438 ±941682.766
P-way, LS Dual, Univ.	Alcohol	0.1 ±0.0	0.299 ±0.001	0.496 ±0.002	0.676 ±0.006	0.509 ±0.013	2.079 ±0.021	0.643 ±0.009	1.436 ±0.011	305418.812 ±111428.387	7469370.333 ±2863617.218
P-way, LS Dual, Multv.	Alcohol	0.1 ±0.0	0.299 ±0.001	0.496 ±0.003	0.678 ±0.013	0.519 ±0.033	2.093 ±0.05	0.668 ±0.004	1.425 ±0.053	1975127.667 ±1804481.827	33060096.333 ±39891533.692
P-way No Side Info	Alcohol	0.051 ±0.005	0.188 ±0.013	0.282 ±0.027	0.279 ±0.038	0.123 ±0.021	**0**.924 ±**0**.104	0.671 ±0.01	**0**.253 ±**0**.106	1334.163 ±35.865	5890.084 ±985.002
P-way, WLR Dual, Univ.	Movielens-20m	0.085 ±0.006	0.231 ±0.017	0.305 ±0.035	0.261 ±0.042	0.102 ±0.02	0.983 ±0.119	**0**.374 ±**0**.003	0.609 ±0.115	468.426 ±44.554	499.725 ±43.162
P-way, WLR Dual, Multv.	Movielens-20m	0.1 ±0.0	0.3 ±0.0	0.5 ±0.0	0.69 ±0.006	0.518 ±0.038	2.108 ±0.044	0.371 ±0.003	1.737 ±0.041	24548.967 ±333.533	29745.238 ±2509.768
P-way, LS Dual, Univ.	Movielens-20m	0.031 ±0.013	0.063 ±0.055	0.075 ±0.055	0.058 ±0.04	0.022 ±0.014	**0**.248 ±**0**.174	0.316 ±0.029	−0.068 ±**0**.196	355.079 ±98.181	361.622 ±100.985
P-way, LS Dual, Multv.	Movielens-20m	0.097 ±0.005	0.277 ±0.035	0.42 ±0.1	0.466 ±0.184	0.244 ±0.134	1.503 ±0.454	0.326 ±0.012	1.177 ±0.454	−56059.229 ±7918.906	−55766.211 ±7650.988

Table 7 (continued)

Model	Dataset	$\Xi_{0.05}$	$\Xi_{0.15}$	$\Xi_{0.25}$	$\Xi_{0.35}$	$\Xi_{0.45}$	Ξ	R^2	η_{criteria}	ELBO	log-likelihood
P-way No Side Info	Movielens-20m	0.089 ±0.005	0.234 ±0.012	0.301 ±0.018	0.245 ±0.015	0.092 ±0.005	0.961 ±0.053	0.285 ±0.004	0.676 ±0.053	99.511 ±21.923	128.062 ±20.528
2-way, WLR Dual, Univ.	Retail	0.006 ±0.006	0.053 ±0.009	0.083 ±0.01	0.072 ±0.008	0.029 ±0.003	0.242 ±0.026	**0**.547 ±**0**.004	−0.304 ±0.029	1007.522 ±33.546	1072.142 ±33.192
2-way, WLR Dual, Multv.	Retail	0.087 ±0.014	0.064 ±0.018	0.037 ±0.017	0.017 ±0.013	0.004 ±0.004	**0**.209 ±**0**.067	0.546 ±0.004	−0.337 ±**0**.065	−68984229.333 ±635903.157	−68984202.667 ±635883.04
2-way, LS Dual, Univ.	Retail	0.086 ±0.004	0.247 ±0.012	0.366 ±0.022	0.386 ±0.031	0.188 ±0.02	1.273 ±0.088	0.4 ±0.014	0.873 ±0.078	372.6 ±118.855	469.483 ±121.855
2-way, LS Dual, Multv.	Retail	0.09 ±0.008	0.259 ±0.026	0.391 ±0.05	0.431 ±0.073	0.225 ±0.047	1.396 ±0.203	0.413 ±0.01	0.983 ±0.194	434415541.333 ±37518140.035	434420117.333 ±37518610.316
2-way No Side Info	Retail	0.033 ±0.016	0.037 ±0.021	0.076 ±0.021	0.075 ±0.016	0.032 ±0.006	0.252 ±0.048	0.536 ±0.003	−0.284 ±0.05	422.562 ±66.308	451.334 ±63.625

We ideally want Ξ_α and Ξ to be as close to zero as possible coupled with a high R^2



(a) Alcohol Sales **(b)** Movielens-20m **(c)** Retail Sales

Fig. 4 Heatmap of $\xi_{1-2\alpha}$ for all datasets. Here the y-axis is location/userID and x-axis is time, where targets have been aggregated on items. We see that the calibration rate over all aggregates consistently adjusts with changes in $1 - 2\alpha$

We compare KFTR to existing Bayesian models with reported results in Table 8, where the model is optimized for η_{criteria}, in contrast to RMSE used in Kim and Choi (2014). KFT performs well in a Bayesian setting compared to Kim and Choi (2014) and also yield calibrated estimates. For forecast problems, we apply KFTR to the Traffic and CCDS dataset and report the results in Table 9. We find that the WLR version of KFT does well and yields calibrated forecasts.

We plot forecasts for the Traffic dataset with uncertainty quantification in Fig. 5.

5 Analysis

We demonstrated the practical utility of KFT in both a frequentist and Bayesian context. We now scrutinize the robustness and effectiveness of KFT as a remedy for constraint-imposing side information.

5.1 Does KFT really amend the constraints of directly apply side information?

To validate this, we train KFT-WLR and a naive model on the Alcohol Sales dataset using kernelized side information for 5 hyperparameter searches. We plot the mean training, validation, and test error of the 5 searches (and the standard errors) against epochs in Fig. 6.

5.2 How does KFT perform when applying constant side information?

To answer this question, we replace all side information with a constant $\mathbf{1}$ and kernelize it. The results in the first row of Table 10 indicate that KFT indeed is robust towards constant side information, as the performance does not degrade dramatically.

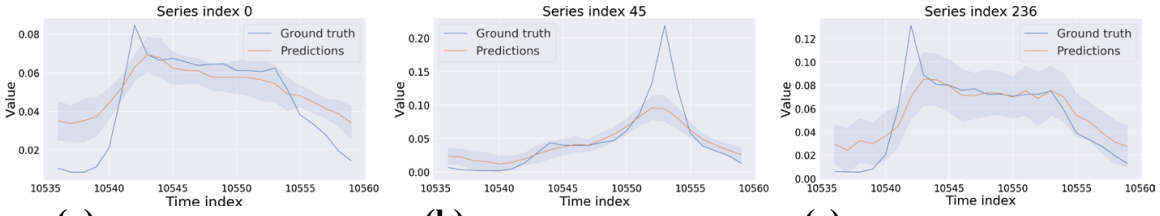
Table 8 Results with RMSE and (Ξ)

Method	Movielens-1M (64/16/20%)	Movielens-1M (85.5/4.5/10%)	Movielens-10M (64/16/20%)	Movielens-10M (85.5/4.5/10%)
VBMF (Kim and Choi 2014)				0.8444[a]
KFT, Univariate, WLR, Dual	0.92 ± 0.008 (0.48 ± 0.023)	0.891 ± 0.006 (0.555 ± 0.046)	0.848 ± 0.012 (0.425 ± 0.018)	0.834 ± 0.007 (0.45 ± 0.021)
KFT, Multivariate, WLR, Dual	0.893 ± 0.002 (0.231 ± 0.02)	0.876 ± 0.013 (0.356 ± 0.181)	0.877 ± 0.009 (0.304 ± 0.084)	0.864 ± 0.004 (0.201 ± 0.092)

[a] As no explicit train/valid/test ratio were provided, the comparison might not be objective

Table 9 Table of results for Bayesian forecast experiments

Model	Dataset	$\Xi_{0.05}$	$\Xi_{0.15}$	$\Xi_{0.25}$	$\Xi_{0.35}$	$\Xi_{0.45}$	Ξ	R^2	η_{criteria}	ELBO	log-likelihood
P-way, WLR Dual, Univ.	Traffic	0.035 ±0.013	0.133 ±0.055	0.147 ±0.078	0.097 ±0.059	0.032 ±0.019	0.444 ±0.221	0.671 ±0.11	−0.227 ±0.28	629.04 ±6.378	629.861 ±6.176
P-way, LS Dual, Univ.	Traffic	0.1 ±0.0	0.3 ±0.0	0.499 ±0.001	0.68 ±0.01	0.489 ±0.046	2.068 ±0.057	0.213 ±0.081	1.855 ±0.096	52845.821 ±47567.564	1031271.526 ±904233.765
P-way, WLR Dual, Univ.	CCDS	0.042 ±0.021	0.109 ±0.05	0.125 ±0.061	0.094 ±0.05	0.036 ±0.021	0.406 ±0.203	0.196 ±0.011	0.21 ±0.196	20591.652 ±6209.691	20614.386 ±6229.005
P-way, LS Dual, Univ.	CCDS	0.1 ±0.0	0.3 ±0.0	0.498 ±0.003	0.649 ±0.079	0.431 ±0.181	1.978 ±0.263	0.103 ±0.005	1.875 ±0.263	14691.796 ±3708.965	66322.241 ±46105.914



(a) Fold 3, Series index 0 **(b)** Fold 3, Series index 45 **(c)** Fold 3, Series index 236

Fig. 5 Forecasts on the Traffic dataset using Bayesian KFTR.

5.3 How does KFT perform when applying noise as side information?

Similar to the previous question, we now replace the side information with standard Gaussian noise instead. The results in the last row of Table 10 indicate that KFT also is robust against noise and surprisingly performant as well. A possible explanation for this is that adding Gaussian noise serves as an implicit regularizer or that the original side information is similarly distributed as standard Gaussian noise. We conclude that KFT is stable against uninformative side information in the form of Gaussian noise.

6 Conclusion

We identified an inherent limitation of side information based tensor regression and gave a method that removes this limitation. Our proposed KFT method yields competitive performance against state-of-the-art large-scale prediction models on a fixed computational budget. Specifically, as the experiments in Table 3 demonstrate, for at least some cases of real practical interest, Weighted Latent Regression is the most performant configuration. Further, KFT offers extended versatility in terms of calibrated Bayesian variational estimates. Our analysis shows that KFT solves the problems we described in Sect. 2 and is robust for adversarial side information in the form of Gaussian noise. A direction for further development would be to characterize identifiability conditions for KFT and extend the Bayesian framework beyond mean-field variational inference.

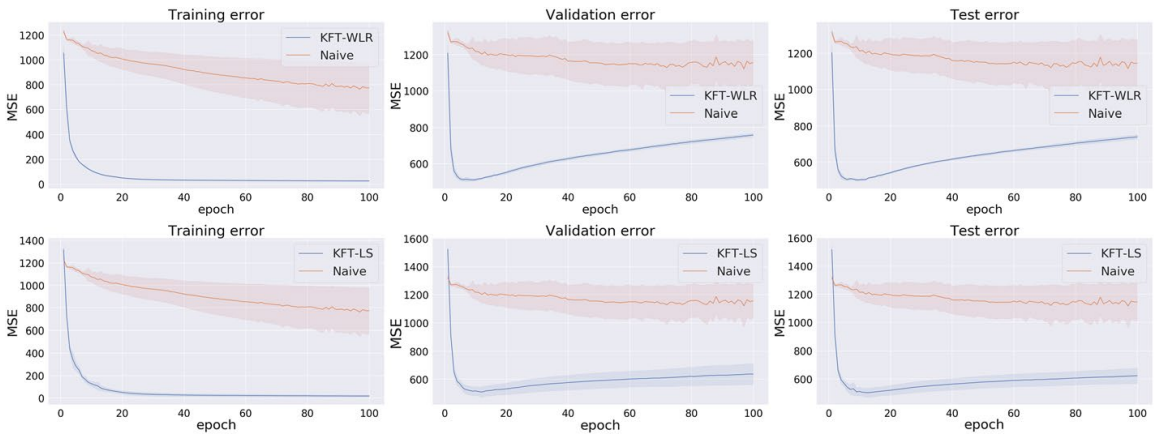


Fig. 6 Training, validation and test error against epoch for KFT-WLR/KFT-LS and a naive model (with "Dual" side information) on the Alcohol Sales dataset

Table 10 Results from additional experiments for P-way dual models

Method/dataset	Retail sales	Movielens-20M	Alcohol sales
WLR, Constant side information, Dual	0.56 ±0.013	0.33 ± 0.039	0.605 ± 0.041
WLR, Noise side information, Dual	0.553 ± 0.006	0.388 ± 0.003	0.704 ± 0.01

Appendix A: $\mathbb{E}_q[\log p(y_{i_1,\ldots,i_P}|\mathbf{V}_1,\ldots,\mathbf{V}_P,\mathbf{V}'_1,\ldots,\mathbf{V}'_P)]$ derivations

A.1 Weighted latent regression

Assuming that σ_y is a constant hyperparameter. We first have that

$$
\begin{aligned}
&\mathbb{E}_q[\log p(y_{i_1,\ldots,i_P}|\mathbf{V}_1,\ldots,\mathbf{V}_P,\mathbf{V}'_1,\ldots,\mathbf{V}'_P)] \\
&\propto \frac{1}{\sigma_y^2}\mathbb{E}_q\left[(y_{i_1,\ldots,i_P} - \sum_{\substack{r_1,\ldots,r_P \\ i'_1,\ldots,i'_P \\ i''_1,\ldots,i''_P}} \prod_{p=1}^{P} v'_{r_p i''_p r_{p-1}} v_{r_p i'_p r_{p-1}} \underbrace{k(\mathbf{x}^p_{i_p},\mathbf{x}^p_{i'_p})\delta_{\mathbf{x}^p_{i_p}}(\mathbf{x}^p_{i''_p})}_{f(k,\delta)})^2\right] \\
&= \frac{1}{\sigma_y^2}\left[y^2_{i_1,\ldots,i_P} - 2y_{i_1,\ldots,i_P}\cdot \sum_{\substack{r_1,\ldots,r_P \\ i'_1,\ldots,i'_P \\ i''_1,\ldots,i''_P}} \prod_{p=1}^{P} \mathbb{E}_q[v'_{r_p i''_p r_{p-1}}]\mathbb{E}_q[v_{r_p i'_p r_{p-1}}] f(k,\delta)\right. \\
&\quad + \sum_{\substack{r_1,\ldots,r_P \\ i'_1,\ldots,i'_P \\ i''_1,\ldots,i''_P \\ q_1,\ldots,q_P \\ j'_1,\ldots,j'_P \\ j''_1,\ldots,j''_P}} \prod_{p=1}^{P} \mathbb{E}_q[v'_{q_p j''_p q_{p-1}} v'_{r_p i''_p r_{p-1}}]\mathbb{E}_q[v_{r_p i'_p r_{p-1}} v_{q_p j'_p q_{p-1}}] k(\mathbf{x}^p_{i_p},\mathbf{x}^p_{i''_p}) \\
&\quad \left.\delta_{\mathbf{x}^p_{i_p}}(\mathbf{x}^p_{i''_p})k(\mathbf{x}^p_{i_p},\mathbf{x}^p_{j'_p})\delta_{\mathbf{x}^p_{i_p}}(\mathbf{x}^p_{j''_p})\right]
\end{aligned}
\tag{56}
$$

Our goal is to find the corresponding tensor operations of the sum terms.

A.1.1 Univariate meanfield

We first have that

$$
\mathbb{E}_q[v_{r_p i'_p r_{p-1}} v_{q_p j'_p q_{p-1}}] = \mu_{r_p i'_p r_{p-1}}\mu_{q_p j'_p q_{p-1}} + \delta_{r_p}(q_p)\delta_{r_{p-1}}(q_{p-1})\delta_{i'_p}(j'_p)\sigma^2_{r_p i'_p r_{p-1}} \tag{57}
$$

Then the last term is can be written as

$$
\begin{aligned}
\implies & \sum_{\substack{r_1,\ldots,r_P \\ i'_1,\ldots,i'_P \\ q_1,\ldots,q_P \\ j'_1,\ldots,j'_P}} \prod_{p=1}^{P} \mathbb{E}_q[v'_{q_p i_p q_{p-1}} v'_{r_p i_p r_{p-1}}]\mathbb{E}_q[v_{r_p i'_p r_{p-1}} v_{q_p j'_p q_{p-1}}]k(\mathbf{x}^p_{i_p}, \mathbf{x}^p_{i'_p})k(\mathbf{x}^p_{i_p}, \mathbf{x}^p_{j'_p}) \\
&= \sum \prod_{p=1}^{P} (\mu'_{q_p i_p q_{p-1}} \mu'_{r_p i_p r_{p-1}} + \delta_{r_p}(q_p)\delta_{r_{p-1}}(q_{p-1})\sigma'^2_{r_p i_p r_{p-1}}) \\
&\quad (\mu_{q_p j'_p q_{p-1}} \mu_{r_p i'_p r_{p-1}} + \delta_{r_p}(q_p)\delta_{r_{p-1}}(q_{p-1})\delta_{i'_p}(j'_p)\sigma^2_{r_p i'_p r_{p-1}})k(\mathbf{x}^p_{i_p}, \mathbf{x}^p_{i'_p})k(\mathbf{x}^p_{i_p}, \mathbf{x}^p_{j'_p}) \\
&= \sum \prod_{p=1}^{P} \mu'_{q_p i_p q_{p-1}} \mu'_{r_p i_p r_{p-1}} \mu_{q_p j'_p q_{p-1}} \mu_{r_p i'_p r_{p-1}} k(\mathbf{x}^p_{i_p}, \mathbf{x}^p_{i'_p})k(\mathbf{x}^p_{i_p}, \mathbf{x}^p_{j'_p}) \\
&\quad + \sum_{\substack{r_1,\ldots,r_P \\ i'_1,\ldots,i'_P}} \prod_{p=1}^{P} \sigma'^2_{r_p i_p r_{p-1}} \sigma^2_{r_p i'_p r_{p-1}} k(\mathbf{x}^p_{i_p}, \mathbf{x}^p_{i'_p})^2 \\
&\quad + \sum_{\substack{r_1,\ldots,r_P \\ i'_1,\ldots,i'_P \\ j'_1,\ldots,j'_P}} \prod \sigma'^2_{r_p i_p r_{p-1}} \mu_{r_p i'_p r_{p-1}} \mu_{r_p j'_p r_{p-1}} k(\mathbf{x}^p_{i_p}, \mathbf{x}^p_{i'_p})k(\mathbf{x}^p_{i_p}, \mathbf{x}^p_{j'_p}) \\
&\quad + \sum_{\substack{r_1,\ldots,r_P \\ i'_1,\ldots,i'_P}} \prod \sigma^2_{r_p i'_p r_{p-1}} \mu'^2_{r_p i_p r_{p-1}} k(\mathbf{x}^p_{i_p}, \mathbf{x}^p_{i'_p})^2
\end{aligned}
\tag{58}
$$

$$
\begin{aligned}
\iff & \left(\prod_p^P \times_{-1} \mathbf{M}'_p \circ (\mathbf{M}_p \times_2 \mathbf{K}_p) \right)^2 + \left(\prod_p^P \times_{-1} \Sigma'_p \circ (\Sigma_p \times_2 (\mathbf{K}_p)^2) \right) \\
& + \left(\prod_p^P \times_{-1} (\Sigma'_p \circ (\mathbf{M}_p \times_2 \mathbf{K}_p)^2) \right) + \left(\prod_p^P \times_{-1} \mathbf{M}'^2_p \circ (\Sigma_p \times_2 (\mathbf{K}_p)^2) \right)
\end{aligned}
\tag{59}
$$

where $\mathbf{M}'_p, \mathbf{M}$ and Σ', Σ correspond to the tensors containing the variational parameters $\mu'_{r_p i_p r_{p-1}}$, $\mu_{r_p i_p r_{p-1}}$ and $\sigma'^2_{r_p i_p r_{p-1}}$, $\sigma^2_{r_p i_p r_{p-1}}$ respectively. For the case of RFF's in dual space, we approximate $\Sigma_p \times_2 (\mathbf{K}_p)^2 \approx \Sigma_p \times_2 (\boldsymbol{\Phi}_p \bullet \boldsymbol{\Phi}_p)^\top \times_2 (\boldsymbol{\Phi}_p \bullet \boldsymbol{\Phi}_p)$, where $\bullet$ is the transposed Khatri-Rao product. It should further be noted that any square term, means elementwise squaring. With the middle term given by

$$
2\mathbf{Y} \circ \left(\prod_{p=1}^{P} \times_{-1} (\mathbf{M}'_p \circ (\mathbf{M}_p \times_2 \mathbf{K}_p)) \right) \tag{60}
$$

the full reconstruction term is expressed as:

$$
\begin{aligned}
&\mathbb{E}_q[\log p(y_{i_1,\ldots,i_P}|\mathbf{V}_1,\ldots,\mathbf{V}_p,\mathbf{V}'_1,\ldots,\mathbf{V}'_p)] \\
&\propto \frac{1}{\sigma_y^2}\Bigg[\mathbf{Y}^2 - 2\mathbf{Y}\circ\Bigg(\prod_{p=1}^{P}\times_{-1}(\mathbf{M}'_p\circ(\mathbf{M}_p\times_2\mathbf{K}_p))\Bigg) \\
&\quad + \Bigg(\prod_{p}^{P}\times_{-1}\mathbf{M}'_p\circ(\mathbf{M}_p\times_2\mathbf{K}_p)\Bigg)^2 + \Bigg(\prod_{p}^{P}\times_{-1}\Sigma'_p\circ(\Sigma_p\times_2(\mathbf{K}_p)^2)\Bigg) \\
&\quad + \Bigg(\prod_{p}^{P}\times_{-1}(\Sigma'_p\circ((\mathbf{M}_p\times_2\mathbf{K}_p)^2)\Bigg) + \Bigg(\prod_{p}^{P}\times_{-1}\mathbf{M}'^2_p\circ(\Sigma_p\times_2(\mathbf{K}_p)^2)\Bigg)\Bigg]
\end{aligned}
\tag{61}
$$

A.1.2 Multivariate meanfield

Similarly first observe that

$$
\mathbb{E}_q[v_{r_p i'_p r_{p-1}} v_{q_p j'_p q_{p-1}}] = \mu_{r_p i'_p r_{p-1}}\mu_{q_p j'_p q_{p-1}} + \delta_{r_p}(q_p)\delta_{r_{p-1}}(q_{p-1})\sigma_{i'_p}\sigma_{j'_p}. \tag{62}
$$

It then follows that the third term is calculated as

$$
\begin{aligned}
&\sum\prod_{p=1}^{P}(\mu'_{q_p i_p q_{p-1}}\mu'_{r_p i_p r_{p-1}} + \delta_{r_p}(q_p)\delta_{r_{p-1}}(q_{p-1})\sigma'^2_{r_p i_p r_{p-1}}) \\
&\quad (\mu_{q_p j'_p q_{p-1}}\mu_{r_p i'_p r_{p-1}} + \delta_{r_p}(q_p)\delta_{r_{p-1}}(q_{p-1})\sigma_{i'_p}\sigma_{j'_p})k(\mathbf{x}^p_{i_p},\mathbf{x}^p_{i'_p})k(\mathbf{x}^p_{i_p},\mathbf{x}^p_{j'_p}) \\
&= \sum\prod_{p=1}^{P}\mu'_{q_p i_p q_{p-1}}\mu'_{r_p i_p r_{p-1}}\mu_{q_p j'_p q_{p-1}}\mu_{r_p i'_p r_{p-1}}k(\mathbf{x}^p_{i_p},\mathbf{x}^p_{i'_p})k(\mathbf{x}^p_{i_p},\mathbf{x}^p_{j'_p}) \\
&\quad + \sum_{\substack{r_1,\ldots,r_P\\ i'_1,\ldots,i'_P\\ j'_1,\ldots,j'_P}}\prod_{p=1}^{P}\sigma'^2_{r_p i_p r_{p-1}}\sigma_{i'_p}\sigma_{j'_p}k(\mathbf{x}^p_{i_p},\mathbf{x}^p_{i'_p})k(\mathbf{x}^p_{i_p},\mathbf{x}^p_{j'_p}) \\
&\quad + \sum_{\substack{r_1,\ldots,r_P\\ i'_1,\ldots,i'_P\\ j'_1,\ldots,j'_P}}\prod_{p=1}^{P}\sigma'^2_{r_p i_p r_{p-1}}\mu_{r_p i'_p r_{p-1}}\mu_{r_p j'_p r_{p-1}}k(\mathbf{x}^p_{i_p},\mathbf{x}^p_{i'_p})k(\mathbf{x}^p_{i_p},\mathbf{x}^p_{j'_p}) \\
&\quad + \sum_{\substack{r_1,\ldots,r_P\\ i'_1,\ldots,i'_P\\ j'_1,\ldots,j'_P}}\prod_{p=1}^{P}\sigma_{i'_p}\sigma_{j'_p}\mu'^2_{r_p i_p r_{p-1}}k(\mathbf{x}^p_{i_p},\mathbf{x}^p_{i'_p})k(\mathbf{x}^p_{i_p},\mathbf{x}^p_{j'_p})
\end{aligned}
\tag{63}
$$

$$\begin{aligned}\iff &\left(\prod_{p}^{P}\times_{-1}\mathbf{M}'_p\circ(\mathbf{M}_p\times_2\mathbf{K}_p)\right)^2\left(\prod_{p}^{P}\times_{-1}\Sigma'_p\circ(\mathbf{1}\times_2\left(\text{diag}((\mathbf{K}_p\cdot\mathbf{B}_p)^2\cdot\bar{1})\right)\right)\\ &+\left(\prod_{p}^{P}\times_{-1}(\Sigma'_p\circ(\mathbf{M}_p\times_2\mathbf{K}_p)^2)\right)+\left(\prod_{p}^{P}\times_{-1}\mathbf{M}'^2_p\circ(\mathbf{1}\times_2\left(\text{diag}((\mathbf{K}_p\cdot\mathbf{B}_p)^2\cdot\bar{1})\right)\right)\end{aligned} \tag{64}$$

where $\Sigma_p = \mathbf{B}_p\mathbf{B}_p^T$, $\mathbf{1}$ denotes a constant one tensor with the same dimensions as Σ'_p, $\bar{1}\in\mathbb{R}^{R\times 1}$ where R is the column dimension of $\mathbf{B}_p$ and Σ'_p is the same as in the univariate case. For RFF's we have that

$$\begin{aligned}(\mathbf{K}_p\cdot\mathbf{B}_p)^2\cdot\bar{1}\approx&((\boldsymbol{\Phi}_p\cdot\boldsymbol{\Phi}_p^\top)\cdot\mathbf{B}_p)^2\cdot\bar{1}+\text{vec}(D_p^2)\\ =&(\boldsymbol{\Phi}_p\cdot(\boldsymbol{\Phi}_p^\top\cdot\mathbf{B}_p))^2\cdot\bar{1}+\text{vec}(D_p^2).\end{aligned} \tag{65}$$

As the middle term remains the same as in the univariate case our full reconstruction term is

$$\begin{aligned}&\mathbb{E}_q[\log p(y_{i_1,\ldots,i_P}|\mathbf{V}_1,\ldots,\mathbf{V}_p,\mathbf{V}'_1,\ldots,\mathbf{V}'_p)]\\ &\propto\frac{1}{\sigma_y^2}\Bigg[\mathbf{Y}^2-2\mathbf{Y}\circ\left(\prod_{p=1}^{P}\times_{-1}(\mathbf{M}'_p\circ(\mathbf{M}_p\times_2\mathbf{K}_p))\right)\\ &+\left(\prod_{p}^{P}\times_{-1}\mathbf{M}'_p\circ(\mathbf{M}_p\times_2\mathbf{K}_p)\right)^2+\left(\prod_{p}^{P}\times_{-1}\Sigma'_p\circ(\mathbf{1}\times_2\left(\text{diag}((\mathbf{K}_p\cdot\mathbf{B}_p)^2\cdot\bar{1})\right)\right)\\ &+\left(\prod_{p}^{P}\times_{-1}(\Sigma'_p\circ(\mathbf{M}_p\times_2\mathbf{K}_p)^2)\right)+\left(\prod_{p}^{P}\times_{-1}\mathbf{M}'^2_p\circ(\mathbf{1}\times_2\left(\text{diag}((\mathbf{K}_p\cdot\mathbf{B}_p)^2\cdot\bar{1})\right)\right)\Bigg]\end{aligned} \tag{66}$$

A.2 Latent scaling

Assuming that σ_y is a constant hyperparameter. We first have that

$$
\begin{aligned}
&\mathbb{E}_q[\log p(y_{i_1,\ldots,i_P}|\mathbf{V}_1,\ldots,\mathbf{V}_P,\mathbf{V}'_1,\ldots,\mathbf{V}'_P)] \\
&\quad\propto \frac{1}{\sigma_y^2}\mathbb{E}_q\Bigg[(y_{i_1,\ldots,i_P} - \Bigg(\sum_{s_1,\ldots,s_P}\prod_{p=1}^{P} v^s_{s_p i_p s_{p-1}} \sum_{\substack{r_1,\ldots,r_P \\ i'_1,\ldots,i'_P}} \prod_{p=1}^{P} v_{r_p i'_p r_{p-1}} k(\mathbf{x}^p_{i_p},\mathbf{x}^p_{i'_p}) \\
&\qquad + \sum_{b_1,\ldots,b_P}\prod_{p=1}^{P} v^b_{b_p i_p b_{p-1}}\Bigg)\Bigg)^2\Bigg] \\
&\quad= \frac{1}{\sigma_y^2}\Bigg[y^2_{i_1,\ldots,i_P} - 2y_{i_1,\ldots,i_P} \\
&\qquad\cdot\Bigg(\underbrace{\sum_{s_1,\ldots,s_P}\prod_{p=1}^{P}\mathbb{E}_q[v^s_{s_p i_p s_{p-1}}]}_{f_{v^s_{s_p i_p s_{p-1}}}} \underbrace{\sum_{\substack{r_1,\ldots,r_P \\ i'_1,\ldots,i'_P}}\prod_{p=1}^{P}\mathbb{E}_q[v_{r_p i'_p r_{p-1}}]k(\mathbf{x}^p_{i_p},\mathbf{x}^p_{i'_p})}_{f_{v_{r_p i'_p r_{p-1}}}} \\
&\qquad + \underbrace{\sum_{b_1,\ldots,b_P}\prod_{p=1}^{P}\mathbb{E}_q[v^b_{b_p i_p b_{p-1}}]}_{f_{v^b_{b_p i_p b_{p-1}}}}\Bigg) \\
&\qquad + \sum_{\substack{s_1,\ldots,s_P \\ s'_1,\ldots,s'_P}}\prod_{p=1}^{P}\mathbb{E}_q[v^s_{s_p i_p s_{p-1}} v^s_{s'_p i_p s'_{p-1}}] \sum_{\substack{r_1,\ldots,r_P \\ q_1,\ldots,q_P \\ i'_1,\ldots,i'_P \\ j'_1,\ldots,j'_P}}\prod_{p=1}^{P}\mathbb{E}_q[v_{r_p i'_p r_{p-1}} v_{q_p j'_p q_{p-1}}]k(\mathbf{x}^p_{i_p},\mathbf{x}^p_{i'_p})k(\mathbf{x}^p_{i_p},\mathbf{x}^p_{j'_p}) \\
&\qquad + 2f_{v^s_{s_p i_p s_{p-1}}} f_{v_{r_p i'_p r_{p-1}}} f_{v^b_{b_p i_p b_{p-1}}} + \sum_{\substack{b_1,\ldots,b_P \\ b'_1,\ldots,b'_P}}\prod_{p=1}^{P}\mathbb{E}_q[v^b_{b_p i_p b_{p-1}} v^b_{b'_p i_p b'_{p-1}}]\Bigg].
\end{aligned}
\tag{67}
$$

A.2.1 Univariate case

For the univariate case, the squared component of the reconstruction term in the ELBO becomes

$$\left(\left(\prod_{p=1}^{P}\times_{-1}\mathbf{M}_p^s\right)^2+\left(\prod_{p=1}^{P}\times_{-1}\Sigma_p^s\right)\right)\circ\left(\left(\prod_{p}^{P}\times_{-1}\mathbf{M}_p\times_2\mathbf{K}_p\right)^2+\prod_{p=1}^{P}\times_{-1}\Sigma_p\times_2(\mathbf{K}_p)^2\right)$$
$$+\,2\left(\prod_{p=1}^{P}\times_{-1}\mathbf{M}_p^s\right)\circ\left(\prod_{p=1}^{P}\times_{-1}\mathbf{M}_p^b\right)\circ\left(\prod_{p=1}^{P}\times_{-1}\mathbf{M}_p\times_2\mathbf{K}_p\right)$$
$$+\left(\left(\prod_{p=1}^{P}\times_{-1}\mathbf{M}_p^b\right)^2+\left(\prod_{p=1}^{P}\times_{-1}\Sigma_p^b\right)\right). \tag{68}$$

Too see this, we notice that all of the sums from the weighted latent regression case reappears here as well. The entire expression is given by

$$\begin{aligned}
&\mathbb{E}_q[\log p(y_{i_1,\ldots,i_P}|\mathbf{V}_1,\ldots,\mathbf{V}_P,\mathbf{V}'_1,\ldots,\mathbf{V}'_P)]\\
&\propto\frac{1}{\sigma_y^2}\Bigg[\mathbf{Y}^2-2\mathbf{Y}\circ\left(\prod_{p=1}^{P}(\mathbf{M}_p^s\circ(\mathbf{M}_p\times_2\mathbf{K}_p)+\mathbf{M}_p^b)\right)\\
&\quad+\left(\left(\prod_{p=1}^{P}\times_{-1}\mathbf{M}_p^s\right)^2+\left(\prod_{p=1}^{P}\times_{-1}\Sigma_p^s\right)\right)\\
&\quad\circ\left(\left(\prod_{p}^{P}\times_{-1}\mathbf{M}_p\times_2\mathbf{K}_p\right)^2+\prod_{p=1}^{P}\times_{-1}\Sigma_p\times_2(\mathbf{K}_p)^2\right)\\
&\quad+2\left(\prod_{p=1}^{P}\times_{-1}\mathbf{M}_p^s\right)\circ\left(\prod_{p=1}^{P}\times_{-1}\mathbf{M}_p^b\right)\\
&\quad\circ\left(\prod_{p=1}^{P}\times_{-1}\mathbf{M}_p\times_2\mathbf{K}_p\right)+\left(\left(\prod_{p=1}^{P}\times_{-1}\mathbf{M}_p^b\right)^2+\left(\prod_{p=1}^{P}\times_{-1}\Sigma_p^b\right)\right)\Bigg].
\end{aligned} \tag{69}$$

A.2.2 Multivariate case

For multivariate case the last term becomes

$$
\begin{aligned}
&\left(\left(\prod_{p=1}^{P}\times_{-1}\mathbf{M}_p^s\right)^2+\left(\prod_{p=1}^{P}\times_{-1}\Sigma_p^s\right)\right)\\
&\quad\circ\left(\left(\prod_{p}^{P}\times_{-1}\mathbf{M}_p\times_2\mathbf{K}_p\right)^2+\prod_{p=1}^{P}\times_{-1}(\mathbf{1}\times_2\left(\mathrm{diag}((\mathbf{K}_p\cdot\mathbf{B}_p)^2\cdot\bar{1})\right))\right)\\
&\quad+2\left(\prod_{p=1}^{P}\times_{-1}\mathbf{M}_p^s\right)\circ\left(\prod_{p=1}^{P}\times_{-1}\mathbf{M}_p^b\right)\\
&\quad\circ\left(\prod_{p=1}^{P}\times_{-1}\mathbf{M}_p\times_2\mathbf{K}_p\right)+\left(\left(\prod_{p=1}^{P}\times_{-1}\mathbf{M}_p^b\right)^2+\left(\prod_{p=1}^{P}\times_{-1}\Sigma_p^b\right)\right)
\end{aligned}
\tag{70}
$$

Again, we notice that all of the sums from the weighted latent regression case reappears here as well. The full expression is given by

$$
\begin{aligned}
&\mathbb{E}_q[\log p(y_{i_1,\ldots,i_P}|\mathbf{V}_1,\ldots,\mathbf{V}_p,\mathbf{V}'_1,\ldots,\mathbf{V}'_p)]\\
&\quad\propto\frac{1}{\sigma_y^2}\Bigg[\mathbf{Y}^2-2\mathbf{Y}\circ\left(\prod_{p=1}^{P}(\mathbf{M}_p^s\circ(\mathbf{M}_p\times_2\mathbf{K}_p)+\mathbf{M}_p^b)\right)\\
&\quad+\left(\left(\prod_{p=1}^{P}\times_{-1}\mathbf{M}_p^s\right)^2+\left(\prod_{p=1}^{P}\times_{-1}\Sigma_p^s\right)\right)\\
&\quad\circ\left(\left(\prod_{p}^{P}\times_{-1}\mathbf{M}_p\times_2\mathbf{K}_p\right)^2+\prod_{p=1}^{P}\times_{-1}(\mathbf{1}\times_2\left(\mathrm{diag}((\mathbf{K}_p\cdot\mathbf{B}_p)^2\cdot\bar{1})\right))\right)\\
&\quad+2\left(\prod_{p=1}^{P}\times_{-1}\mathbf{M}_p^s\right)\circ\left(\prod_{p=1}^{P}\times_{-1}\mathbf{M}_p^b\right)\circ\left(\prod_{p=1}^{P}\times_{-1}\mathbf{M}_p\times_2\mathbf{K}_p\right)\\
&\quad+\left(\left(\prod_{p=1}^{P}\times_{-1}\mathbf{M}_p^b\right)^2+\left(\prod_{p=1}^{P}\times_{-1}\Sigma_p^b\right)\right)\Bigg]
\end{aligned}
\tag{71}
$$

A.3 Forecasting

We derive an expression for the forecasting term for the WLR case

$$
\begin{aligned}
&\mathbb{E}_q[\log p(\mathbf{X}_T \mid \mathbf{X}_{l_1}, \ldots, \mathbf{X}_{l_K})] \\
&\quad \propto \frac{1}{\sigma^2_{\text{TRMF}}} \mathbb{E}_q \left[\left(x_{r_p, i_t = T, r_{p+1}} - \sum_{k=1}^{K} w_k x_{r_p, i_t = l_k, r_{p+1}} \right)^2 \right] \\
&\quad = \frac{1}{\sigma^2_{\text{TRMF}}} \mathbb{E}_q \left[\left(x^2_{r_p, i_t = T, r_{p+1}} - 2 x_{r_p, i_t = T, r_{p+1}} \sum_{k=1}^{K} w_k x_{r_p, i_t = l_k, r_{p+1}} \right.\right. \\
&\qquad \left.\left. + (\sum_{k=1}^{K} w_k x_{r_p, i_t = l_k, r_{p+1}})^2 \right) \right].
\end{aligned} \tag{72}
$$

We first establish that

$$
x_{r_p, i_t = \tau, r_{p+1}} = \sum_{\substack{t' = 1 \ldots T \\ t'' = \tau}} v'_{r_p t'' r_{p-1}} v_{r_p t' r_{p-1}} k(\mathbf{x}_t, \mathbf{x}_{t'}) \delta_{\mathbf{x}_t}(\mathbf{x}_{t''}). \tag{73}
$$

Consequently,

$$
\begin{aligned}
\mathbb{E}_q[x_{r_p, i_t = \tau, r_{p+1}}] &= \sum_{\substack{t' = 1 \ldots T \\ t'' = \tau}} \mathbb{E}_q[v'_{r_p t'' r_{p-1}}] \mathbb{E}_q[v_{r_p t' r_{p-1}}] k(\mathbf{x}_t, \mathbf{x}_{t'}) \delta_{\mathbf{x}_t}(\mathbf{x}_{t''}) \\
&= \sum_{\substack{t' = 1 \ldots T \\ t'' = \tau}} \mu'_{r_p t'' r_{p-1}} \mu_{r_p t' r_{p-1}} k(\mathbf{x}_t, \mathbf{x}_{t'}) \delta_{\mathbf{x}_t}(\mathbf{x}_{t''}) \\
&= \left[\mathbf{M}' \circ (\mathbf{M} \times_2 \mathbf{K}) \right]_\tau.
\end{aligned} \tag{74}
$$

where $[\cdot]_\tau$ denotes slicing in the temporal dimension at index τ. We then calculate the first term

$$
\begin{aligned}
x^2_{r_p, i_t = \tau, r_{p+1}} &= \sum_{\substack{t' = 1 \ldots T \\ t'' = \tau \\ z' = 1 \ldots T \\ z'' = \tau}} v'_{r_p t'' r_{p-1}} v'_{r_p z'' r_{p-1}} v_{r_p t' r_{p-1}} k(\mathbf{x}_t, \mathbf{x}_{t'}) \delta_{\mathbf{x}_t}(\mathbf{x}_{t''}) k(\mathbf{x}_t, \mathbf{x}_{z'}) \delta_{\mathbf{x}_t}(\mathbf{x}_{z''}) \\
&= v'^2_{r_p \tau r_{p-1}} \sum_{\substack{t' = 1 \ldots T \\ z' = 1 \ldots T}} v_{r_p z' r_{p-1}} v_{r_p t' r_{p-1}} k(\mathbf{x}_t, \mathbf{x}_{t'}) k(\mathbf{x}_t, \mathbf{x}_{z'}).
\end{aligned} \tag{75}
$$

For univariate meanfield $\mathbf{V}, \mathbf{V}'$, taking the expectation we arrive at

$$\begin{aligned}
\mathbb{E}_q[x^2_{r_p,i_t=\tau,r_{p+1}}] &= \mathbb{E}_q[v'^2_{r_p\tau r_{p-1}}] \sum_{\substack{t'=1\dots T\\ z'=1\dots T}} \mathbb{E}_q[v_{r_p z' r_{p-1}} v_{r_p t' r_{p-1}}] k(\mathbf{x}_t, \mathbf{x}_{t'}) k(\mathbf{x}_t, \mathbf{x}_{z'}) \\
&= (\mu'^2_{r_p\tau r_{p-1}} + \sigma'^2_{r_p\tau r_{p-1}}) \left[\sum_{\substack{t'=1\dots T\\ z'=1\dots T}} \mu_{r_p z' r_{p-1}} \mu_{r_p t' r_{p-1}} k(\mathbf{x}_t, \mathbf{x}_{t'}) k(\mathbf{x}_t, \mathbf{x}_{z'}) \right. \\
&\quad \left. + \sum_{t'=1} \sigma^2_{r_p t' r_{p-1}} k(\mathbf{x}_t, \mathbf{x}_{t'})^2 \right] \\
&= \left[(\mathbf{M}'^2 + \Sigma') \circ \left[(\mathbf{M} \times_2 \mathbf{K})^2 + \Sigma \times_2 \mathbf{K}^2 \right] \right]_\tau
\end{aligned} \tag{76}$$

If we instead consider a multivariate meanfield $\mathbf{V}, \mathbf{V}'$, the expression becomes

$$\begin{aligned}
&\mathbb{E}_q[x^2_{r_p,i_t=\tau,r_{p+1}}] \\
&= (\mu'^2_{r_p\tau r_{p-1}} + \sigma'^2_{r_p\tau r_{p-1}}) \left[\sum_{\substack{t'=1\dots T\\ z'=1\dots T}} (\mu_{r_p z' r_{p-1}} \mu_{r_p t' r_{p-1}} + \sigma_{z'} \sigma_{t'}) k(\mathbf{x}_t, \mathbf{x}_{t'}) k(\mathbf{x}_t, \mathbf{x}_{z'}) \right] \\
&= \left[(\mathbf{M}'^2 + \Sigma') \circ \left[(\mathbf{M} \times_2 \mathbf{K})^2 + (\mathbf{1} \times_2 \left(\mathrm{diag}((\mathbf{K} \cdot \mathbf{B})^2 \cdot \bar{1}) \right) \right] \right]_\tau
\end{aligned} \tag{77}$$

For the third expression, we take

$$\begin{aligned}
\mathbb{E}_q[(\sum_{k=1}^K w_k x_{r_p,i_t=l_k,r_{p+1}})^2] &= \sum_{f=1}^K \sum_{k=1}^K w_k w_f \mathbb{E}_q[x_{r_p,i_t=l_k,r_{p+1}} x_{r_p,i_t=l_f,r_{p+1}}] \\
&= \underbrace{\sum_{k=1}^K w_k^2 \mathbb{E}_q[x^2_{r_p,i_t=l_k,r_{p+1}}]}_{:=\text{Expression 1}} + \underbrace{\sum_{k\neq f}^K w_k w_f \mathbb{E}_q[x_{r_p,i_t=l_k,r_{p+1}} x_{r_p,i_t=l_f,r_{p+1}}]}_{:=\text{Expression 2}}.
\end{aligned} \tag{78}$$

The tensor expression for Expression 1 is given by

$$\text{Expression 1} = \sum_{k=1}^K \mathbf{W}_k^2 \circ \left[(\mathbf{M}'^2 + \Sigma') \circ \left[(\mathbf{M} \times_2 \mathbf{K})^2 + \Sigma \times_2 \mathbf{K}^2 \right] \right]_{l_k}. \tag{79}$$

This follows directly from equation 76. Expression 2 can be written as

$$\text{Expression 2} = \left(\sum_{k=1}^K \mathbf{W}_k \circ \left[\mathbf{M}' \circ (\mathbf{M} \times_2 \mathbf{K}) \right]_{l_k} \right)^2 - \sum_{k=1}^K \mathbf{W}_k^2 \circ \left[\mathbf{M}' \circ (\mathbf{M} \times_2 \mathbf{K}) \right]^2_{l_k}. \tag{80}$$

Here the idea is to reduce the memory footprint by calculating a slightly erroneous expression and removing the error rather than calculate the exact expression due to the difficulty of expressing the cross terms $w_k w_f$. The above derivation extends directly to the LS case by

simply removing the $\mathbf{V}'$ variables and calculating the forecasting term for each component $\mathbf{V}_s, \mathbf{V}, \mathbf{V}_b$.

Appendix B: Derivation for WLR regularization term

$$
\begin{aligned}
&\sum_{r_1 r_2} \|\mathsf{v}_r\|^2_{\mathcal{H}} = \sum_{r_1 r_2} \langle \mathsf{v}_r, \mathsf{v}_r \rangle_{\mathcal{H}} \\
&= \sum_{r_1 r_2} \sum_{\substack{n_1,\ldots,n_q \\ n'_1,\ldots,n'_q}} v_{r_1 n_1 \ldots n_q r_1} v'_{r_1 n'_1 \ldots n'_q r_2} \prod_{q=1}^{Q} \delta_{\mathbf{x}^Q_{n'_q}}(\cdot) \prod_{q=1}^{Q} k(\cdot, \mathbf{x}^Q_{n_q}) \\
&\quad \sum_{\substack{\bar{n},\ldots,\bar{n}_q \\ \bar{n}'_1,\ldots,\bar{n}'_q}} v_{r_1 \bar{n}_1 \ldots \bar{n}_q r_2} v'_{r_1 \bar{n}'_1 \ldots \bar{n}'_q r_2} \prod_{q=1}^{Q} \delta_{\mathbf{x}^Q_{\bar{n}'_q}}(\cdot) \prod_{q=1}^{Q} k(\cdot, \mathbf{x}^Q_{\bar{n}_q}) \\
&= \sum_{r_1 r_2} \sum_{\substack{n_1,\ldots,n_q \\ n'_1,\ldots,n'_q \\ \bar{n}_1,\ldots,\bar{n}_q}} {v'_{r_1 n'_1,\ldots,n'_q r_2}}^2 v_{r_1 n_1 \ldots n_q r_2} v_{r_1 \bar{n}_1 \ldots \bar{n}_q r_2} \prod_{q=1}^{Q} k(\mathbf{x}^Q_{n_q}, \mathbf{x}^Q_{\bar{n}_q}) \\
&= \mathrm{tr}\left(\left(\prod_{q=1}^{Q} \otimes \mathbf{K}_q \right) \cdot \mathbf{V}^{\top}_{R_1 \cdot R_2 \times \prod_{q=1}^{Q} n_q} \cdot \left(\left(\prod_{q=1}^{Q} (\mathbf{V}' \circ \mathbf{V}') \times_q \mathbf{1}_{n_q \times n_q} \right) \circ \mathbf{V} \right)_{R_1 \cdot R_2 \times \prod_{q=1}^{Q} n_q} \right) \\
&= \mathrm{tr}\left(\left(\prod_{q=1}^{Q} \mathbf{V} \times_{q+1} \mathbf{K}_q \right)^{\top}_{R_1 \cdot R_2 \times \prod_{q=1}^{Q} n_q} \cdot \left(\left(\prod_{q=1}^{Q} (\mathbf{V}' \circ \mathbf{V}') \times_q \mathbf{1}_{n_q \times n_q} \right) \circ \mathbf{V} \right)_{R_1 \cdot R_2 \times \prod_{q=1}^{Q} n_q} \right) \\
&= \left(\left(\prod_{q=1}^{Q} \mathbf{V} \times_{q+1} \mathbf{K}_q \right) \circ \left(\left(\prod_{q=1}^{Q} (\mathbf{V}' \circ \mathbf{V}') \times_q \mathbf{1}_{n_q \times n_q} \right) \circ \mathbf{V} \right) \right)_{++}
\end{aligned}
\tag{81}
$$

B.1 RFF regularization term

The regularization term can similarly be generalized to

$$\sum_{r_1 r_2} \|\mathsf{v}_r\|_{\mathcal{H}}^2 = \sum_{r_1 r_2} \sum_{\substack{n_1,\ldots,n_q \\ n'_1,\ldots,n'_q \\ \bar{n}_1,\ldots,\bar{n}_q}} {v'_{r_1 n'_1,\ldots,n'_q r_2}}^2 v_{r_1 n_1 \ldots n_q r_2} v_{r_1 \bar{n}_1 \ldots \bar{n}_q r_2} \prod_{q=1}^{Q} k(\mathbf{x}^Q_{n_q}, \mathbf{x}^Q_{\bar{n}_q})$$

$$\approx \sum_{r_1 r_2} \sum_{\substack{n_1,\ldots,n_q \\ n'_1,\ldots,n'_q \\ \bar{n}_1,\ldots,\bar{n}_q}} {v'_{r_1 n'_1,\ldots,n'_q r_2}}^2 \prod_{q=1}^{Q} (v_{r_1 n_1 \ldots n_q r_2} \phi(\mathbf{x}^Q_{n_q}))^\top (v_{r_1 \bar{n}_1 \ldots \bar{n}_q r_2} \phi(\mathbf{x}^Q_{\bar{n}_q})) \tag{82}$$

$$= \left(\left(\prod_{q=1}^{Q} \mathbf{V} \times_{q+1} \boldsymbol{\Phi}_q \times_q \boldsymbol{\Phi}_q^\top \right) \circ \left(\left(\prod_{q=1}^{Q} (\mathbf{V}' \circ \mathbf{V}') \times_q \mathbf{1}_{n_q \times n_q} \right) \circ \mathbf{V} \right) \right)_{++}$$

Appendix C: Complexity proofs

We recall the theorem presented.

Theorem 2 *KFT has computational complexity*

$$\mathcal{O}\left(\max_p \left(n_p c_p r_p r_{p-1} \right) + \left(\max_p r_p \right)^P \right)$$

and memory footprint $\mathcal{O}\left(P \cdot \left(\max_p \left(n_p r_p r_{p-1}\right) + \max_p \left(n_p c_p\right)\right)\right)$ *for a gradient update on a batch of data. In the dual case, we take* $c_p = n_p$.

Proof When training on batches of data, the reconstruction for each datapoint is calculated by sequentially multiplying slices $\mathbf{V}_p(i_p), \mathbf{V}'_p(i_p) \in \mathbb{R}^{r_{p-1} \times r_p}$ from each TT-core. We start with the frequentist model for WLR and LS. We list the operations needed:

1. Any $\mathbf{V} \times_{p+1} \mathbf{D}_p$ operation. Complexity: $\mathcal{O}\left(\max_p \left(n_p c_p r_p r_{p-1}\right)\right)$. Memory: $\mathcal{O}\left(\max_p \left(n_p r_p r_{p-1}\right)\right) + \mathcal{O}\left(\max_p \left(n_p c_p\right)\right)$.
2. Any $\mathbf{V}_p \circ \mathbf{V}'_p$ operation. Complexity: $\mathcal{O}\left(\max_p \left(r_p r_{p-1}\right)\right)$. Memory: $\mathcal{O}\left(\max_p \left(r_p r_{p-1}\right)\right)$.
3. Reconstructing the tensor $\prod_{p=1}^{P} \times_{-1} \mathbf{V}_p$. Complexity: $\mathcal{O}\left(\left(\max_p r_p\right)^P\right)$. Memory: $\mathcal{O}\left(P \cdot \max_p \left(r_p r_{p-1}\right)\right)$.

For the Bayesian case we have the additional operations for the multivariate case:

1. $(\mathbf{K}_p \cdot \mathbf{B}_p)^2 \cdot \bar{1}$ component. Complexity: $\mathcal{O}\left(\max_p \left(n_p^2 k_p\right)\right)$. Here we take $\mathbf{B}_p \in \mathbb{R}^{n_p \times k_p}$ with $k_p \ll n_p$. Memory: $\mathcal{O}\left(\max_p \left(n_p^2\right)\right)$.
2. Prior determinant $\det\left(\Sigma_P\right)$. Complexity: $\mathcal{O}(\max_p(M_p^2))$, assuming we use RFFs when n_p is large. When using RFFs, we expect $M_p \ll N_p$. Memory: $\mathcal{O}(\max_p(n_p \cdot M_p))$.
3. Variational determinant $\det\left(\Sigma_Q\right)$. Complexity: $\mathcal{O}(\max_p(n_p))$, Memory: $\mathcal{O}(\max_p(n_p^2))$.

The dominating terms for complexity and memory are $\mathcal{O}\Big(\max_p \left(n_p c_p r_p r_{p-1}\right) + \left(\max_p r_p\right)^P\Big)$ and $\mathcal{O}\big(P \cdot \left(\max_p \left(n_p r_p r_{p-1}\right) + \max_p \left(n_p c_p\right)\right)\big)$ respectively. □

Appendix D: Interpretability example analysis

As a guiding example of utilizing the interpretability of KFT, we analyse the temporal component of WLR applied to the alcohol dataset in the primal setting. We visualize the weights for the time component and in particular the "year" covariate.

Applying an LS model instead, we can directly decompose the predictions into scaling contribution, regression contribution and bias contribution (Fig. 7; Table 11).

Appendix E: Training procedure

E.1 Frequentist

To motivate the sequential updating scheme, consider a data matrix $\frac{\mathbf{X}}{\sigma} \in \mathbb{R}^{N\times d}$ where σ is a hyperparameter that controls the scaling of $\mathbf{X}$ and a target $\mathbf{Y} \in \mathbb{R}^N$. Assume N is too large and we have to resort to stochastic first order gradient methods to approximate $\mu \in \mathbb{R}^d$ in our regression $\left(\frac{\mathbf{X}}{\sigma}\right)\mu \approx \mathbf{Y}$. Using autograd (Paszke et al. 2017a), together with ADAM (Kingma and Ba 2014) we can in practice optimize $\{\sigma, \mu\}$ simultaneously for each iteration. However doing this, we commit a fallacy as when updating $\mu_t = \mu_{t-1} - \frac{\partial \mathcal{L}(\sigma_{t-1}, \mu_{t-1})}{\partial \mu_{t-1}}$ and $\sigma_t = \sigma_{t-1} - \frac{\partial \mathcal{L}(\sigma_{t-1}, \mu_{t-1})}{\partial \sigma_{t-1}}$, we do not account for the mixed partial $\frac{\partial}{\partial \sigma_{t-1}} \frac{\partial \mathcal{L}(\sigma_{t-1}, \mu_{t-1})}{\partial \mu_{t-1}} \propto \frac{-1}{\sigma_{t-1}^3}$ assuming $\mathcal{L}$ is mean square error loss. Thus updating $\{\sigma, \mu\}$ simultaneously using first order derivatives would yield an update error for σ as the updating gradient does not adjust for the mixed partial $\frac{\partial}{\partial \sigma_{t-1}} \frac{\partial \mathcal{L}(\sigma_{t-1}, \mu_{t-1})}{\partial \mu_{t-1}}$ when μ is being updated at the same time. This scenario extends one-to-one for the variables of KFT, as we would commit a similar fallacy by updating all parameters at once. Hence we take an EM inspired approach when updating $l_p, \mathbf{V}_p, \mathbf{V}'_p$.

E.2 Bayesian: variational inference

As the practical utility of calibrated uncertainty estimates rely on a good fit of the model, we motivate our sequential update scheme as a method to encourage good predictive performance by first finding the optimal modes determined by $\mathbf{M}, \mathbf{M}'$ and then the associated variance determined by Σ, Σ'. As we are considering the Gaussian meanfield family of models, this strategy is well motivated as Gaussian distribution is symmetric.

Appendix F: Data and data processing

F.1 Data processing

Our strategy in this paper is to limit data processing to simple operations that does not require excessive engineering for a fair comparison in both utility and input data. For all the models, we carry out the exact same preprocessing modulo the models requirements of data format. We do the following general processing steps:

1. Extract relevant features and parse them to be continuous or categorical.
2. Scale all features using a z-transformation.

For each model specifically we do the following:

1. KFT: tensorize all data by expressing all main modes (i.e. person, movie, time etc) as a tensor with side information associated with each mode.
2. LightGBM: here, we don't scale the features as boosting trees generally performs better with unscaled data. In some cases we have applied PCA to some of the side information that was joined on the data matrix to decrease the memory footprint of the data matrix to contain it to a practical size.
3. FFM: here we bin all continuous features, as FFM requires all data to be categorical.
4. Linear regression: for large categorical features, we using feature hashing to avoid data matrices of infeasible sizes. All other categorical features get one-hot encoded.

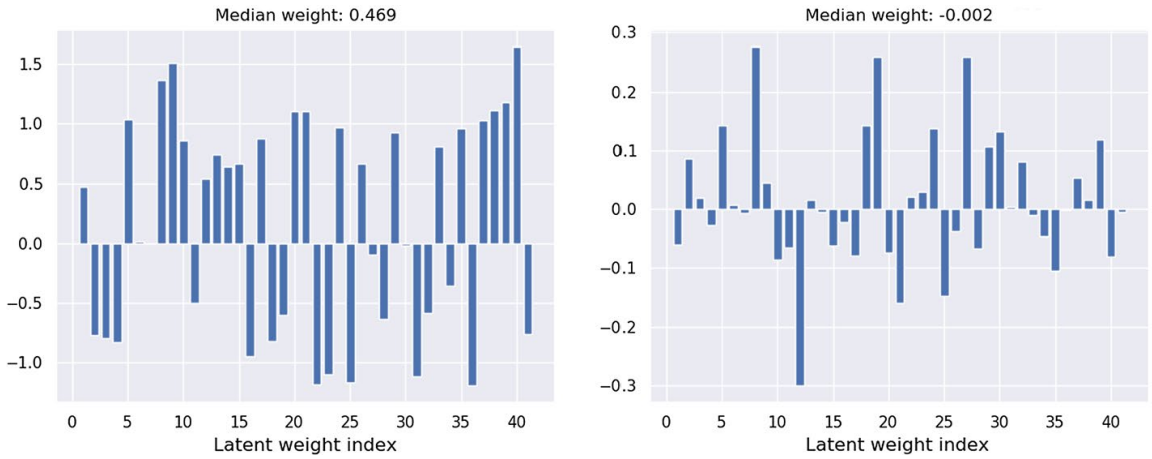


Fig. 7 The first plot shows the magnitude of the latent weights $\mathbf{V}$ for the "year" covariate. The second plot show the weights of the latent weights $\mathbf{V}'$ for year = 0

Table 11 5 decomposed example predictions for the LS model

Prediction nr	$\mathbf{V}_s$	$\mathbf{V}$	$\mathbf{V}_b$
0	−0.032064	46.948650	15.560461
1	−0.001596	333.260956	0.218784
2	−0.110225	29.397287	13.096196
3	0.028649	−45.733292	10.711301
4	−0.067812	−34.203766	3.496001

F.2 Retail sales data

We detail the features of the Retail Sales data in Table 12. Here we choose our modes to be store, articles and time.

F.3 Movielens-20M

We detail the features of the Retail Sales data in Table 13. Here we choose our modes to be users, movies and time of rating given. For the Movielens-20M data, it should be noted that we filter the movies on existing entries in the side information. This is why we only have roughly 11 million observations rather than 20 million.

F.4 Alcohol sales

We detail the features of the Alcohol Sales data in Table 14. Here we choose our modes to be location, item and time.

Appendix G: Hyperparameter configuration

G.1 KFT hyperparameters

We run all KFT experiments for 10 epochs with 20 hyperparameter search iterations. We consider two decomposition types:

1. P-way latent factorization, where each dimension has a latent component. In principle, this can be thought of as each dimension being independently factorized. Further we utilize all possible side information.
2. 2-way latent factorization, where time is grouped with another dimension as one latent component and with the other dimensions grouped in a second latent component. Here we only consider time as side information, for the purpose of only modelling temporal changes.

Our models are generally searched over the configurations described in Table 15. For exact details we refer to the code base.

Table 12 Dataset description for retail sales data

Property	Unit	Quantity
Monthly sales	Count	42,490,633
Unique location_id's	Count	80
Unique article_id's	Count	463,763
Article_id side info	Description	Example
Appearance_id	Article style	26
Colour_id	Article color	1337
Product_id	Article group	15
Size_id	Article size	L
Department_id	Article group theme	Teen
Product_season_id	Article group season	Spring
Product_type_id	Product group type	Shirt
Product_group_no	Product group no	Accessories
Price	Article price	5usd
Location_id side info	Description	Example
City_id	City of store	56
Longitude	Longitude	34.23
Latitude	Latitude	34.24
Brand_id	Brand	12
Opening_year_id	Opening year	123
Opening_month_id	Opening month	92
Opening_day_id	Opening day	97
Opening_hours_mo-su	Hours open mo-su	12,...,10
No_of_floors	Store no. of floors	10
Total_sales_area	Store area size	1231 sqm
Special_sales_area	Special sales area	789 sqm

Table 12 (continued)

Time side info	Description	Example
Year	Year	2019
Month	Month	1

Note that we only present the ID of each feature to preserve anonymity and that the examples are not real

Table 13 Dataset description of Movielens-20M

Property	Unit	Quantity
Ratings	Count	11,880,265
Users	Count	138,493
Movie_id	Count	10,370
Movie_id side info	Description	Example
Genome_score_1	Genome score dimension 1	0.345
⋮	⋮	⋮
Genome_score_1000	Genome score dimension 1000	0.1337
Time side info	Description	Example
Hour	Hour ratings was given	12

Table 14 Dataset description of Iowa alcohol sales

Property	Unit	Quantity
Bottles sold	Count	3,036,063
Unique store_location_id's	Count	3476
Unique item_id's	Count	4542
Item_id side info	Description	Example
Category	Type of alcohol	Irish whiskey
Pack	Size of package	6
Bottle volume (ml)	Bottle volume	750
State bottle cost	Cost for retailer to buy	4 usd
State bottle retail	Retail price	5 usd
Store_location_id side info	Description	Example
City_id	City of store	56
Longitude	Longitude	34.23
Latitude	Latitude	34.24
County	County	Shelby
Store_number	Store number	12
Zip code	Zip code	157
Time side info	Description	Example
Year	Year	2019
Month	Month	1

Table 15 Hyperparameter search space for KFT.

Property	Range/choices
Batch size[a]	[0.01, 0.1]
Learning rate	{1e−3, 1e−2, 1e−1}
R^b	[5, 70]
λ_p	[0, 1]
λ'_p	[0, 1]
Kernel choice	{rbf, matern 0.5, matern 1.5, matern 2.5}

[a]The upper bound varies for each dataset, as memory limitation changes with dataset.

[b]Batch size expressed as a proportion of total training data examples. As an example our smallest batch size use 1% of the training data as the batch size

G.1.1 LightGBM hyperparameters

We provide hyperparameters for LightGBM in Table 16

Table 16 Hyperparameter search space for LightGBM.

Property	Range/choices
Num leaves	[7, 4095]
Learning rate	[exp(−5), exp(−2.3)]
Min data in leaf	[10, 30]
Min sum hessian in leaf	[exp(0), exp(2.3)]
Bagging freq	[1, 5]
λ_1	[0, 10]
λ_2	[0, 10]

G.1.2 FFM hyperparameters

We provide hyperparameters for FFM in Table 17.

Table 17 Hyperparameter search space for FFM.

Property	Range/choices
R	[2, 20]
Batch size	[1e−6, 0.1]
Learning rate	[0.05, 1.0]
λ	[0, 0.005]

G.1.3 Linear regresison hyperparameters

We provide hyperparameters for linear regression in Table 18.

Table 18 Hyperparameter search space for linear regression.

Property	Range/choices
Batch size	[1e−6, 0.1]
Learning rate	[0.05, 1.0]
λ	[0, 1.0]

G.2 Bayesian hyperparameters

We run all experiments for 25 epochs with 5 hyperparameter search iterations. For exact details we refer to the code base.

Open Access This article is licensed under a Creative Commons Attribution 4.0 International License, which permits use, sharing, adaptation, distribution and reproduction in any medium or format, as long as you give appropriate credit to the original author(s) and the source, provide a link to the Creative Commons licence, and indicate if changes were made. The images or other third party material in this article are included in the article's Creative Commons licence, unless indicated otherwise in a credit line to the material. If material is not included in the article's Creative Commons licence and your intended use is not permitted by statutory regulation or exceeds the permitted use, you will need to obtain permission directly from the copyright holder. To view a copy of this licence, visit http://creativecommons.org/licenses/by/4.0/.

References

Agarwal, D., & Chen, B. C. (2009). *Regression-based latent factor models* (Vol. '09, pp. 19–28). Association for Computing Machinery. https://doi.org/10.1145/1557019.1557029

Batselier, K. (2018). The trouble with tensor ring decompositions. Preprint

Bergstra, J., Yamins, D., Cox, D. D. (2013) Making a Science of Model Search: Hyperparameter Optimization in Hundreds of Dimensions for Vision Architectures. Proc. of the 30th International Conference on Machine Learning (ICML 2013).

Bobadilla, J., Ortega, F., Hernando, A., & Bernal, J. (2012). A collaborative filtering approach to mitigate the new user cold start problem. *Knowledge-Based Systems, 26*, 225–238. https://doi.org/10.1016/j.knosys.2011.07.021

Bobadilla, J., Ortega, F., Hernando, A., & Gutiérrez, A. (2013). Recommender systems survey. *Knowledge-Based Systems, 46*, 109–132. https://doi.org/10.1016/j.knosys.2013.03.012

Caro, F., & Gallien, J. (2010). Inventory management of a fast-fashion retail network. *Operations Research*. https://doi.org/10.1287/opre.1090.0698

Dempster, A. P., Laird, N. M., & Rubin, D. B. (1977). Maximum likelihood from incomplete data via the em algorithm. *Journal of the Royal Statistical Society. Series B (Methodological)* **39**(1), 1–38. http://www.jstor.org/stable/2984875

Draper, N., & Smith, H. (1966). *Applied regression analysis. Wiley series in probability and mathematical statistics*. Wiley. http://gso.gbv.de/DB=2.1/CMD?ACT=SRCHA&SRT=YOP&IKT=1016&TRM=ppn+022791892&sourceid=fbw_bibsonomy

Du, Chao, Chongxuan Li, Yin Zheng, Jun Zhu, and Bo Zhang. (2018). Collaborative Filtering With User-Item Co-Autoregressive Models. *Proceedings of the AAAI Conference on Artificial Intelligence* **32**(1). https://ojs.aaai.org/index.php/AAAI/article/view/11884.

Gönen, M., Khan, S., & Kaski, S. (2013). In S. Dasgupta & D. McAllester (Eds.), *Kernelized Bayesian matrix factorization* (Vol. 28, pp. 864–872). PMLR. http://proceedings.mlr.press/v28/gonen13a.html

Hawkins, C., & Zhang, Z. (2018). Variational Bayesian Inference for Robust Streaming Tensor Factorization and Completion. 2018 IEEE International Conference on Data Mining (ICDM), 1446–1451.

He, L., Lu, C. T., Ma, G., Wang, S., Shen, L., Yu, P. S., & Ragin, A. B. (2017). In D. Precup & Y. W. Teh (Eds.) *Kernelized support tensor machines* (Vol. 70, pp. 1442–1451). PMLR, International Convention Centre. http://proceedings.mlr.press/v70/he17a.html

Hoffman, M. D., Blei, D. M., Wang, C., & Paisley, J. (2013). Stochastic variational inference. *Journal of Machine Learning Research,* **14**(1), 1303–1347. http://dl.acm.org/citation.cfm?id=2502581.2502622

Juan, Y., Zhuang, Y., Chin, W. S., & Lin, C. J. (2016). Field-aware factorization machines for ctr prediction. In *Proceedings of the 10th ACM Conference on Recommender Systems, RecSys '16* (pp. 43–50). ACM. https://doi.org/10.1145/2959100.2959134

Ke, G., Meng, Q., Finley, T., Wang, T., Chen, W., Ma, W., Ye, Q., & Liu, T. Y. (2017). Lightgbm: A highly efficient gradient boosting decision tree. In *NeurIPS*.

Kim, H., Lu, X., Flaxman, S., & Teh, Y. W. (2016). Collaborative filtering with side information: A Gaussian process perspective. Preprint

Kim, H., & Teh, Y.W. (2018). Scaling up the Automatic Statistician: Scalable Structure Discovery using Gaussian Processes. AISTATS.

Kim, Y. D., & Choi, S. (2014). Scalable variational Bayesian matrix factorization with side information. In *AISTATS*.

Kingma, Diederik & Ba, Jimmy. (2014). Adam: A Method for Stochastic Optimization. International Conference on Learning Representations.

Kolda, T. G., & Bader, B. W. (2009). Tensor decompositions and applications. *SIAM Review, 51*(3), 455–500.

Kuang, L., Hao, F., Yang, L. T., Lin, M., Luo, C., & Min, G. (2014). A tensor-based approach for big data representation and dimensionality reduction. *IEEE Transactions on Emerging Topics in Computing, 2*(03), 280–291. https://doi.org/10.1109/TETC.2014.2330516

Liu, T., Wang, Z., Tang, J., Yang, S., Huang, G. Y., & Liu, Z. (2019). *Recommender systems with heterogeneous side information* (Vol. '19, pp. 3027–3033). Association for Computing Machinery. https://doi.org/10.1145/3308558.3313580

Narita, A., Hayashi, K., Tomioka, R., & Kashima, H. (2012). Tensor factorization using auxiliary information. *Data Mining and Knowledge Discovery, 25*(2), 298–324. https://doi.org/10.1007/s10618-012-0280-z

Ong, Victor & Nott, David & Smith, Michael. (2017). Gaussian Variational Approximation With a Factor Covariance Structure. Journal of Computational and Graphical Statistics. 27. https://doi.org/10.1080/10618600.2017.1390472.

Oseledets, I. V. (2011). Tensor-train decomposition. *SIAM Journal on Scientific Computing, 33*(5), 2295–2317. https://doi.org/10.1137/090752286

Pal, B., & Jenamani, M. (2018). *Kernelized probabilistic matrix factorization for collaborative filtering: Exploiting projected user and item graph* (pp. 437–440). Association for Computing Machinery. https://doi.org/10.1145/3240323.3240402

Paszke, A., Gross, S., Chintala, S., Chanan, G., Yang, E., DeVito, Z., Lin, Z., Desmaison, A., Antiga, L. & Lerer, A. (2017). Automatic Differentiation in PyTorch. NIPS 2017 Workshop on Autodiff.

Rahimi, A., & Recht, B. (2007). *Random features for large-scale kernel machines* (Vol. NIPS'07, pp. 1177–1184). Curran Associates Inc.

Rendle, S. (2010). Factorization machines. In G.I. Webb, B. Liu, C. Zhang, D. Gunopulos, & X. Wu (Eds.) *ICDM 2010, The 10th IEEE international conference on data mining, Sydney, Australia, 14–17 December 2010* (pp. 995–1000). IEEE Computer Society. https://doi.org/10.1109/ICDM.2010.127

Rudin, Cynthia. (2019). Stop explaining black box machine learning models for high stakes decisions and use interpretable models instead. *Nature Machine Intelligence.* 1, 206–215. https://doi.org/10.1038/s42256-019-0048-x.

Saha, A., Misra, R., Acharya, A., & Ravindran, B. (2017). Scalable variational Bayesian factorization machine. Preprint

Salakhutdinov, R., & Mnih, A. (2007). Probabilistic matrix factorization. In *Proceedings of the 20th international conference on neural information processing systems, NeurIPS* (pp. 1257–1264). Curran Associates Inc. http://dl.acm.org/citation.cfm?id=2981562.2981720

Flunkert, Valentin & Salinas, David & Gasthaus, Jan. (2017). DeepAR: Probabilistic Forecasting with Autoregressive Recurrent Networks. International Journal of Forecasting. 36. https://doi.org/10.1016/j.ijforecast.2019.07.001.

Schölkopf, B., Herbrich, R., & Smola, A. J. (2001). A generalized representer theorem. In *COLT/EuroCOLT*.

Seeger, M., Salinas, D., & Flunkert, V. (2016). Bayesian intermittent demand forecasting for large inventories. In *NeurIPS*.

Vovk, V. (2013). Kernel Ridge Regression. Empirical Inference.

Wu, X., Shi, B., Dong, Y., Huang, C., & Chawla, N. V. (2019). *Neural tensor factorization for temporal interaction learning* (Vol. WSDM '19, pp. 537–545). Association for Computing Machinery. https://doi.org/10.1145/3289600.3290998

Xu, J., Yao, Y., Tong, H., Tao, X., & Lu, J. (2015). Ice-breaking: Mitigating cold-start recommendation problem by rating comparison. In *IJCAI international joint conference on artificial intelligence. 24th international joint conference on artificial intelligence, IJCAI 2015; conference date: 25-07-2015 through 31-07-2015* (Vol. 2015, pp. 3981–3987).

Yu, H., Rao, N.S., & Dhillon, I.S. (2016). Temporal Regularized Matrix Factorization for High-dimensional Time Series Prediction. NIPS.

Yu, R., Bahadori, M. T., & Liu, Y. (2014). Fast multivariate spatio-temporal analysis via low rank tensor learning. In *Advances in neural information processing systems* (pp. 3491–3499).

Zellner, A. (1986). On Assessing Prior Distributions and Bayesian Regression Analysis With g-Prior Distributions. Basic Bayesian Inference and Decision Techniques: Essays in Honor of Bruno de Finetti

Zhai, Y., Ong, Y., & Tsang, I. W. (2014). The emerging "big dimensionality". *IEEE Computational Intelligence Magazine, 9*(3), 14–26. https://doi.org/10.1109/MCI.2014.2326099

Zhang, J., Shi, X., Zhao, S., & King, I. (2019). STAR-GCN: Stacked and Reconstructed Graph Convolutional Networks for Recommender Systems. IJCAI.

Zhang, M., Tang, J., Zhang, X., & Xue, X. (2014). Addressing cold start in recommender systems: A semi-supervised co-training algorithm. *Association for Computing Machinery.* **14**, 73–82 https://doi.org/10.1145/2600428.2609599

Zhao, Q., Zhou, G., Adali, T., Zhang, L., & Cichocki, A. (2013). Kernelization of tensor-based models for multiway data analysis: Processing of multidimensional structured data. *IEEE Signal Processing Magazine, 30*(4), 137–148. https://doi.org/10.1109/MSP.2013.2255334

Zhao, Q., Zhou, G., Xie, S., Zhang, L., & Cichocki, A. (2016). Tensor ring decomposition. Preprint

Publisher's Note Springer Nature remains neutral with regard to jurisdictional claims in published maps and institutional affiliations.

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	Large scale tensor regression using kernels and variational inference
Publication Status	☒Published ☐ Accepted for Publication ☐Submitted for Publication ☐Unpublished and unsubmitted work written in a manuscript style
Publication Details	Hu, R., Nicholls, G.K. & Sejdinovic, D. (2020) Large scale tensor regression using kernels and variational inference. Mach Learn (2021). https://doi.org/10.1007/s10994-021-06067-7

Student Confirmation

Student Name:	Robert Hu		
Contribution to the Paper	My contributions were the implementation, experiments and proofs. During our weekly meetings, my supervisors contributed to this work by providing valuable insights and helpful remarks. They also contributed a lot to the writing of the draft, checking the proofs, and proof-reading.		
Signature		Date	29/8-2022

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: Dino Sejdinovic		
Supervisor comments I am in agreement with the description of the contributions.		
Signature 	Date	09/09/2022

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

5

Survival Regression with Proper Scoring Rules and Monotonic Neural Networks

Survival Regression with Proper Scoring Rules and Monotonic Neural Networks

David Rindt* **Robert Hu*** **David Steinsaltz** **Dino Sejdinovic**
University of Oxford

Abstract

We consider frequently used scoring rules for right-censored survival regression models such as time-dependent concordance, survival-CRPS, integrated Brier score and integrated binomial log-likelihood, and prove that neither of them is a proper scoring rule. This means that the true survival distribution may be scored worse than incorrect distributions, leading to inaccurate estimation. We prove that, in contrast to these scores, the right-censored log-likelihood is a proper scoring rule, i.e., the highest expected score is achieved by the true distribution. Despite this, modern feed-forward neural-network-based survival regression models are unable to train and validate directly on the right-censored log-likelihood, due to its intractability, and resort to the aforementioned alternatives, i.e., non-proper scoring rules. We therefore propose a simple novel survival regression method capable of directly optimizing log-likelihood using a monotonic restriction on the time-dependent weights, coined SurvivalMonotonic-net (SuMo-net). SuMo-net achieves state-of-the-art log-likelihood scores across several datasets with 20–100× computational speedup on inference over existing state-of-the-art neural methods, and is readily applicable to datasets with several million observations.

*Equal contribution, author order decided by coin flip. Correspondence to: robert.hu@stats.ox.ac.uk.

Proceedings of the 25th International Conference on Artificial Intelligence and Statistics (AISTATS) 2022, Valencia, Spain. PMLR: Volume 151. Copyright 2022 by the author(s).

1 INTRODUCTION AND RELATED WORKS

Survival analysis is a class of statistical methods for analyzing the time until the occurrence of an event, such as death in biological organisms (Ziehm and Thornton, 2013) or failure in mechanical systems (Susto et al., 2015). We focus on survival regression, where we relate the time that passes before some event occurs to one or more covariates that may be associated with that quantity of time. In particular, we consider the case in which the event times are right-censored, meaning that for some individuals it is known their event happened after a certain time, but not exactly when. For example, a patient who is still alive when the study ends would be right censored. Applications of survival regression on right-censored data are numerous, with the most classical example being the study of recovery time from medical treatments (Laurie et al., 1989). Other applications include modeling the time until the default of a business (Dirick et al., 2017), and churn prediction (Van den Poel and Lariviere, 2004).

Related Work Classical models for survival regression include the Cox proportional hazards model (Cox, 1972), the accelerated failure time (MacKenzie, 1982), as well as various parametric models (Kleinbaum and Klein, 2005). Over the last decade, many machine learning methods for survival data have been proposed. We here mention the most relevant methods which allow more flexible survival functions than the Cox model, and refer to Kvamme et al. (2019) for further references. In Katzman et al. (2018) the Cox model is extended by replacing the linear function with a neural network. Kvamme et al. (2019) further extends on this idea and includes the survival time to the neural network, achieving state-of-the-art performance. In Ishwaran et al. (2008) a random-forest approach is taken, in which cumulative hazard functions are computed using the Nelson–Aalen estimator. In Lee et al. (2018) a discrete distribution is learned by optimizing a mix of the discrete likelihood and a rank-based score. Ranganath et al. (2016) and Miscouridou et al. (2018) first learn a latent representation of the covariates, and then

learn a survival function using the latent representation. In particular, the latter method can deal with missing data in the context of electronic health records. In Chapfuwa et al. (2018) an adversarial model is used.

Intractable Likelihood As we model a survival *distribution*, it is natural to consider approaches based on maximizing the likelihood. In the particular case of right-censoring, we later prove that the right-censored likelihood actually is a proper scoring rule (Gneiting and Raftery, 2007), meaning that the highest expected score is obtained by the true distribution. However, none of the survival regression methods overviewed above have a tractable likelihood, impeding its direct use in estimation. In particular, Kvamme et al. (2019); Katzman et al. (2018) and Ishwaran et al. (2008) all learn discrete distributions, consisting of point masses at fixed times. In Lee et al. (2018), time is discretized into bins and while Ranganath et al. (2016), and Miscouridou et al. (2018) take a Bayesian approach, the resulting posterior distribution is also intractable. In each case computing the likelihood of new observations is problematic, as no method has direct access to the cumulative survival distribution. Some recent work by Groha et al. (2021) and Tang et al. (2020) proposes NeuralODEs to model the right-censored survival likelihood directly as a differential equation in time. While NeuralODEs are a very flexible class of models, they instead have pathologies relating to scalability (Dupont et al., 2019), since they scale polynomially in the number of parameters used (Massaroli et al., 2021).

Alternative scoring rules While the likelihood is a standard measure of model fit for uncensored data, due to its intractability in right-censored data one typically resorts to alternative scoring rules. The most commonly used scores to evaluate the fit of a right-censored regression model are the time-dependent concordance (Antolini et al., 2005), the survival-CRPS (Avati et al., 2020), the Brier score for right-censored data (Graf et al., 1999) and the binomial log-likelihood. In this paper we explore pathologies of these alternative scoring rules and propose *SurvivalMonotonic neural net* (SuMo-net) as a scalable and flexible survival regression model that optimizes likelihood directly. We summarize our contributions as:

1. We show theoretically and experimentally that time-dependent concordance, (integrated) Brier score and binomial log-likelihood, survival-CRPS are not proper scoring rules, meaning inaccurate distributions may achieve better scores than the true distribution.
2. We give a simple proof that right-censored log-likelihood is a proper scoring rule.
3. We introduce a novel survival regression model which directly optimizes right-censored log-likelihood and compare it to the existing methods in terms of performance and scalability, achieving state-of-the-art results on a range of datasets, together with 20–100 times speedups. Codebase: `https://github.com/MrHuff/Sumo-Net.git`

The paper is organized as follows: In Section 2 we introduce the notation used for survival regression. In Section 3, we define what a proper score is and show that time-dependent concordance, (integrated) Brier score and binomial log-likelihood, survival-CRPS are not proper scores. We further prove that the right-censored likelihood is a proper score. *SurvivalMononotic* neural net is introduced in Section 4. Section 5 provides extensive experiments. We conclude our work in Section 6.

2 BACKGROUND AND NOTATION

We are interested in an event-time $T \in \mathbb{R}_{\geq 0}$ and how it depends on a covariate $X \in \mathbb{R}^p$ for $p \geq 1$. We consider the case in which T is subject to right-censoring, where the event-time T is not known for every individual in the sample. Instead of observing T directly, we observe $Z = \min\{T, C\}$ where $C \in \mathbb{R}_{\geq 0}$ is a censoring time, as well as the indicator $D = 1\{Z = T\}$ indicating if we observe the event-time. A sample of size n can thus be denoted by $\{(X_i, Z_i, D_i)\}_{i=1}^n$, where we use the uppercase letters when we treat the dataset as random and lowercase letters otherwise. We let $S(t|x) \coloneqq 1 - F(t|x) = \mathbb{P}(T > t|X = x)$ denote the (conditional) survival distribution, and, assuming T has a density, we let $f(t|x) \coloneqq \frac{d}{dt}F(t|x)$ denote the density of the event-time.

The goal of this work is the following: If we let $S(t|x)$ denote the survival function of T, our aim is to estimate S with $\hat{S}(t|x)$ based on the sample $\{(X_i, Z_i, D_i)\}_{i=1}^n$. We also look at ways to assess if $\hat{S}$ is an accurate estimate of the true distribution S.

In survival analysis one typically assumes that censoring is not informative about the event-time conditional on the covariate, which we formally state in the following assumption.

Assumption 1. *(Independent censoring) We assume that conditionally on the covariate, the censoring- and event-time are independent. That is, we assume* $T \perp\!\!\!\perp C|X$.

Let $S(t|x)$ be a survival function and let $f(t|x) = -\partial S(t|x)/\partial t$. Then under Assumption 1, the right-censored log-likelihood is defined as:

$$\log L_n = \sum_{i=1}^{n} d_i \log f(z_i|x_i) + (1 - d_i) \log S(z_i|x_i). \quad (1)$$

Right-censored survival regression

$$\begin{cases} F(t|x) = \int_0^t f(t'|x)dt' \\ S(t|x) = 1 - F(t|x) \\ S_Z(t|x) = 1 - \int_0^t f(t'|x)1_{t'<Z}dt' \end{cases}$$

Observed event-time	t_i	$\mathbb{R}_{\geq 0}$
Censoring time	c_i	$\mathbb{R}_{\geq 0}$
Right-censored times	$z_i = \min\{t_i, c_i\}$	$\mathbb{R}_{\geq 0}$
Indicator of observed event-time	d_i	$\{0, 1\}$
Covariates	x_i	$\mathbb{R}^d$

Both the density- and the survival function imply that, if one models the density, then the likelihood contains n integrals which need to be estimated, making it a difficult optimization objective (see e.g. p. 288 of Gu (2013)). For that reason many methods optimize the partial likelihood instead of the right-censored likelihood. (Cox, 1972).

3 EVALUATION CRITERIA AND PROPER SCORING RULES

A scoring rule $\mathcal{S}$ takes a distribution S over some set $\mathcal{Y}$ and an observed value $y \in \mathcal{Y}$ and returns a score $\mathcal{S}(S, y)$, assessing how well the model predicts the observed value. For a positive scoring rule, a higher score indicates a better model fit. A scoring rule is called proper if the true distribution achieves the optimal expected score, i.e., if

$$\mathbb{E}_{y\sim S}\mathcal{S}(S, y) \geq \mathbb{E}_{y\sim S}\mathcal{S}(\hat{S}, y) \quad \text{for all distributions } \hat{S}.$$

In the context of survival regression, we call the scoring rule $\mathcal{S}$ proper if for every true distribution S, every censoring distribution C and for every covariate x it holds that

$$\mathbb{E}_{T,C|X=x}\mathcal{S}\big(S(t|x), (Z, D)\big) \geq \mathbb{E}_{T,C|X=x}\mathcal{S}\big(\hat{S}(t|x), (Z, D)\big)$$

for every distribution $\hat{S}(t|x)$, where in the above expression $T|X = x \sim S(t|x)$. By taking the expectation with respect to X in the above expression, we find that if $\mathcal{S}$ is a proper score, then also

$$\mathbb{E}_{T,C,X}\mathcal{S}\big(S(t|X), (Z, D)\big) \geq \mathbb{E}_{T,C,X}\mathcal{S}\big(\hat{S}(t|X), (Z, D)\big)$$

for every family of distributions $\hat{S}(t|x)$ with $x \in \mathbb{R}^p$.

3.1 Existing evaluation metrics are not proper scoring rules

We now show that time-dependent concordance (Antolini et al., 2005), the Brier score for right-censored data (Graf et al., 1999), the binomial log-likelihood and the survival-CRPS (Avati et al., 2020) are not proper scoring rules. For time-dependent concordance and the survival-CRPS, these are a novel results. Time-dependent concordance is not to be confused with (Blanche et al., 2018), which considers *time-independent* concordance.

Time-dependent concordance Let (X, Z, D) and (X', Z', D') be i.i.d. covariates and right-censored event-times. Time-dependent concordance $\mathcal{S}_{C^{\text{td}}}$ (Antolini et al., 2005) is defined by

$$\mathbb{P}\left(\hat{S}(Z|X) < \hat{S}(Z|X')|Z \leq Z', D = 1\right) + \frac{1}{2}\mathbb{P}\left(\hat{S}(Z|X) = \hat{S}(Z|X')|Z \leq Z', D = 1\right)$$

where ties in the survival probabilities are dealt with as proposed in Ishwaran et al. (2008) (Step 3 of Section 5.1). It has been remarked that poorly calibrated models can still have a high time-dependent concordance (Kvamme et al., 2019). Rizopoulos et al. (2014) and Antolini et al. (2005) have pointed out that censoring affects time-dependent concordance. We now show that time-dependent concordance is not a proper score by constructing an example where a false distribution has higher score than the true distribution.

Example: optimizing concordance for a binary covariate. Let $X, X' \sim$ Bernoulli(1/2) independently. Then $\hat{S}(\cdot|\cdot)$ maximizes $\mathcal{S}_{C^{\text{td}}}$ if and only if it maximizes

$$\begin{aligned} &\mathbb{P}\big(\hat{S}(Z|X) < \hat{S}(Z|X'), Z \leq Z', D = 1|X = 1, X' = 0\big) + \\ &\mathbb{P}\big(\hat{S}(Z|X) < \hat{S}(Z|X'), Z \leq Z', D = 1|X = 0, X' = 1\big) = \\ &\int_0^\infty \big(1\{\hat{S}(t|1) < \hat{S}(t|0)\} f_T(t|1)S_C(t|1)S_T(t|0)S_C(t|0) \\ &+ 1\{\hat{S}(t|0) < \hat{S}(t|1)\} f_T(t|0)S_C(t|0)S_T(t|1)S_C(t|1)\big)dt. \end{aligned}$$

This expression is optimized by constructing $\hat{S}(\cdot|\cdot)$ such that $\hat{S}(t|1) < \hat{S}(t|0)$ if and only if

$$\begin{aligned} &f_T(t|1)S_C(t|1)S_T(t|0)S_C(t|0) \geq \\ &f_T(t|0)S_C(t|0)S_T(t|1)S_C(t|1) \iff \lambda_T(t|1) \geq \lambda_T(t|0) \end{aligned}$$

where $\lambda_T(t|x) = f_T(t|x)/S_T(t|x)$ is called the hazard rate. Hence, in this example, to maximize concordance we need the ordering of $\hat{S}(t|1)$ and $\hat{S}(t|0)$ to be the reversed order of the true hazard rates. In Figure 1, we use this insight to generate a false distribution with concordance much higher than the concordance of the true distribution (Example 1). See Appendix 1.2 for details.

Brier score The Brier score at time t for the estimated survival distribution $\hat{S}(\cdot|\cdot)$ is typically defined

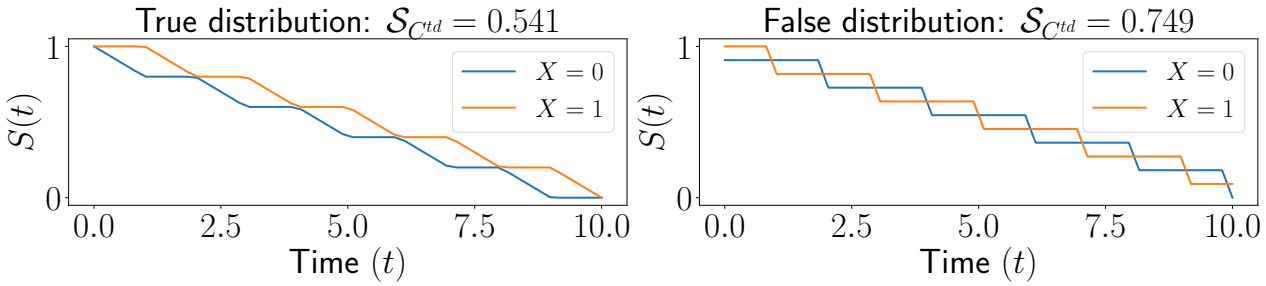


Figure 1: The survival curves for the true distribution on the left and a false distribution, designed to optimize concordance, on the right. By drawing a sample of size 1000 from the true distribution, we find that the concordance score of the true distribution is lower than the score of the false distribution, confirming that an inaccurate distribution can have high concordance.

	Example 1		Example 2					Example 3	
	$\mathcal{S}_L \uparrow$	$\mathcal{S}_{C^{\text{td}}} \uparrow$	$\mathcal{S}_L \uparrow$	$\mathcal{S}_{\text{Brier}}^{t=4.0} \downarrow$	$\mathcal{S}_{\text{BLL}}^{t=4.0} \uparrow$	$\mathcal{S}_{\text{IBS}} \downarrow$	$\mathcal{S}_{\text{IBLL}} \uparrow$	$\mathcal{S}_L \uparrow$	$\mathcal{S}_{\text{CRPS}} \downarrow$
True distribution score	-1.375	0.541	-1.795	0.220	0.085	0.032	-0.009	-0.513	3.884
Fake distribution score	-11.029	0.749	-1.820	0.204	0.122	0.031	-0.006	-0.626	1.646
Consistent?	✓	×	✓	×	×	×	×	✓	×

Table 1: Scores for true and fake distribution respectively. Bottom line indicates if the true distribution scores better than the fake distribution. Right-censored likelihood consistently scores the true distribution higher than a fake distribution, in contrast to non-proper scores.

as (Graf et al., 1999)

$$\mathcal{S}^t_{\text{Brier}}(\hat{S}(t|x), (z, d)) = \frac{\hat{S}(t|x)^2 1\{z \le t, d = 1\}}{\hat{G}(z)} + \frac{(1 - \hat{S}(t|x))^2 1\{z > t\}}{\hat{G}(t)}$$

where $\hat{G}$ is the Kaplan-Meier estimate of the censoring distribution. Assuming that the estimated survival function of the censoring time C, $\hat{G}(\cdot)$ is the exact marginal distribution $G(\cdot) = \mathbb{P}(C > \cdot)$, then the expectation, conditional on the X_i equals

$$\mathbb{E}\big[\mathcal{S}_{\text{Brier}}(\hat{S}(t|x), (Z, D))|X = x\big] = \tag{2}$$

$$\hat{S}(t|x)^2 \int_0^t \frac{G(z|x)}{G(z)} f_T(z|x) dz \tag{3}$$

$$+ (1 - \hat{S}(t|x))^2 \frac{G(t|x)}{G(t)} S_T(t|x). \tag{4}$$

If $C \perp\!\!\!\perp X$, and $G(\cdot|x) = G(\cdot)$, for all x, then it is easy to see the above equals

$$\hat{S}(t|x)^2(1 - S_T(t|x)) + (1 - \hat{S}(t|x))^2 S_T(t|x).$$

By setting $a_i = \hat{S}(t|x)$ and optimizing with respect to each of the a_i, we find that the expected Brier score is minimized for $\hat{S}(t|x) = S_T(t|x)$, i.e., when the estimated survival probabilities equal the true survival probabilities. *Under the assumption of censoring independent of the covariate and a perfectly estimated censoring distribution,* the Brier score is thus a proper score. However, as Equation 3 makes clear, the Brier score may not be a proper score when $C \not\perp\!\!\!\perp X$. The problematic assumption on the censoring distribution has, for example, been reported in Kvamme and Borgan (2019). We construct an example (Example 2) where an inaccurate distribution achieves a lower (i.e., better) Brier score than the true distribution in Table 1. For details see Appendix 1.3. Indeed, one could use estimates that include the covariate instead of the Kaplan–Meier estimate, so as to estimate $\mathbb{P}(C > t|x)$. Since this requires a regression of the censoring distribution to compute a score of the regression of the event-time, this raises the question of how to choose the best regression of the censoring distribution. Because of this loop, in practice the Kaplan–Meier estimate is often chosen (Kvamme and Borgan, 2019; Kvamme et al., 2019). The same analysis of the Brier score can be applied to the binomial log-likelihood (BLL). In particular, the BLL is a proper score under the assumption that $C \perp\!\!\!\perp X$, but it need not be proper when this assumption is violated. We use Example 2 to demonstrate that BLL also breaks in Table 1. It should be stressed that the pathology of Brier Score lies in the difficulty of estimating the weights rather than the score itself; work such as Han et al. (2021), propose inverse weighted survival games to correctly estimate the Brier Score.

Integrated Brier score and integrated binomial log-likelihood The Brier score above was defined for a fixed time t. The integrated Brier score (IBS) has been defined as a measure of goodness-of-fit

of the estimate $\hat{S}(\cdot|\cdot)$ by

$$\mathcal{S}_{\text{IBS}}\big(\hat{S}(t|x),(z,d)\big) = \frac{1}{t_2-t_1}\int_{t_1}^{t_2}\mathcal{S}^t_{\text{Brier}}\big(\hat{S}(t|x),(z,d)\big)dt,$$

where typically $t_1 = 0$ and $t_2 = z_{\max} = \max\{z_1,\dots,z_n\}$. In experiments, the integral is approximated by a sum over an equally spaced grid of times. The binomial log-likelihood at time t, $\mathcal{S}^t_{\text{BLL}}$ is defined by:

$$\mathcal{S}^t_{\text{BLL}}\big(S(t|x),(z,d)\big) = \frac{\log(1-\hat{S}(t|x))1\{z\le t, d=1\}}{\hat{G}(z)} + \frac{S(t|x)1\{z\ge t\}}{\hat{G}(t)}.$$

Analogously to the definition of IBS, the integrated binomial log-likelihood, IBLL, is defined as

$$\mathcal{S}_{\text{IBLL}}(S(t|x),(z,d)) = \frac{1}{t_2-t_1}\int_{t_1}^{t_2}\mathcal{S}^t_{\text{BLL}}(S(t|x),(z,d))dt.$$

As neither $\mathcal{S}^t_{\text{Brier}}$ nor S^t_{BLL} are proper scores, it follows that $S_{\text{Integrated-Brier}}$ and $\mathcal{S}_{\text{IBLL}}$ are not proper scores. We use Example 2 to show that IBS and IBLL breaks in Table 1.

Survival-CRPS The survival continuous ranked probability score (survival-CRPS) is defined by

$$\mathcal{S}_{\text{CRPS}}\big(\hat{F}_X,(Z,D)\big) = \int_0^Z \hat{F}(t|X)^2dt + D\int_Z^\infty (1-\hat{F}(t|X))^2dt$$

and was proposed in Avati et al. (2020) as an extension of the continuous ranked probability score (CRPS) (Gneiting et al., 2007). This is equivalent to the integrated Brier score without inverse probability weighting. In Avati et al. (2020), the score was also claimed to be a proper score. The proof strategy given there argues that the survival-CRPS is simply the weighted CRPS of Gneiting and Ranjan (2011) with the weight function set to the indicator of uncensored regions. As we show in Appendix 1.4.1, however, this weight function does not recover the survival-CRPS. This suggests that the survival-CRPS is not a proper score. For example, if the censoring time C is deterministic, i.e., $C = c$ w.p. 1 for $c \in \mathbb{R}_{\ge 0}$, then it is clear that to minimize the survival-CRPS, one would set $\hat{F}(c) = 1$, even if the true CDF is less than 1. This principle also holds in the case of random censoring, where one can construct false CDFs which yield a lower (i.e., better) survival-CRPS. We use this insight to construct an example (Example 3) where survival-CRPS breaks in Table 1. For details, see Appendix 1.4.2.

3.2 The right-censored log-likelihood is a proper score

The right-censored log-likelihood score of an observation (z,d), given a covariate x and distribution $S(t|x)$ is simply the likelihood given in Equation 1, i.e.:

$$\mathcal{S}_L(S(t|x),(z,d)) = d\log f(z|x) + (1-d)\log S(z|x).$$

Let $\hat{S}$ denote any continuous survival curve, and let S denote the true survival distribution of the event-time T, and make the following assumption.

Assumption 2. *Assume that $KL(S||\hat{S}) < \infty$, where KL denotes the Kullback–Leibler divergence.*

Under this assumption, we prove the following lemma.

Lemma 1. *Let S be the true survival distribution of the event-time T. For every distribution $\hat{S}$ that satisfies Assumption 2 and for every $x \in \mathbb{R}^p$, it holds that*

$$\mathbb{E}\left[\mathcal{S}_L(S(t|x),(Z,D))|X=x\right] \ge \mathbb{E}[\mathcal{S}_L(\hat{S}(t|x),(Z,D))|X=x]$$

Proof. We need to prove that for every $x \in \mathbb{R}^p$, for every distribution $S(t|x)$

$$\mathbb{E}\left[\mathcal{S}(S(t|x),(Z,D))|X=x\right] \ge \mathbb{E}[\mathcal{S}(\hat{S}(t|x),(Z,D))|X=x]$$

where $T \sim S(t|x)$, the censoring time C follows an arbitrary distribution and it is assumed that $S(t|x)$ and $\hat{S}(t|x)$ satisfy Assumption 1. We can safely omit x from the notation and prove that

$$\mathbb{E}\mathcal{S}(S,(Z,D)) \ge \mathbb{E}\mathcal{S}(\hat{S},(Z,D))$$

where again $T \sim S$, the censoring time C follows an arbitrary distribution and it is assumed that S and $\hat{S}$ satisfy Assumption 1. This has been proved in a slightly different context in Diks et al. (2011) Lemma 1. In that context the censoring variable is assumed known. The same argument can be applied to the case of random censoring, by first conditioning on the censoring variable. Namely, if we can prove that

$$\mathbb{E}\left[\mathcal{S}(S,(Z,D))|C\right] \ge \mathbb{E}[\mathcal{S}(\hat{S},(Z,D))|C]$$

then it follows, by taking the expectation over C, that also

$$\mathbb{E}\mathcal{S}(S,(Z,D)) \ge \mathbb{E}\mathcal{S}(\hat{S},(Z,D)).$$

We now prove the first inequality following the proof of Lemma 1 in Diks et al. (2011). Let f denote the density of S, and let $\hat{f}$ denote the density of $\hat{S}$. Note that the expected score can be written as follows:

$$\mathbb{E}\left[\mathcal{S}(\hat{S},(Z,D))|C\right] =$$

$$\mathbb{E}\left[D\log\hat{f}(Z)+(1-D)\log\hat{S}(Z)\middle|C\right]$$
$$=\int_0^\infty 1\{t\le C\}f(t)\log\hat{f}(t)+1\{t>C\}f(t)\log\hat{S}(C)dt$$
$$=\int_0^C f(t)\log\hat{f}(t)dt+S(C)\log\hat{S}(C).$$

Hence,

$$\mathbb{E}\big[\mathcal{S}(S,(Z,D))-\mathcal{S}(\hat{S},(Z,D))\ \big|\ C\big]$$
$$=\int_0^C f(t)\log\frac{f(t)}{\hat{f}(t)}dt+S(C)\log\frac{S(C)}{\hat{S}(C)}$$
$$=F(C)\int_0^C\frac{f(t)}{F(C)}\log\frac{f(t)/F(C)}{\hat{f}(t)/\hat{F}(C)}dt+S(C)\log\frac{S(C)}{\hat{S}(C)}-$$
$$F(C)\int_0^C\frac{f(t)}{F(C)}\log\frac{1/F(C)}{1/\hat{F}(C)}dt$$
$$=F(C)\int_0^C\frac{f(t)}{F(C)}\log\frac{f(t)/F(C)}{\hat{f}(t)/\hat{F}(C)}dt+$$
$$S(C)\log\frac{S(C)}{\hat{S}(C)}+F(C)\log\frac{F(C)}{\hat{F}(C)}$$
$$=F(C)\mathrm{KL}\big(f(t)/F(C)\,\|\,\hat{f}(t)/\hat{F}(C)\big)+$$
$$\mathrm{KL}\big(\mathrm{Ber}(S(C))\,\|\,\mathrm{Ber}(\hat{S}(C))\big)\ge 0.$$

The distributions $f(t)/F(C)$ and $\hat{f}(t)/\hat{F}(C)$ are the densities of f and $\hat{f}$ conditioned on the event $T\le C$. The first KL divergence exists by Assumption 1. The second term is the KL divergence between Bernoulli random variables with success probabilities $S(C)=1-F(C)$ and $\hat{S}(C)=1-\hat{F}(C)$ respectively, with the KL divergence between Bernoulli random variables given by $\mathrm{KL}(p\|q)=p\log\frac{p}{q}+(1-p)\log\frac{1-p}{1-q}$ □

Hence, under Assumption 2, the right-censored log-likelihood is a proper scoring rule. We provide a proof in Appendix 1.1, as a simple extension of Lemma 1 of Diks et al. (2011), by conditioning on the censoring variable and the covariate. We note that this result may also be deduced from Section 3.5 of Dawid and Musio (2014), as a non-trivial special case where one uses the function $\psi(p)=p\log p-p$, but the more general claim there is provided without a proof. We empirically validate that the right-censored log-likelihood is a proper score by applying it to the true and fake distributions for Example 1, 2 and 3 in Table 1. As expected, the right-censored log-likelihood correctly gives the true distribution a higher score than the fake distribution for all examples.

Summary of insights We have shown both theoretically and empirically that $\mathcal{S}_C^{\mathrm{td}}$, $\mathcal{S}_{\mathrm{IBS}}$, $\mathcal{S}_{\mathrm{IBLL}}$ and $\mathcal{S}_{\mathrm{CRPS}}$ are not proper scores, meaning that optimizing them gives no guarantee of learning the true survival distribution for right-censored data. Since the right-censored log-likelihood is proven to be a proper score, it is imperative to construct models that train and evaluate on the log-likelihood directly, rather than using non-proper proxy scores.

4 OPTIMIZING LIKELIHOOD USING PARTIALLY MONOTONIC NEURAL NETWORKS

To optimize the right-censored log-likelihood of (1), one needs to jointly model the survival distribution and the density. To do so, we adapt the monotonic neural density estimator (MONDE) of Chilinski and Silva (2020) in the case of a univariate response variable to the survival context. A similar strategy has been used in Omi et al. (2019) to model temporal point processes. We adapt MONDE to jointly model the survival function and the density function of a right-censored event-time, resulting in a scalable and flexible survival regression model that can train and evaluate directly on log-likelihood. The model is coined *SurvivalMonotonic-neural network* (SuMo-net) and we provide a brief overview of its architecture in Figure 2.

Architecture Figure 2 pictures the SuMo-network structure as a directed graph with two types of edges. The network can be defined in terms of two fully connected feed-forward sub-networks. A covariate x is first passed through a network $h^{\mathrm{cov}}(x)$ with L^{cov} layers in which each layer is fully connected to the next, and no other edges are present. The output of the covariate network, together with the time input t forms the input of a second network $h^{\mathrm{mixed}}(t,u)$, where u has the same dimension as the output of the covariate network $h^{\mathrm{cov}}(x)$. In this network, too, all edges are present from layer l to $l+1$ for $l=1,\dots,L^{\mathrm{mixed}}-1$ and no additional edges are added. The output layer of the mixed network consists of a single neuron with value $h(t,x)$, thus returning a scalar. The total network h thus equals

$$h(t,x)=h^{\mathrm{mixed}}(t,h^{\mathrm{cov}}(x)).$$

We then model $S(t|x)$ through $\hat{S}(t|x)=1-\mathrm{sigmoid}\,(h(t,x))$. It should be noted that we propose a minimal implementation of SuMo-net, only using a vanilla feed-forward architecture. Any other type of architecture such as Resnet (He et al., 2015), LSTM (Hochreiter and Schmidhuber, 1997), Densenet (Huang et al., 2018) etc., can be used to enhance model flexibility further, depending on the nature of the application.

Training After passing $h(t,x)$ through the sigmoid function, the final edge in Figure 2 represents differentiation of $\mathrm{sigmoid}(h(t,x))$ with respect to t and we set $\hat{f}(t|x)=-\partial\hat{S}(t|x)/\partial t$. To compute the derivative,

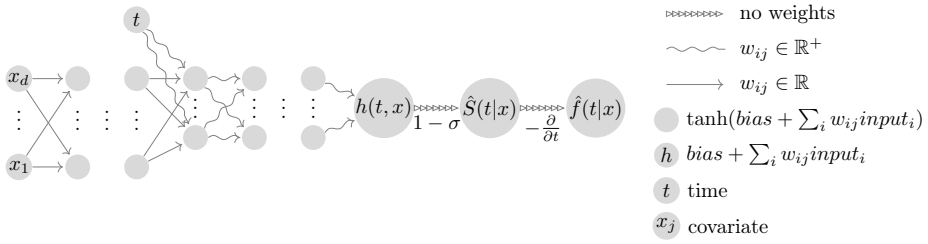


Figure 2: [Figure adapted from Figure 1 of Chilinski and Silva (2020)] The graph representing SuMo-net, modelling the survival curve. The last edge symbolizes the operation of differentiating the conditional survival distribution function $\hat{S}(t|x)$ with respect to the input t, resulting in the conditional density of the survival time. The second-to-last edge represents passing $h(t,x)$ through $1-\text{sigmoid}$. If one models the cumulative hazard instead, we replace $1-\text{sigmoid}$ by SoftReLU and use $\frac{\partial}{\partial t}$ in the last edge instead of $-\frac{\partial}{\partial t}$.

we use automatic differentiation as implemented in PyTorch (Paszke et al., 2019), in line with the approach taken in Chilinski and Silva (2020). One can also approximate the derivative by applying the chain rule to $\text{sigmoid}(h(t,x))$, yielding that

$$f(t|x) = -\text{sigmoid}(h(t,x))(1-\text{sigmoid}(h(t,x)) \times \left(\frac{h(t+\epsilon,x)-h(t,x)}{\epsilon}\right) + o(1).$$

We implemented SuMo-net both using this approximation and using auto-grad and found the implementations performed comparably during the experiments. We found further that both of these implementations improved performance over simply evaluating $-\left(S(t+\epsilon|x) - S(t|x)\right)/\epsilon$, which may be due to the vanishing gradient of the sigmoid function as one moves away from 0.

Universality Since the survival function and the cumulative hazard function are non-increasing and non-decreasing in t respectively, we can ensure that the output of the neural net does not decrease as t increases by restricting the weights which are applied to the time t to be non-negative. As discussed in Lang (2005), non-decreasing behaviour in t is guaranteed if all weights that are descendants of t in Figure 2 are non-negative. In Lang (2005) it is furthermore shown that if there are at least 2 hidden layers in the mixed network $h_{\text{mixed}}(t,u)$, then, given enough nodes in each layer, the network can universally approximate any true survival function. In our implementation, we ensure that weights are non-negative by defining them as the square of unrestricted weights. It should be noted that despite being a relatively simple idea, the neural network property of SuMo-net allows survival regression to straightforwardly use *any type of data* such as images, graphs or any type of sequences, which allows for an unrivaled flexibility in data choices.

5 EXPERIMENTS

We apply SuMo-net to five real datasets SUPPORT, METABRIC, GBSG, FLCHAIN, and KKBOX presented in Kvamme et al. (2019). We compare the results with the methods DeepHit (Lee et al., 2018), Cox-Time and Cox-CC (Kvamme et al., 2019), DeepSurv (Katzman et al., 2018) and the classical Cox model (Cox, 1972). To nuance the comparisons, we contrast against reported results in SODEN Tang et al. (2020) and SurvNODE Groha et al. (2021), which are both NeuralODE based models that can train on likelihood directly. We also compare against modelling the parameters (scale & shape and mean & variance) of the Weibull and Log-normal distribution with neural nets as functions of covariates and time, analogous to Avati et al. (2020). When parametrising a survival distribution directly, the right-censored likelihood can in fact be calculated directly.

We use a 5-fold cross-validation procedure in which the model is trained on three folds, one fold is used for validation, and one fold is reserved for testing. We then average the evaluation criteria calculated on five different test folds. For datasets FLCHAIN, GBSG, METABRIC, and SUPPORT we optimize SuMo-net and baselines by using the hyperopt library (Bergstra et al., 2013) for 300 iterations. For the KKBOX dataset we do the same, but for 100 hyperopt iterations, due to computational constraints. All experiments were run on a GPU cluster consisting of 8 Nvidia 1080 GTX TI cards.

Results In Table 3 we compare the methods in terms of right-censored likelihood. As the benchmarks do not have tractable right-censored likelihood, this is approximated by interpolating their survival curves. While the obtained likelihoods of the benchmarks are only approximations, the results suggest SuMo-net is performant in terms of right-censored likelihood even when compared to much more complex NeuralODE based methods. It should be noted that neither SurvNODE or SODEN has been run on datasets larger than 40000 observations, due to computational costs scaling with observations and function evaluations (Massaroli et al., 2021). In Table 4 we compare the binomial log-likelihood of the models. Also in terms of IBLL, SuMo-net scores well.

Dataset	SUPPORT	METABRIC	GBSG	FLCHAIN	KKBOX	Weibull	Checkerboard	Normal
Size	8,873	1,904	2,232	6,524	2,646,746	25,000	25,000	25,000
Covariates	14	9	7	8	15	1	1	1
Unique Durations	1,714	1,686	1,230	2,715	2,344,004	25000	25000	25000
Prop. Censored	0.32	0.42	0.43	0.70	0.28	0.55	0.75	0.50

Table 2: Summary of all datasets used in the experiments. Weibull, checkerboard and normal are synthetic datasets and the others are real datasets. See Kvamme et al. (2019) for more details on the real datasets.

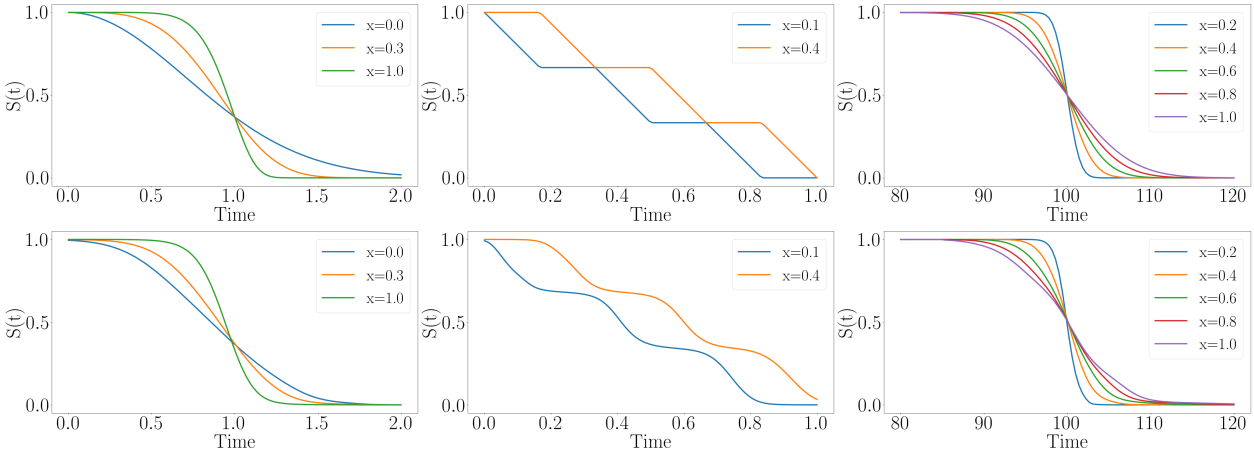


Figure 3: True distributions (top row) and learned distributions using SuMo-net (bottom row). From left to right, the distributions are Weibull, Checkboard, and Normal.

Results for IBS and time-dependent concordance are given in the Appendix. Notably, we find that DeepHit, which optimizes a concordance-like loss function, generally has the best performance in terms of time-dependent concordance, yet achieves the worst scores in terms of likelihood. This agrees with our findings that time-dependent concordance is not a proper scoring rule and that optimizing time-dependent concordance may lead to inaccurate distributions.

Correlation between likelihood and existing metrics Following our result in Section 3 that likelihood is the only proper score among those considered, we investigate how the other scores compare to likelihood by calculating the rank correlation between all scores. In particular, for each of the datasets FLCHAIN, GBSG, METABRIC, SUPPORT and KKBOX and each hyperparameter run of SuMo-net, we compute all scores on the test set and turn them into ranks. The correlation matrix averaged over the datasets is displayed in Table 5. The experiment indicates that concordance is almost independent of likelihood, while IBLL exhibits the strongest correlation, suggesting that it could serve as a good proxy when the likelihood is not available.

Estimation of the likelihood for models other than SuMo-net All the baseline models proposed in Kvamme et al. (2019), learn discrete distributions and hence have no density and the likelihood cannot be computed. For those methods we use interpolation of the survival curve to approximate the density. Let the survival curve of the discrete distribution be denoted by S and denote the points at which the survival curve makes jumps by $t_0 \leq \cdots \leq t_m$. Then, given a time t, we find i so that $t_i \leq z < t_{i+1}$ and then estimate

$$\hat{f}(t) = -\frac{S(t_{i+k}) - S(t_{i-k+1})}{t_{i+k} - t_{i-k+1}}, \tag{5}$$

where k determines the width of the interval we use to approximate the derivative. We found that setting $k = 1$ resulted in a highly variable density and low likelihood, and hence we use $k = 2$ in the main text. SuMo-net has tractable likelihood, so this approximation does not have to be made. While comparing models with an approximated and exact likelihood score is not ideal, it illustrates a general issue with modern survival models: they are not endowed with a tractable likelihood, making it difficult to evaluate them on a proper score at test time. As likelihood is an easy proper score to calculate for survival models, we stress that future models should be developed with tractable likelihood evaluation procedure in mind.

Calibration When training survival models on likelihood directly we have direct access to the density $f(t|x)$ *and* to the cumulative distribution $F(t|x)$, which means we can quantify how calibrated the fitted survival distribution is. Following Kuleshov et al. (2018) and Goldstein et al. (2021), a calibrated survival distribution would have an empirical frequency

	$S_L \uparrow$								
	CoxCC	Cox-Linear	Cox-Time	DeepHit	DeepSurv	SODEN	Weibull	Log-normal	SuMo-net
FLCHAIN	-0.507 ± 0.014	-0.513 ± 0.014	-0.584 ± 0.239	-0.85 ± 0.493	-0.529 ± 0.015	N/A	-0.467 ± 0.085	-0.535 ± 0.251	$\mathbf{-0.376} \pm 0.008$
GBSG	-0.501 ± 0.038	-0.498 ± 0.04	-0.454 ± 0.034	-0.719 ± 0.19	-0.502 ± 0.04	N/A	-0.577 ± 0.117	-0.563 ± 0.115	$\mathbf{-0.391} \pm 0.027$
KKBOX	0.797 ± 0.013	0.706 ± 0.01	1.411 ± 0.014	-4.174 ± 4.128	0.758 ± 0.036	N/A	-0.118 ± 0.0	0.201 ± 0.289	$\mathbf{1.553} \pm 0.132$
METABRIC	-0.305 ± 0.037	-0.316 ± 0.031	-0.233 ± 0.066	-0.565 ± 0.348	-0.318 ± 0.029	-0.149 ± 0.015	-0.29 ± 0.042	-0.299 ± 0.051	$\mathbf{-0.142} \pm 0.042$
SUPPORT	0.533 ± 0.047	0.545 ± 0.03	0.57 ± 0.038	0.098 ± 0.141	0.549 ± 0.045	-0.676 ± 0.009	0.035 ± 0.024	0.527 ± 0.098	$\mathbf{0.658} \pm 0.027$

Table 3: Log-likelihood scores and standard deviations over the 5 folds. Higher is better. SuMo-net, Weibull, Log-normal and SODEN provides exact likelihood. For other methods we have approximated the likelihood. In particular, we approximate the density by interpolating the survival curve. See Equation (5) for details.

	$\mathcal{S}_{\text{IBLL}} \uparrow$									
	CoxCC	Cox-Linear	Cox-Time	DeepHit	DeepSurv	SurvNODE	SODEN	Weibull	Log-normal	SuMo-net
FLCHAIN	-0.464 ± 0.194	-0.336 ± 0.012	-0.47 ± 0.241	-0.646 ± 0.462	-0.339 ± 0.01	N/A	N/A	-0.389 ± 0.045	-0.504 ± 0.209	$\mathbf{-0.332} \pm 0.006$
GBSG	-0.546 ± 0.02	-0.549 ± 0.011	-0.541 ± 0.021	-1.157 ± 1.287	-0.539 ± 0.013	N/A	N/A	-1.513 ± 1.525	-1.093 ± 0.633	$\mathbf{-0.528} \pm 0.012$
KKBOX	-0.382 ± 0.005	-0.429 ± 0.001	$\mathbf{-0.33} \pm 0.004$	-0.745 ± 0.513	-0.368 ± 0.002	N/A	N/A	-3.032 ± 1.766	-1.776 ± 2.187	-0.395 ± 0.029
METABRIC	-0.53 ± 0.035	-0.522 ± 0.012	-0.526 ± 0.012	-0.656 ± 0.156	-0.52 ± 0.013	$\mathbf{-0.477}$	-0.484 ± 0.010	-0.554 ± 0.004	-0.597 ± 0.064	-0.485 ± 0.015
SUPPORT	-0.572 ± 0.008	-0.577 ± 0.005	-0.577 ± 0.009	-0.701 ± 0.065	-0.569 ± 0.01	-0.580	$\mathbf{-0.561} \pm 0.002$	-0.705 ± 0.013	-0.654 ± 0.053	-0.569 ± 0.004

Table 4: IBLL scores and standard deviations over the 5 folds. Higher is better

	$\mathcal{S}_L$	$\mathcal{S}_{C^{\text{td}}}$	$\mathcal{S}_{\text{IBS}}$	$\mathcal{S}_{\text{IBLL}}$
$\mathcal{S}_L$	1.000	0.174	0.688	0.851
$\mathcal{S}_{C^{\text{td}}}$	0.174	1.000	0.223	0.179
$\mathcal{S}_{\text{IBS}}$	0.688	0.223	1.000	0.841
$\mathcal{S}_{\text{IBLL}}$	0.851	0.179	0.841	1.000

Table 5: Rank correlation matrix between likelihood, concordance, IBS and IBLL. The correlation matrix is averaged over all datasets.

	Calibration score		
	Weibull	Log-normal	SuMo-net
FLCHAIN	1.846 ± 0.641	$\mathbf{0.934} \pm 0.579$	1.165 ± 0.116
GBSG	0.762 ± 0.442	0.635 ± 0.245	$\mathbf{0.128} \pm 0.03$
KKBOX	0.897 ± 0.401	0.596 ± 0.443	$\mathbf{0.112} \pm 0.077$
METABRIC	0.327 ± 0.177	0.561 ± 0.517	$\mathbf{0.097} \pm 0.056$
SUPPORT	0.342 ± 0.276	$\mathbf{0.237} \pm 0.139$	0.313 ± 0.049

Table 6: SuMo-net generally provides a calibrated survival distribution

$\hat{p}_j = \frac{|\{z_i | F(z_i|x_i) \leq p_j, i=1,\ldots,n\}|}{n}$ equal to p_j. The calibration error is then defined as $\text{cal}\,(F_1, y_1, \ldots, F_T, y_T) = \sum_{j=1}^{m} w_j \cdot (p_j - \hat{p}_j)^2$, where we take $w_j = 1, \forall j$ and $p_j \in \{0.1, 0.2, 0.3, 0.4, 0.5, 0.6, 0.7, 0.8, 0.9\}$. As the empirical frequency can only be coarsely approximated without direct access to $F(t|x)$, we only evaluate calibration for methods that train directly on likelihood.

Inference and training timings Here we contrast the timing of predictions of the survival probability $S(t|x)$ using a trained model. We compare the time of SuMo-net against the time of Cox-Time presented in Kvamme et al. (2019) and SODEN (Tang et al., 2020). To compute the survival probability, Cox-Time integrates the learned hazard function over time, which is computationally costly. In contrast, SuMo-net evaluates S through a single forward pass. To compare the timings we initialize SuMo-net, Cox-Time and SODEN using an identical architecture and compare inference timings on the test-fold of all datasets in Table 7. The findings indicate SuMo-net does indeed achieve its anticipated speedup.

	SODEN (s)	Cox-Time (s)	SuMo-net (s)	Speedup vs	
				SODEN	Cox-Time
SUPPORT	1.124 ± 0.070	2.97 ± 0.126	0.047 ± 0.004	24	63
METABRIC	0.459 ± 0.044	1.573 ± 0.126	0.016 ± 0.001	29	100
GBSG	N/A	1.46 ± 0.112	0.021 ± 0.002	N/A	69
FLCHAIN	N/A	4.199 ± 0.246	0.044 ± 0.003	N/A	96
KKBOX	N/A	140.505 ± 7.976	7.034 ± 1.092	N/A	20

Table 7: Inference timings on 20% of the data. SuMo-net achieves a speedup factor of between 20 and 100.

6 CONCLUSION

We have shown that several existing scoring rules for right-censored regression are not proper, i.e., they may not faithfully reflect the accuracy of a fitted distribution. We also proved that the right-censored log-likelihood is a proper scoring rule. While other scoring rules can still be useful for certain purposes, their limitations should be taken into consideration. Since the right-censored log-likelihood is thus attractive both as a scoring rule and to assess model fit, we proposed SuMo-net, a simple, scalable and flexible regression model for survival data, for which the right-censored log-likelihood is tractable by construction. Future fruitful directions would be to extend SuMo-net to more involved architectures and datatypes such as genetic data, image data and other high dimensional data.

Downstream applications While SuMo-net considers a monotonic neural network, the idea of enforcing monotonicity on the time-to-event variable t can be extended to any model with trainable weights to train directly on any type of censored survival likelihood. This allows fields such as epidemiology, medical statistics, and economics, among many others, to formulate highly flexible non-parametric models that can evaluate directly on a proper scoring rule instead of non-proper proxy objectives. This is particularly important in safety critical domains where accurately approximating the true underlying model impacts subsequent decision making. We hope that that the introduction of SuMo-net can inspire future developments of sophisticated survival models endowed with a tractable likelihood for proper evaluation and comparison between models.

ACKNOWLEDGMENTS

We thank the reviewers for their helpful remarks. We are grateful for our colleagues Jean-Francois Ton, Rob Zinkov, Siu Lun Chau and Zoi Tsangalidou for their helpful comments and remarks.

References

Laura Antolini, Patrizia Boracchi, and Elia Biganzoli. A time-dependent discrimination index for survival data. *Statistics in medicine*, 24(24):3927–3944, 2005.

Anand Avati, Tony Duan, Sharon Zhou, Kenneth Jung, Nigam H Shah, and Andrew Y Ng. Countdown regression: sharp and calibrated survival predictions. In *Uncertainty in Artificial Intelligence*, pages 145–155. PMLR, 2020.

J. Bergstra, D. Yamins, and D. D. Cox. Making a science of model search: Hyperparameter optimization in hundreds of dimensions for vision architectures. In *Proceedings of the 30th International Conference on International Conference on Machine Learning - Volume 28*, ICML'13, page I–115–I–123. JMLR.org, 2013.

Paul Blanche, Michael W Kattan, and Thomas A Gerds. The c-index is not proper for the evaluation of t
-year predicted risks. *Biostatistics*, 20(2):347–357, 02 2018. ISSN 1465-4644. doi: 10.1093/biostatistics/kxy006. URL `https://doi.org/10.1093/biostatistics/kxy006`.

Paidamoyo Chapfuwa, Chenyang Tao, Chunyuan Li, Courtney Page, Benjamin Goldstein, Lawrence Carin Duke, and Ricardo Henao. Adversarial time-to-event modeling. In *International Conference on Machine Learning*, pages 735–744. PMLR, 2018.

Pawel Chilinski and Ricardo Silva. Neural likelihoods via cumulative distribution functions. In *Conference on Uncertainty in Artificial Intelligence*, pages 420–429. PMLR, 2020.

David R Cox. Regression models and life-tables. *Journal of the Royal Statistical Society: Series B (Methodological)*, 34(2):187–202, 1972.

Alexander Philip Dawid and Monica Musio. Theory and applications of proper scoring rules. *Metron*, 72 (2):169–183, 2014.

Cees Diks, Valentyn Panchenko, and Dick Van Dijk. Likelihood-based scoring rules for comparing density forecasts in tails. *Journal of Econometrics*, 163(2): 215–230, 2011.

Lore Dirick, Gerda Claeskens, and Bart Baesens. Time to default in credit scoring using survival analysis: a benchmark study. *Journal of the Operational Research Society*, 68(6):652–665, 2017.

Emilien Dupont, Arnaud Doucet, and Yee Whye Teh. Augmented neural ODEs. In *Advances in Neural Information Processing Systems*, volume 32, 2019.

Tilmann Gneiting and Adrian E Raftery. Strictly proper scoring rules, prediction, and estimation. *Journal of the American Statistical Association*, 102(477): 359–378, 2007.

Tilmann Gneiting and Roopesh Ranjan. Comparing density forecasts using threshold-and quantile-weighted scoring rules. *Journal of Business & Economic Statistics*, 29(3):411–422, 2011.

Tilmann Gneiting, Fadoua Balabdaoui, and Adrian E Raftery. Probabilistic forecasts, calibration and sharpness. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, 69(2):243–268, 2007.

Mark Goldstein, Xintian Han, Aahlad Puli, Adler J. Perotte, and Rajesh Ranganath. X-cal: Explicit calibration for survival analysis, 2021.

Erika Graf, Claudia Schmoor, Willi Sauerbrei, and Martin Schumacher. Assessment and comparison of prognostic classification schemes for survival data. *Statistics in Medicine*, 18(17-18):2529–2545, 1999.

Stefan Groha, Sebastian M Schmon, and Alexander Gusev. A general framework for survival analysis and multi-state modelling, 2021.

Chong Gu. *Smoothing spline ANOVA models*, volume 297. Springer Science & Business Media, 2013.

Xintian Han, Mark Goldstein, Aahlad Puli, Thomas Wies, Adler J Perotte, and Rajesh Ranganath. Inverse-weighted survival games, 2021.

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition, 2015.

Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural computation*, 9:1735–80, 12 1997. doi: 10.1162/neco.1997.9.8.1735.

Gao Huang, Zhuang Liu, Laurens van der Maaten, and Kilian Q. Weinberger. Densely connected convolutional networks, 2018.

Hemant Ishwaran, Udaya B Kogalur, Eugene H Blackstone, Michael S Lauer, et al. Random survival forests. *The Annals of Applied Statistics*, 2(3):841–860, 2008.

Jared L Katzman, Uri Shaham, Alexander Cloninger, Jonathan Bates, Tingting Jiang, and Yuval Kluger. Deepsurv: personalized treatment recommender system using a Cox proportional hazards deep neural network. *BMC Medical Research Methodology*, 18(1): 24, 2018.

David G. Kleinbaum and Mitchel Klein. *Survival Analysis: A Self-Learning Text*. Springer Science and Business Media, LLC, New, 2005.

Volodymyr Kuleshov, Nathan Fenner, and Stefano Ermon. Accurate uncertainties for deep learning using calibrated regression, 2018.

Håvard Kvamme and Ørnulf Borgan. The brier score under administrative censoring: Problems and solutions. *arXiv preprint arXiv:1912.08581*, 2019.

Håvard Kvamme, Ørnulf Borgan, and Ida Scheel. Time-to-event prediction with neural networks and Cox regression. *Journal of machine learning research*, 20(129):1–30, 2019.

Bernhard Lang. Monotonic multi-layer perceptron networks as universal approximators. In *International conference on artificial neural networks*, pages 31–37. Springer, 2005.

John A Laurie, Charles G Moertel, Thomas R Fleming, Harry S Wieand, John E Leigh, Jebal Rubin, Greg W McCormack, James B Gerstner, James E Krook, and James Malliard. Surgical adjuvant therapy of large-bowel carcinoma: an evaluation of levamisole and the combination of levamisole and fluorouracil. the north central cancer treatment group and the Mayo Clinic. *Journal of Clinical Oncology*, 7 (10):1447–1456, 1989.

Changhee Lee, William R Zame, Jinsung Yoon, and Mihaela van der Schaar. Deephit: A deep learning approach to survival analysis with competing risks. In *AAAI*, pages 2314–2321, 2018.

Gilbert MacKenzie. The statistical analysis of failure time data. *The Statistician*, 31:278–278, 1982.

Stefano Massaroli, Michael Poli, Jinkyoo Park, Atsushi Yamashita, and Hajime Asama. Dissecting neural odes, 2021.

James E Matheson and Robert L Winkler. Scoring rules for continuous probability distributions. *Management science*, 22(10):1087–1096, 1976.

Xenia Miscouridou, Adler Perotte, Noémie Elhadad, and Rajesh Ranganath. Deep survival analysis: Nonparametrics and missingness. In *Machine Learning for Healthcare Conference*, pages 244–256. PMLR, 2018.

Takahiro Omi, Naonori Ueda, and Kazuyuki Aihara. *Fully Neural Network Based Model for General Temporal Point Processes*. Curran Associates Inc., Red Hook, NY, USA, 2019.

Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, Alban Desmaison, Andreas Kopf, Edward Yang, Zachary DeVito, Martin Raison, Alykhan Tejani, Sasank Chilamkurthy, Benoit Steiner, Lu Fang, Junjie Bai, and Soumith Chintala. PyTorch: An imperative style, high-performance deep learning library. In H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox, and R. Garnett, editors, *Advances in Neural Information Processing Systems 32*, pages 8024–8035. Curran Associates, Inc., 2019.

Rajesh Ranganath, Adler Perotte, Noémie Elhadad, and David Blei. Deep survival analysis. In *Machine Learning for Healthcare Conference*, pages 101–114. PMLR, 2016.

Dimitris Rizopoulos, Laura A Hatfield, Bradley P Carlin, and Johanna JM Takkenberg. Combining dynamic predictions from joint models for longitudinal and time-to-event data using bayesian model averaging. *Journal of the American Statistical Association*, 109(508):1385–1397, 2014.

Gian A. Susto, Andrea Schirru, Simone Pampuri, Seán McLoone, and Alessandro Beghi. Machine learning for predictive maintenance: A multiple classifier approach. *IEEE Transactions on Industrial Informatics*, 11(3): 812–820, 2015.

Weijing Tang, Jiaqi Ma, Qiaozhu Mei, and Ji Zhu. Soden: A scalable continuous-time survival model through ordinary differential equation networks, 2020.

Dirk Van den Poel and Bart Lariviere. Customer attrition analysis for financial services using proportional hazard models. *European Journal of Operational Research*, 157(1):196–217, 2004.

Matthias Ziehm and Janet Thornton. Unlocking the potential of survival data for model organisms through a new database and online analysis platform: Survcurv. *Aging cell*, 12, 07 2013. doi: 10.1111/acel.12121.

Supplementary Material: Survival Regression with Proper Scoring Rules and Monotonic Neural Networks

A Appendix

A.1 Details on time-dependent concordance example

Let $X \in \{0, 1\}$. Let $K \sim \text{Uniform}\{0, 2, 4, 6, 8\}$ and let $U \sim \text{Uniform}[0, 1]$. Then $T|X = 0 \sim K + U$ and $T|X = 1 \sim K + U + 1$. We further set $C \sim \text{Exponential}(\text{mean} = 20)$. Note that the hazard rates of the two distributions are nonzero on disjoint sets. The False distribution in Figure 1, is constructed such that at each time, the distribution with the nonzero hazard rate has the lowest survival probability.

A.2 Details on Brier score example

Let $X \sim \text{Bernoulli}(0.1)$. We create an extreme example of unequal censoring in which $C \sim \text{Exp}(1)$ if $X = 0$ and $C = \infty$ if $X = 1$. Let $T \sim \text{Exponential}(10)$ (independent of X), so that $F_{\text{True}}(t) = 1 - \exp(-t/10)$. Finally create a fake distribution $\hat{F}$ so that $\hat{F}(t|X = 1) = F_{\text{true}}(t)$ and $\hat{F}(t|X = 0) = 1 - \exp(-t/4)$. We take a sample of size $n = 10^6$. We evaulate the Brier score at $t = 4$. We evaluate the Integrated Brier score from $t = 0$ to $t = 40$. We repeat the procedure for BLL and IBLL.

A.3 Discussion of survival-CRPS

A.3.1 Discussion of proof given in Avati et al

In Gneiting and Ranjan (2011) it is stated that the following score is proper (see Matheson and Winkler (1976) for a proof),

$$\begin{aligned}\mathcal{S}(F, z) &= \int_0^\infty (F(r) - 1\{r \geq z\})^2 u(r) dr \\ &= \int_0^z F(r)^2 u(r) dz + \int_z^\infty (1 - F(r))^2 u(r) dr.\end{aligned}$$

for non-negative weight functions u. In Avati et al. (2020) it is argued that the survival-CRPS is simply the above score, with weight function equal to the uncensored region. It is not immediately clear how this is meant to be interpreted, as the weight function needs to be a non-random function. If it were a proper score when conditioned on a known censoring time $C = c$, for every value of c, then we could proceed as in Section 3.2, using the conditional independence of C and T. Assuming $C = c$, the uncensored region equals $u(r) = 1\{r \leq c\}$. Proceeding with this as the weight function we would obtain

$$\begin{aligned}S(F, z) &= \int_0^\infty (F(r) - 1\{r \geq z\})^2 1\{r \leq c\} dr \\ &= \int_0^z F(r)^2 1\{r \leq c\} dz + \int_z^\infty (1 - F(r))^2 1\{r \leq c\} dr.\end{aligned}$$

Note that $z \leq c$ since $z = \min(t, c)$ and therefore the first indicator is always true and can be removed, to arrive at:

$$S(F,z) = \int_0^z F(r)^2 dr + \int_z^\infty (1-F(r))^2 1\{r \le c\} dr$$
$$= \int_0^z F(r)^2 dr + \int_z^c (1-F(r))^2 dr.$$

Hence, the survival-CRPS has an additional term

$$d \int_c^\infty (1-F(r))^2 dr \tag{6}$$

which is missing from the above expression. This shows, at least, that this interpretation of the proof strategy would not succeed. Furthermore, assuming that c is a fixed censoring time, then $F(r)$ is only evaluated for $r > c$ in Equation 6. Clearly to minimize the survival-CRPS, one then sets $F(r) = 1$ for $r \ge c$, even if the true distribution satisfies $F_{\text{true}}(r) < 1$ for $r \ge c$.

A.3.2 An example

Assume for simplicity that there are no covariates. Let $T \sim \text{Exp}(\text{mean} = 100)$ with every individual set to fail at $t = 200$ so that $F_{True}(t) = 1 - \exp(-t/100)$ for $t < 200$ and $F_{\text{true}}(200) = 1$. Let $C \sim \text{Exp}(\text{mean} = 10)$. Observations are thus likely to be censored. Let a fake distribution be $F_{\text{False}} = \text{Exp}(\text{mean} = 25)$. We then take a sample of size 10^6 from the fake distribution and true distribution and evaluate and compute their survival-CRPS against the true distribution.

A.4 Calculating life expectancies using SuMo-net

The life expectancies are calculated by direct numerical integration using the learned survival distribution $S_\theta(t|x)$ since $\mathbb{E}[T|\,x] = \int_0^\infty S_\theta(t|x)dt$.

A.5 Log-likelihood scores for $d = 1, 4, 8$

dataset Method	$\mathcal{S}_L \uparrow$ FLCHAIN	GBSG	KKBOX	METABRIC	SUPPORT
CoxCC	-0.583 ± 0.014	-0.632 ± 0.032	0.156 ± 0.075	-0.451 ± 0.06	0.437 ± 0.06
Cox-Linear	-0.586 ± 0.026	-0.603 ± 0.032	0.05 ± 0.111	-0.444 ± 0.03	0.451 ± 0.053
Cox-Time	-0.565 ± 0.071	-0.569 ± 0.063	0.926 ± 0.09	-0.346 ± 0.056	0.481 ± 0.047
DeepHit	-0.878 ± 0.496	-0.805 ± 0.212	-1.432 ± 0.226	-0.686 ± 0.5	0.04 ± 0.159
DeepSurv	-0.605 ± 0.01	-0.609 ± 0.04	0.179 ± 0.073	-0.455 ± 0.027	0.446 ± 0.036
SuMo-net	$\mathbf{-0.376} \pm 0.008$	$\mathbf{-0.391} \pm 0.027$	$\mathbf{1.553} \pm 0.132$	$\mathbf{-0.142} \pm 0.042$	$\mathbf{0.658} \pm 0.027$

Table 8: Log-likelihood scores for $d = 1$

Dataset Method	$\mathcal{S}_L \uparrow$ FLCHAIN	GBSG	KKBOX	METABRIC	SUPPORT
CoxCC	-0.482 ± 0.01	-0.451 ± 0.033	0.181 ± 0.088	-0.286 ± 0.054	0.564 ± 0.04
Cox-Linear	-0.49 ± 0.017	-0.465 ± 0.043	0.095 ± 0.113	-0.265 ± 0.02	0.565 ± 0.032
Cox-Time	-0.496 ± 0.166	-0.409 ± 0.043	0.93 ± 0.089	-0.189 ± 0.061	0.6 ± 0.041
DeepHit	-0.846 ± 0.497	-0.94 ± 0.415	-4.454 ± 4.441	-0.56 ± 0.338	0.113 ± 0.126
DeepSurv	-0.498 ± 0.015	-0.457 ± 0.021	0.158 ± 0.117	-0.282 ± 0.027	0.562 ± 0.029
SuMo-net	$\mathbf{-0.376} \pm 0.008$	$\mathbf{-0.391} \pm 0.027$	$\mathbf{1.553} \pm 0.132$	$\mathbf{-0.142} \pm 0.042$	$\mathbf{0.658} \pm 0.027$

Table 9: Log-likelihood scores for $d = 4$

Dataset Method	$\mathcal{S}_L \uparrow$ FLCHAIN	GBSG	KKBOX	METABRIC	SUPPORT
CoxCC	-0.485 ± 0.014	-0.451 ± 0.033	0.42 ± 0.021	-0.243 ± 0.023	0.568 ± 0.036
Cox-Linear	-0.477 ± 0.012	-0.442 ± 0.033	0.344 ± 0.037	-0.253 ± 0.012	0.572 ± 0.037
Cox-Time	-0.55 ± 0.178	-0.398 ± 0.036	0.985 ± 0.037	-0.179 ± 0.064	0.599 ± 0.043
DeepHit	-0.835 ± 0.491	-0.637 ± 0.138	-3.622 ± 3.38	-0.552 ± 0.299	0.086 ± 0.134
DeepSurv	-0.495 ± 0.01	-0.435 ± 0.025	0.414 ± 0.04	-0.249 ± 0.018	0.573 ± 0.042
SuMo-net	$\mathbf{-0.376} \pm 0.008$	$\mathbf{-0.391} \pm 0.027$	$\mathbf{1.553} \pm 0.132$	$\mathbf{-0.142} \pm 0.042$	$\mathbf{0.658} \pm 0.027$

Table 10: Log-likelihood scores for $d = 8$

A.6 Other experiments

Dataset Method	$\mathcal{S}_{C^{\text{td}}} \uparrow$ FLCHAIN	GBSG	KKBOX	METABRIC	SUPPORT
CoxCC	0.792 ± 0.005	0.672 ± 0.014	0.832 ± 0.001	0.642 ± 0.03	0.608 ± 0.011
Cox-Linear	0.792 ± 0.006	0.66 ± 0.014	0.795 ± 0.001	0.633 ± 0.022	0.596 ± 0.012
Cox-Time	0.792 ± 0.005	0.678 ± 0.017	$\mathbf{0.848} \pm 0.008$	0.649 ± 0.026	0.613 ± 0.007
DeepHit	$\mathbf{0.796} \pm 0.004$	$\mathbf{0.683} \pm 0.013$	0.799 ± 0.005	$\mathbf{0.692} \pm 0.022$	$\mathbf{0.642} \pm 0.003$
DeepSurv	0.791 ± 0.004	0.671 ± 0.014	0.828 ± 0.001	0.645 ± 0.02	0.609 ± 0.011
SuMo-net	0.79 ± 0.004	0.671 ± 0.013	0.76 ± 0.045	0.658 ± 0.029	0.603 ± 0.004
SODEN	N/A	N/A	N/A	0.661 ± 0.004	0.624 ± 0.004
SurvNODE	N/A	N/A	N/A	0.667	0.622

Table 11: Concordance scores. Higher is better

Dataset Method	$\mathcal{S}_{\text{IBS}} \downarrow$ FLCHAIN	GBSG	KKBOX	METABRIC	SUPPORT
CoxCC	0.112 ± 0.014	0.185 ± 0.006	0.118 ± 0.001	0.179 ± 0.013	$\mathbf{0.194} \pm 0.003$
Cox-Linear	0.102 ± 0.004	0.188 ± 0.004	0.138 ± 0.0	0.176 ± 0.005	0.197 ± 0.002
Cox-Time	0.114 ± 0.017	0.183 ± 0.008	$\mathbf{0.104} \pm 0.001$	0.178 ± 0.005	0.196 ± 0.004
DeepHit	0.162 ± 0.079	0.227 ± 0.064	0.16 ± 0.012	0.21 ± 0.037	0.227 ± 0.019
DeepSurv	0.103 ± 0.004	0.183 ± 0.005	0.116 ± 0.001	0.176 ± 0.005	0.194 ± 0.004
SuMo-net	$\mathbf{0.101} \pm 0.004$	$\mathbf{0.179} \pm 0.005$	0.13 ± 0.023	0.163 ± 0.006	0.196 ± 0.003
SODEN	N/A	N/A	N/A	0.162 ± 0.003	$\mathbf{0.190} \pm 0.001$
SurvNODE	N/A	N/A	N/A	**0.157**	0.198

Table 12: IBS scores. Lower is better

A.7 Implementation Details

We present the implementation details to SuMo-net, including network architecture parameters, optimizer parameters and the code base.

	Inference Times						Training times (1 epoch)					
	CoxCC	Cox linear	Cox-Time	Deephit	DeepSurv	SuMo-net	CoxCC	Cox linear	Cox-Time	Deephit	DeepSurv	SuMo-net
SUPPORT	0.027 ± 0.003	0.026 ± 0.003	3.769 ± 0.056	0.003 ± 0.0	0.023 ± 0.001	0.044 ± 0.0	0.664 ± 0.068	4.565 ± 6.892	7.411 ± 1.273	0.347 ± 0.136	0.525 ± 0.027	3.236 ± 0.026
METABRIC	0.008 ± 0.001	0.008 ± 0.0	1.586 ± 0.003	0.001 ± 0.0	0.008 ± 0.0	0.016 ± 0.0	0.156 ± 0.01	0.153 ± 0.023	2.676 ± 0.02	0.136 ± 0.002	0.147 ± 0.002	0.337 ± 0.01
GBSG	0.008 ± 0.0	0.008 ± 0.0	1.737 ± 0.029	0.002 ± 0.0	0.009 ± 0.001	0.021 ± 0.0	0.194 ± 0.069	0.161 ± 0.028	2.829 ± 0.053	0.147 ± 0.003	0.159 ± 0.013	0.395 ± 0.01
FLCHAIN	0.021 ± 0.0	0.021 ± 0.001	4.794 ± 0.007	0.003 ± 0.0	0.021 ± 0.0	0.041 ± 0.001	0.403 ± 0.027	0.394 ± 0.028	7.997 ± 0.019	0.208 ± 0.002	0.386 ± 0.014	1.605 ± 0.062
KKBOX	1.836 ± 0.021	1.804 ± 0.002	158.141 ± 0.121	0.351 ± 0.048	1.814 ± 0.021	5.806 ± 0.009	990.645 ± 0.817	990.144 ± 1.737	1156.948 ± 0.338	872.381 ± 1.083	987.592 ± 1.365	700.269 ± 0.517

Table 13: Additional experiments detailining run-times: SuMo-net is the fastest in terms of training speed per epoch on larger datasets.

A.7.1 Hyperparameter space

Hyperparameter	Values
Layers	{1, 2,4,5}
Layers (Covariate part)	{1, 2,4,5}
Nodes per layer	{8,16,32}
Nodes per layer (Covariate part)	{8,16,32}
Dropout	[0.0,0.1,0.2,0.3,0.4,0.5]
Weigh decay	{0.4, 0.2, 0.1, 0.05, 0.02, 0.01, 0}
Batch size	{5,10,25,50,100,250}

Table 14: Hyperparameter search space for experiments on Rot. & GBSG, SUPPORT, METABRIC, and FLCHAIN.

Hyperparameter	Values
Layers	{1, 2,4}
Layers (Covariate part)	{4, 6, 8}
Nodes per layer	{8, 16, 32,64}
Nodes per layer (Covariate part)	{128, 256, 512}
Dropout	[0, 0.7]
Weigh decay	{0.4, 0.2, 0.1, 0.05, 0.02, 0.01, 0}
Batch size	{1000,2500,5000}

Table 15: KKBox hyperparameter configurations. (*) denotes parameters found with a two layer network with 128 nodes.

For exact details, we refer to the code base attached.

A.7.2 Toy datasets

We provide here the distributions of the toy datasets. In each of the distributions $X \sim \text{Unif}[0, 1]$.

Dataset	$T\|X$	$C\|X$
Weibull	Weib(shape $= 2 + 6X$)	Exp(mean $= 1.5$)
Normal	$\text{N}(100, 6X)$	$\text{N}(100, 6)$
Checkerboard	$\text{CB}(4, 6\|X)$	Exp(mean $= 1.5$)

Table 16: Distributions of the toy datasets Weibull, Normal and Checkerboard. Weib(shape $= s$) refers to the Weibull distribution with shape s and scale 1. For the definition of $\text{CB}(4, 6)$ see below. Parameters are chosen such that appropriate numbers of individuals is censored (see data description main text) and the survival curves of the Normal and Weibull distributions cross (see plots main text).

The distribution $\text{CB}(4, 6|X)$ is defined as follows. Let X be on the horizontal axis and let T be on the vertical axis. We split the square $[0, 1]^2$ up in a grid of 6×4 equally large rectangles, where there are 4 columns and 6

rows. If X is in the 1st or 3rd column, then T is distributed uniformly over the 1st, 3rd, or 5th row. If X is in the 2nd or 4th column, then T is distributed uniformly over the 2nd, 4th or 6th row.

A.7.3 Training time comparisons

We consider additional training time comparisons for all models in Table 17

Model	**Timing (1 epoch)**
SODEN	5.946 ± 0.54
CoxCC	0.34 ± 0.02
Cox-Linear	1.787 ± 2.537
Cox-Time	7.343 ± 1.004
DeepHit	0.299 ± 0.075
DeepSurv	0.419 ± 0.089
SuMo-net	1.406 ± 0.412

Table 17: Training time for each model for 1 epoch on the SUPPORT dataset. All models used the same batch size of 512 and 2 hidden layers with 32 nodes.

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	Survival Regression with Proper Scoring Rules and Monotonic Neural Networks
Publication Status	☒Published ☐ Accepted for Publication ☐Submitted for Publication ☐Unpublished and unsubmitted work written in a manuscript style
Publication Details	Rindt, D.*, Hu, R.*, Steinsaltz, D. and Sejdinovic, D. (2020) Survival Regression with Proper Scoring Rules and Monotonic Neural Networks. Proceedings of the 25th International Conference on Artificial Intelligence and Statistics (AISTATS 2022)

Student Confirmation

Student Name:	Robert Hu		
Contribution to the Paper	My contributions to the paper was implementation and optimization of the proposed method, experiments and writing. David did the theoretical results and some experiments. During our weekly meetings the supervisors helped by providing suggestions and feedback.		
Signature		Date	29/8-2022

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: Dino Sejdinovic		
Supervisor comments I am in agreement with the description of the contributions		
Signature 	Date	09/09/2022

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

6

Giga-scale Kernel Matrix Vector Multiplication on GPU

Giga-scale Kernel Matrix-Vector Multiplication on GPU

Robert Hu
Amazon *
robyhu@amazon.co.uk

Siu Lun Chau
Department of Statistics
University of Oxford
siu.chau@stats.ox.ac.uk

Dino Sejdinovic
School of Computer and Mathematical Sciences
University of Adelaide *
dino.sejdinovic@adelaide.edu.au

Joan Alexis Glaunès
MAP5
Université Paris Descartes
alexis.glaunes@mi.parisdescartes.fr

Abstract

Kernel matrix-vector multiplication (KMVM) is a foundational operation in machine learning and scientific computing. However, as KMVM tends to scale quadratically in both memory and time, applications are often limited by these computational constraints. In this paper, we propose a novel approximation procedure coined *Faster-Fast and Free Memory Method* (F^3M) to address these scaling issues of KMVM for tall ($10^8 \sim 10^9$) and skinny ($D \leq 7$) data. Extensive experiments demonstrate that F^3M has empirical *linear time and memory* complexity with a relative error of order 10^{-3} and can compute a full KMVM for a billion points *in under a minute* on a high-end GPU, leading to a significant speed-up in comparison to existing CPU methods. We demonstrate the utility of our procedure by applying it as a drop-in for the state-of-the-art GPU-based linear solver FALKON, *improving speed 1.5-5.5 times* at the cost of $< 1\%$ drop in accuracy. We further demonstrate competitive results on *Gaussian Process regression* coupled with significant speedups on a variety of real-world datasets.

1 Introduction

Kernel matrix-vector multiplication (KMVM) is one of the most important operations needed in scientific computing with core applications in diffeomorphic registration, geometric learning [11], [31], numerical analysis [28], fluid dynamics [6], and machine learning [27]. For a dataset of size n, KMVM using direct computation has complexity and memory footprint $\mathcal{O}(n^2)$, both unfeasible for modern large scale applications where $n \approx 10^9$ is becoming increasingly common. Pioneering contributions presented in the *Fast Multipole Method* (FMM) [10] amend the complexity of these problems to $\mathcal{O}(n \log(\epsilon^{-1}))$, where ϵ is the chosen error tolerance, with varying reductions in memory footprint for data restricted to dimension $D = 2$. Subsequent developments in [8, 17] mainly focused on extending approximations for a broader set of kernels for a fixed dimensionality $D \leq 3$, tailored for problems in physics with narrow data such as electrostatics, stellar dynamics, Stokes flow, and acoustic problems, amongst others.

*Work mainly done while the authors were with the Department of Statistics, University of Oxford.

36th Conference on Neural Information Processing Systems (NeurIPS 2022).

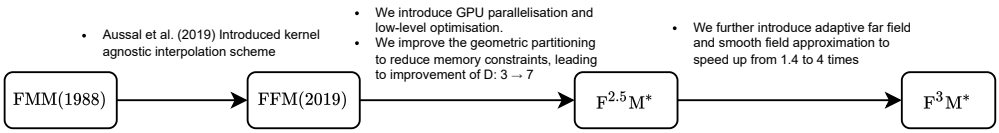


Figure 1: A brief summary of the evolution of the FMM family. Our contributions in *.

In this paper, we introduce *Faster-Fast and Free Memory Method* (F^3M), a novel algorithm built upon the FFM [3] framework to perform KMVM on a GPU for *tall and skinny* ($D \leq 7$) data of order $n \sim 10^9$ in under a minute with user-specified error tolerance, providing between $2 - 8500$ times speed-up over existing methods. It should be noted that the constraints on D and n are not inherent formal constraints, but a reflection of practical limits with typical current computational resources.

Notations. We use capital and lower case bold letters to represent matrices and vectors, respectively. In this paper, we will work with matrices $\mathbf{X} \in \mathbb{R}^{n_x \times D}$, $\mathbf{Y} \in \mathbb{R}^{n_y \times D}$ and vector $\mathbf{b} \in \mathbb{R}^{n_y}$. For a kernel k, the goal for KMVM is to compute $\mathbf{v} := \mathbf{K} \cdot \mathbf{b}$, where $\mathbf{K} := k(\mathbf{X}, \mathbf{Y}) = \{k(\mathbf{x}_i, \mathbf{y}_j)\}_{i=1,j=1}^{n_x,n_y}$, and $\mathbf{x}_i, \mathbf{y}_j$ denote the i^{th}, j^{th} row of $\mathbf{X}, \mathbf{Y}$ respectively.

2 Motivation and Related Work

Kernel methods are often limited by their $\mathcal{O}(n^2)$ memory footprint and computational complexity for KMVM. These constraints make scaling beyond $n = 10^6$ challenging. Many recent developments have been made to improve both of these constraints, ranging from hardware acceleration using GPUs in KeOps [11], to various approximation techniques proposed in [36, 33, 34, 3, 9]. In this work, we focus our attention on kernel independent KMVM methods.

KeOps. Charlier et al. [11] proposes a map-reduce scheme to compute kernels using exactly $\mathcal{O}(n)$ memory and $\mathcal{O}(n^2)$ complexity on GPU. This is achieved by computing the full KMVM product on-the-fly by summing $v_i = \sum_{j=1}^{n} k(\mathbf{x}_i, \mathbf{y}_j) b_j$ directly, without ever storing the kernel matrix $\mathbf{K} \in \mathbb{R}^{n \times n}$ explicitly. Extensive experiments show that this method is practical when $n \leq 10^6$, as the GPU hardware acceleration allows the KMVM product to be computed in less than a second on a conventional GPU. Moreover, the method places no constraint on the number of features D it can be applied to, making it favourable for KMVM on medium size datasets. In application contexts, KeOps is currently adopted into conjugate gradient solver FALKON [24, 26] as part of the default pipeline. [2]

Table 1: Comparison between methods(* indicate ours).

Method	FMM	KeOps	FFM	$\text{F}^{2.5}\text{M}^*$	F^3M^*
Kernel Independent		✓	✓	✓	✓
Linear Time	✓		✓	✓	✓
Linear Memory		✓	✓	✓	✓
Restriction in D	≤ 3		≤ 3	≤ 7	≤ 7
GPU	✓	✓		✓	✓
Scales to $n = 10^9$ under 1 hour [1]				✓	✓
under 1 minute! [1]					✓

The Fast and Free Memory Method (FFM). While KeOps can theoretically scale to a billion points, it becomes practically infeasible as the $\mathcal{O}(n^2)$ complexity would imply a computational time of 10^6 seconds, or roughly 11 days. To overcome this billion points barrier, Aussal et al. [3] deploys a geometric space partitioning scheme, and proposed the *Fast and Free Memory Method* (FFM), a KMVM approach that extends the FMM [10] family of algorithms. In contrast to traditional FMM methods, which require specific series expansion of the kernel, FFM deploys Lagrange interpolations to approximate them instead. This allows FFM to be applied to almost any conventional kernel and further enables the user to trade off accuracy with computational efficiencies by controlling the order of the approximating polynomial [19]. Compared to KeOps, FFM demonstrates both linear memory and time complexity in experiments and scales to compute a billion-points KMVM on a smaller CPU cluster under 4 hours, outscaling the GPU implementation of FMM [21]. While 4 hours is a significant improvement compared to 11 days from KeOPS, it still renders many machine learning techniques infeasible. Further, as recursive partitioning of the data space scales poorly with D [5], both FMM and FFM can only be applied to $D \leq 3$ data, a price to pay for the speed-up of KMVM operations when $n = 10^9$. Furthermore, we show in our experiments that a direct FFM port to GPU gives unstable results for $n = 10^9, D = 3$ for non-trivial data simulations (bottom row in Appendix 10).

[2] KMVM applied to 3D data for $n = 10^9$ on a Nvidia V100 GPU.

Our contribution. To surpass the billion point barrier while maintaining high-speed and stable computation, we propose F$^{2.5}$M and our main algorithm F^{3}M, the first pair of KMVM algorithms that can reliably scale to $n = 10^9$ on skinny data using a single GPU. We build F$^{2.5}$M on top of FFM by introducing non-trivial GPU parallelisation and low-level optimisations. We further stabilize and improve the original geometric partitioning scheme in FFM to significantly reduce memory constraints, leading to a relaxation of dimensionality constraints from 3 to 7. At last, we introduce an adaptive far-field and smooth field approximation scheme for kernel interpolation, resulting in our main algorithm F^{3}M, which runs $2.0 - 33.3$ times quicker and more stable than a direct port of FFM on GPU. See Fig. 1 and Table 1 for an overview and comparisons of the methods. We summarise our contribution as follows:

1. We propose *Faster-Fast and Free Memory Method* (F^{3}M), a KMVM algorithm building on top of FFM by applying multiple low-level enhancements, GPU parallelisation, and algorithmic computational and memory enhancements, allowing for KMVM operations on $n \leq 10^9$ data in under a minute. Codebase is released here [14].

2. We characterize theoretical time and memory complexity of F^{3}M.

3. We run extensive KMVM experiments of F^{3}M on a variety of tall and skinny data with $n \leq 10^9$, demonstrating empirical linear time and memory scaling, and achieving speedups between 2–8500 times when compared to FFM (GPU and CPU) and KeOps.

4. We run a practical application of F^{3}M as a drop-in replacement for KeOps in conjugate gradient solver FALKON [24, 26] for kernel ridge regression and classification (KRR) on giga-scale data, obtaining a solution 3.4 times faster with <1% drop in accuracy. We further demonstrate competitive results on Gaussian process regression against KISS-GP [35], SVGP [18] and SVGR [30] with significant speed-ups.

3 Background

The FFM method considers KMVM for a kernel k evaluated on two data matrices $\mathbf{X}, \mathbf{Y}$ and $\mathbf{b}$ are weights associated with $\mathbf{Y}$. The KMVM is expressed as $\mathbf{v} := k(\mathbf{X}, \mathbf{Y}) \cdot \mathbf{b} = \mathbf{K} \cdot \mathbf{b}$. For example, $\mathbf{b}$ could be the weights in a KRR or the strength of electronic charges. As n_x and n_y are taken to be very large, a full computation is unfeasible. In this section, we illustrate and detail the main steps of FFM, before presenting our improvements in Section 4. For illustration purposes, we first consider a simple 2D KMVM. Our goal is to calculate $k(\mathbf{X}, \mathbf{Y}) \cdot \mathbf{b}$ for $\mathbf{X}, \mathbf{Y}$ in Figure 2. The intuition behind FFM is to reduce the complexity of calculating the full KMVM by partitioning $\mathbf{X}$ and $\mathbf{Y}$ such that certain calculations can be approximated in a fast manner, based on the pairwise distances between partitions.

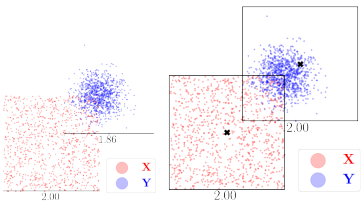


Figure 2: Enclosing $\mathbf{X}$ and $\mathbf{Y}$ within a box, "x" marks the center of the box. Numbers under the the boxes denotes edge length. In the right plot, we have enclosed the blue points with the largest box.

Enclosing and partitioning the data. The first step is to partition the data. To begin, we find a large enough box that can just enclose $\mathbf{X}$ or $\mathbf{Y}$. The edge length of this box is calculated as

$$\mathcal{E} := \max\left(\max_d(x^{(d)}_{\max} - x^{(d)}_{\min}), \max_d(y^{(d)}_{\max} - y^{(d)}_{\min})\right)$$

where $x^{(d)}_{\max}, x^{(d)}_{\min}$ denotes the largest value and the smallest value along the d-dimension in $\mathbf{X}$ and similarly for $\mathbf{Y}$. Figure 2 illustrates this enclosing procedure.

Defining near and far-field In FMM, an octree [23] is applied to recursively partition data into smaller boxes $B^X_p \subset \mathbf{X}, B^Y_q \subset \mathbf{Y}$, with p, q denoting box indices. Here each box corresponds to a subset of rows in the data matrix. Let us also denote $\mathbf{b}_q$ as the partition of b_j's grouped with the same indices as B^Y_q. To calculate the KMVM between two boxes B^X_p, B^Y_q with the grouped vector $\mathbf{b}_q$, for each $\mathbf{x}_i \in B^X_q$, we com-

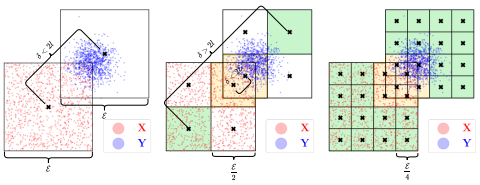


Figure 3: Recursive partitioning of $\mathbf{X}$ and $\mathbf{Y}$ for 2D data. Far-field interactions are colored green while near-field interactions are colored orange and δ denotes the euclidean distance between the centers.

pute

$$v_i^{p,q} = \sum_{\mathbf{y}_j \in B_q^Y, b_j \in \mathbf{b}_q} k(\mathbf{x}_i, \mathbf{y}_j) b_j \qquad (1)$$

with $\mathbf{v}^{p,q} = [v_1^{p,q} \dots v_{n_x}^{p,q}]$. Now the target $\mathbf{v}$ can be computed as $\mathbf{v} = \sigma([\mathbf{v}^{p=1}, \dots, \mathbf{v}^{p=P}]^\top)$, where $\mathbf{v}^p = \sum_{q=1}^{Q} \mathbf{v}^{p,q}$, P, Q denote the total number of boxes and $\sigma(\cdot)$ a permutation such that v_i^p appear in the same order as $\mathbf{x}_i$ appears in $\mathbf{X}$. Figure 3 shows how boxes are recursively partitioned.

Far and near-field interactions FFM relies on a divide-and-conquer strategy to effectively compute a KMVM product; data is partitioned into boxes and then separated into far-field and near-field interactions, where near-field interactions are computed exactly and far-field interactions are approximated using Lagrange interpolation for speed, explained in the paragraph below. The partitioning procedure in FFM is recursive, where the recursion depth `tree_depth` controls the size of the edge $l = \frac{\mathcal{E}}{2^{\texttt{tree_depth}}}$ of the box. An interaction is defined to be in the far-field if the distance between the two center points of the boxes exceeds $2l$, i.e. $\texttt{IsFarField} := \|\mathbf{x}_{\text{center}} - \mathbf{y}_{\text{center}}\| \geq 2l$. While l for each box will decrease with the number of divisions, this rule ensures a fixed minimal distance for a given depth for far-field interactions. Figure 3 illustrates how far(green) and near(orange)-field interactions arise between $\mathbf{X}$ and $\mathbf{Y}$ when `tree_depth` increases.

Lagrange interpolation We review Lagrange interpolation used for far-field approximations in FFM. Given a function $f(x) : [-1, 1] \to \mathbb{R}$ and $r+1$ unique points $s_i \in [-1, 1], \quad i = 0, \dots, r$, there exists a unique polynomial $p_r(x)$ of degree $\leq r$ that interpolates f at $p_r(s_i) = f(s_i)$. The Lagrange polynomial is given by $p_r(t) = \sum_{i=0}^{r} f(s_i)\mathcal{L}_i(t)$, where $\mathcal{L}_i(t) = \frac{\prod_{j=0, j\neq i}^{r}(t - s_j)}{\prod_{j=0, j\neq i}^{r}(s_i - s_j)}, \quad i = 0, \dots, r$. We are free to chose the degree r as well as the points s_i to interpolate through. The choice of s_i is especially important in minimizing large oscillations around the edges of the interpolation interval (Runge's phenomenon [12]). For this reason, Chebyshev nodes of the second kind are used [7] $s_i = \cos\theta_i$, where $\theta_i = \frac{i\pi}{r}, \quad i = 0, \dots, r$.

Interpolating $k(\mathbf{x}, \mathbf{y})$ By noticing that $k(\mathbf{x}, \mathbf{y})$ is a bivariate function, we can apply Lagrange interpolation twice, thus interpolating $k(\mathbf{x}, \mathbf{y})$ as $k(\mathbf{x}, \mathbf{y}) \approx \sum_{i=1}^{r_X} \mathcal{L}_i(\mathbf{x}) \sum_{j=1}^{r_Y} k\left(\mathbf{s}_i^x, \mathbf{s}_j^y\right) \mathcal{L}_j(\mathbf{y})$. Here r_X, r_Y denotes the number of the interpolation nodes and $\mathbf{s}_i^x, \mathbf{s}_j^y \in \mathbb{R}^D$ denotes the grid of interpolation nodes for B_p^X and B_q^Y. Note that since $\mathbf{x} \in \mathbb{R}^D$, we take $\mathcal{L}_i(\mathbf{x}) := \prod_{d=1}^{D} \frac{\prod_{j=0, j\neq i}^{r}\left(x^{(d)} - s_j^{(d)}\right)}{\prod_{j=0, j\neq i}^{r}\left(s_i^{(d)} - s_j^{(d)}\right)}, \quad i = 0, \dots, r$. These operations can be vectorized and computed sequentially on-the-fly with linear memory footprint $\mathbf{v} \approx \mathbf{L}_X^T \cdot \overbrace{(\mathbf{K} \cdot \underbrace{(\mathbf{L}_Y \cdot \mathbf{b})}_{\mathbf{v}_1})}^{\mathbf{v}_2}$, which is done by first computing $\mathbf{v}_1$, then $\mathbf{v}_2$ and lastly $\mathbf{v}$. Here $\mathbf{L}_X$ denotes a matrix with entries $\mathcal{L}_i(\mathbf{x}_j)$, where i indexes the rows and j the columns, with $\mathbf{L}_Y$ following the same definition for $\mathbf{y}_j$'s instead. A far-field KMVM between two boxes $\mathbf{v}_{p,q} = k(B_p^X, B_q^Y) \cdot \mathbf{b}_q$ is then approximated by using double Lagrange interpolation according to Figure 4.

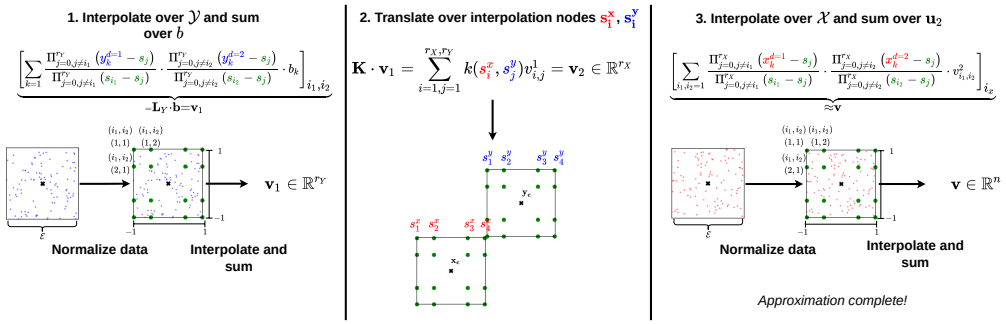


Figure 4: Here we approximate a far-field interaction between two boxes. In **1.** we first normalize the data between $[-1, 1]$ and conduct 2D interpolation of $k(\cdot, \mathbf{y})$ while summing over $\mathbf{b}$. In **2.**, the interpolation for $k(\mathbf{x}, \cdot)$ will also be normalized and hence we need to translate the distance between the boxes by calculating $\mathbf{K} \cdot \mathbf{v}_1$. Lastly in **3.** we interpolate $k(\mathbf{x}, \cdot)$ while we sum $\mathbf{v}_2$.

4 Faster-FFM (F$^{2.5}$M and F^{3}M)

To fully leverage the port of FFM to GPU, we enhance FFM with novel approximation procedures for improved complexity and memory optimizations to scale to $n = 10^9$. We coin this improved version Faster-FFM (F^{3}M). The capabilities of F^{3}M against previous methods are summarized in Table 1.

4.1 CPU to GPU optimizations

In FFM, every computation is serial and on CPU. When moving to GPU, we have parallelized all major computations. These parallelizations are non-trivial and require low-level algorithmic optimizations, with challenges such as:

Box-to-threadblock alignment – A major challenge in the implementation of both the parallel far-field and near-field computations was correctly aligning thread blocks to boxes. This aligning requirement imposed non-trivial boundary conditions on data indexing when using shared memory. To minimize memory usage of box and block indicators for our implementation, we represented the box belonging of each point as index intervals (i.e. box 1 consists of points with $i \in [1, \ldots, 500]$ and box 2 with $i \in [501, \ldots, 1337]$, etc.) and modulo arithmetic to infer the block belonging. This clearly requires that the points are sorted or grouped according to their box belonging. However, as we detail in the next paragraph, arranging the points could not be done straightforwardly with native sorting methods. We further illustrate how parallelization is done for calculating near-field interactions in Appendix I.

No native sorting methods – We found that LibTorch [25] sorting methods often led to out-of-memory (OOM) due to allocation of large long-type vectors on GPU. When $n = 10^9$, this implies allocating 8GB of memory, 25% of the 32GB card used, making it a necessity to avoid native sorting methods.

In-place grouping data on boxes – Due to infeasible LibTorch sorting methods, we additionally had to design an algorithm that finds a permutation that would group $\mathbf{X}$ into its corresponding boxes in linear time and memory. We used a count and increment-based strategy that would:

1. Count the number of points in each box during the assignment operation ($\mathcal{O}(n)$) and store the count in a vector ξ. Then run a cumulative sum over ξ, starting from 0.

2. Initialize a n long permutation vector π. Using the counting vector ξ, we would re-run the assignment operation and arrange a point with index i as following $\pi[\texttt{atomicAdd}(\xi'[\texttt{box_index}], 1) + \xi[\texttt{box_index}]] = i$, where ξ' is a running count of points in each box. We specifically have to use the function `atomicAdd` to increment the count for each box in parallelized GPU environments to avoid thread locks.

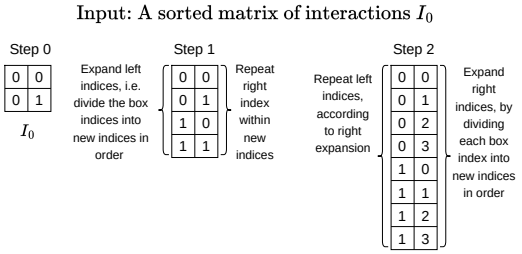


Figure 5: Here we assume $D = 1$, hence we only divide each box $2^D = 2$ times each time.

We refer to the `box_division_cum_hash` and `box_division_assign_hash` function in `n_tree.cu` for exact details.

Ensuring interactions are sorted – To avoid any unnecessary sorting, we ensure that the matrix containing interactions is always sorted by recursively dividing old interactions. We illustrate the procedure in Figure 5. We refer to the `get_new_interactions` function in `n_tree.cu` for the exact implementation.

However, we found that these optimizations and porting alone were not enough to scale to $n = 10^9$ on 3D datasets, as Figure 6 demonstrates. FFM doesn't remove empty boxes or handle boxes with few points in them and keeps exponentially creating new empty boxes and interactions, thus leading to out-of-memory (OOM) errors on non-uniform data (see Appendix 10).

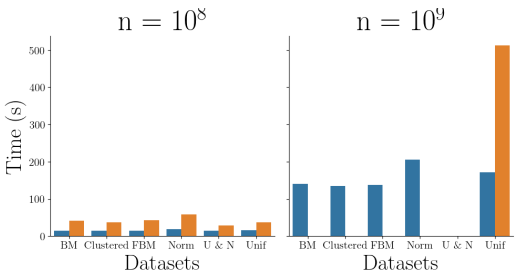


Figure 6: KMVM times on several datasets between FFM(GPU) vs F^{3}M. U&N was only run up to $n = 5 \cdot 10^8$

4.2 Scaling to $n = 10^9$ on GPU ($\mathbf{F}^{2.5}\mathbf{M}$)

In this section, we detail the memory enhancements that allow $\mathrm{F}^3\mathrm{M}$ to consistently scale to $n = 10^9$.

Removing empty boxes with hash list indexing To ensure linear memory on GPU, we only keep a reindexing vector σ of size n_x(resp. n_y) in memory during the computation of the algorithm in addition to a list of interactions and box centers. This reindexing vector rearranges the data points so they appear in the order of the box they belong to. We optimize both the computation and the memory footprint of these objects by avoiding recursive formulas and hash lists.

Naively, points can be assigned to boxes by direct comparison to all existing box centers. As the number of centers grows exponentially with depth `tree_depth`, this method quickly becomes pathological. To amend this, we propose a linear complexity formula to retrieve the box index β_i a point $\mathbf{x} \in \mathbf{X} \subset \mathbb{R}^D$ belongs to $\beta_i = \underbrace{\sum_{d=1}^{D} 2^{\texttt{tree_depth}\cdot(d-1)}}_{\text{Summing over } D \text{ dimensions}} \cdot \underbrace{\lfloor 2^{\texttt{tree_depth}} \frac{x_d - \alpha_d}{\mathcal{E}} \rfloor}_{\in\{0,1\},\ \text{Denotes left or right of center of box edge}}$, where α_d denotes the minimum value of $\mathbf{X}$ in dimension D and x_d is the value of $\mathbf{x}$ in dimension D. To prevent the number of boxes from growing exponentially, we remove empty boxes with each division. To assign points to the corresponding boxes, we use a hash list to store β_i and the order i. We can then group points $\{\mathbf{x}_i\}_{i=1}$ to their respective ordering i using the hash list in $\mathcal{O}(n)$ time in contrast to $\mathcal{O}(n \cdot 2^{D\cdot\texttt{tree_depth}})$ by direct computation.

Handling boxes with few points with *small field* In cases when the number of points in each box can vary greatly, we separately consider the interactions where the number of points in boxes is small. Hence, we say that there is a *small field* interaction between boxes B_p^X, B_q^X if both have a small number of points, i.e. if $|B_p^X| + |B_q^X| \leq \rho$, for some threshold number ρ. To minimize the computations needed, ρ can be set to $\rho = r_X + r_Y$. This intuitively allows $\mathrm{F}^3\mathrm{M}$ to directly compute interactions that are too small to benefit from interpolation savings (i.e. $|B_p^X| + |B_q^X| \leq r_X + r_Y$), thus limiting memory usage by stopping partitions from dividing further than necessary. In higher dimensions where the division rate is faster, ρ can be set to a higher value to limit memory usage at the expense of more direct computations which are slower.

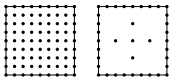
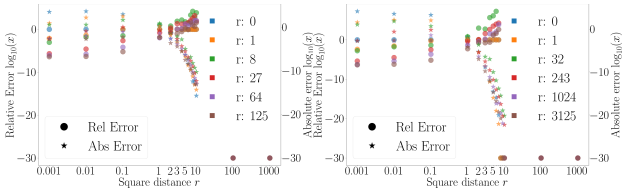


Figure 7: Full grid vs sparse grid.

Sparse grids As the number of Lagrange polynomials increases exponentially with dimension, we implement sparse grids [29] to allow for a finer selection of interpolation nodes. With sparse grids, the number of nodes needed grows slower [20], thus saving memory. We give an example of a sparse grid versus a full grid in 2D in Figure 7.

4.3 Speeding up $\mathbf{F}^{2.5}\mathbf{M}$ $\left(\mathbf{F}^{2.5}\mathbf{M} \to \mathbf{F}^{3}\mathbf{M}\right)$

Smoothness criteria FFM speeds up its computations with minimal loss in accuracy by selectively interpolating interactions that are far apart. To improve speed, we introduce the *smoothness criterion* to widen the selection of interactions that can be interpolated with minimal loss in accuracy. For a Gaussian Kernel $k(\mathbf{x}, \mathbf{y}) = \exp\left(\frac{\|\mathbf{x}-\mathbf{y}\|^2}{2\gamma^2}\right)$, with lengthscale γ, the smoothness criteria is defined as $\texttt{is_smooth} := \frac{\frac{1}{N_{B_p^X}} \sum_{\mathbf{x}_i \in B_p^X} \sum_{d=1}^{D} \|x_i^{(d)} - \bar{x}^{(d)}\|^2}{2\gamma^2} + \frac{\frac{1}{N_{B_q^Y}} \sum_{\mathbf{y}_j \in B_q^Y} \sum_{d=1}^{D} \|y_j^{(d)} - \bar{y}^{(d)}\|^2}{2\gamma^2} \leq \eta$ between an adjacent interaction of boxes B_p^X, B_q^Y. The quantity computed can be understood as "Effective Variance" (EV), as it considers total variation in the exponent of the Gaussian kernel. We justify the *smoothness criteria* with the following proposition.



(a) Dimension $d = 3$ (b) Dimension $d = 5$

Figure 8: Plotting relative and absolute error against squared distance between boxes. "0" nodes mean we use the zero vector as an approximation to the KMVM for a gaussian kernel. We observe that its not beneficial to interpolate at all when the square distance exceeds 5 and that is is sufficient to only use 3^D nodes when square distance is ≤ 0.01.

Proposition 1. *Consider* $\mathbf{x}, \mathbf{y} \in \mathcal{X} \subset \mathbb{R}^d$ *such that* $d(\mathbf{x}, \mathbf{y}) := \frac{\|\mathbf{x}-\mathbf{y}\|^2}{2\gamma^2} = \frac{1}{2\gamma^2}\sum_i^d (x^{(i)} - y^{(i)})^2 \leq \eta < 1$ *for all* $\mathbf{x}, \mathbf{y}$. *When interpolating* $k(\mathbf{x}, \mathbf{y}) = \exp(-d(\mathbf{x}, \mathbf{y}))$ *using bivariate Lagrange interpolation* $\mathcal{L}_r(\mathbf{x}, \mathbf{y}) := \mathbf{L}_X^T \cdot \mathbf{K} \cdot (\mathbf{L}_Y \cdot \mathbf{b})$ *with degree* $r = 2p$, *for any* $p \in \mathbb{N}_{>0}$ *there exist nodes* $\mathbf{s}^{\mathbf{x}}, \mathbf{s}^{\mathbf{y}}$ *for* $\mathcal{L}_r(\mathbf{x}, \mathbf{y})$ *such that the pointwise interpolation error is bounded by* $\mathcal{O}(\eta^{p+1})$.

See Appendix D for proof. Hence for small $\eta < 1$, we see that the error becomes small for well specified $\mathcal{L}_r(\mathbf{x}, \mathbf{y})$. To avoid calculating the sample variance during computations which costs $\mathcal{O}(n)$, we exploit that data is partitioned into hypercubes with a *known* edge $\mathcal{E}$ and take the upper bound of the variance in each cube as $\frac{\mathcal{E}^2}{4}$ along a dimension. A proof for this bound is provided in Appendix B. Adjacent interactions are then classified as smooth when $\sum_d^D \frac{\mathcal{E}^2}{4\cdot 2\gamma}(\text{EV of } B_p^X) + \frac{\mathcal{E}^2}{4\cdot 2\gamma}(\text{EV of } B_q^Y) = \frac{D\mathcal{E}^2}{\gamma^2 \cdot 4} \leq \eta$ which only costs $\mathcal{O}(1)$ to compute.

Adaptive far-field approximation To further improve speed we introduce an adaptive rule to select the number of interpolation nodes used when calculating far-field interactions. Error bounds for multidimensional Lagrange interpolation have been proposed in [22], however, these bounds cannot be directly used to create an adaptive interpolation rule. We thus simulate KMVM errors for $k(\mathbf{X}, \mathbf{Y}) \cdot \mathbf{b}$ where $\mathbf{X}, \mathbf{Y}$ are uniformly distributed and $\mathbf{b}$ is normally distributed. We fix a distance between $\mathbf{X}$ and $\mathbf{Y}$ and vary the squared of this distance between boxes against nodes in Figure 8. We use a Gaussian Kernel with $\gamma = \frac{1}{\sqrt{2}}$.

Based on Figure 8, we use the following rule for selecting the number of interpolation nodes for far-field interactions

$$r_{\text{far}}(r) = \begin{cases} \min(r, 3^D) & \text{if } \left(\frac{\mathcal{E}}{2^{c_{\text{depth}}}}\right)^2 \cdot \frac{1}{2\gamma^2} \leq 0.01 \\ r & \text{if } 0.01 < \left(\frac{\mathcal{E}}{2^{c_{\text{depth}}}}\right)^2 \cdot \frac{1}{2\gamma^2} \leq 5 \\ 0 & \text{if } 5 < \left(\frac{\mathcal{E}}{2^{c_{\text{depth}}}}\right)^2 \cdot \frac{1}{2\gamma^2} \end{cases}$$

where r is the number of nodes chosen to interpolate with in the general case.

Barycentric lagrange interpolation We slightly improve the complexity further by implementing barycentric Lagrange interpolation [7] evaluated at the Chebyshev nodes of the second kind. As this is a well-known technique, we refer to the appendix for more details. It should be noted that the above methods can straightforwardly be extended to any *translation-invariant* kernel by recalculating the Taylor expansion for *smoothness criteria* and rerunning the simulation for *adaptive far-field approximation*.

4.4 Complexity

The time complexity of FFM is $\mathcal{O}(n \log(n))$ [3] and we use a similar derivation strategy for F^{3}M to obtain a complexity that is dependent on the effective variance limit η (chosen parameter) and the box width $\mathcal{E}$ (data). We first present two propositions needed to derive the complexity of F^{3}M.

Proposition 2. *A far-field interaction between two boxes containing* n_x *and* n_y *points respectively has time complexity* $\mathcal{O}(n)$, *where* $n = \max(n_x, n_y)$.

Proposition 3. *Given* n *data points in dimension* D, *the maximum number of divisions* $\mathit{Tree}_{\mathit{max\ divisions}}$ *is given by*

$$\mathit{Tree}_{\mathit{max\ divisions}} = \log_{2^D}(n). \tag{2}$$

With the above results, the complexity of FFM is taken as the maximum number of divisions multiplied by the complexity of far-field interactions at each division which yields $\mathcal{O}(n \log(n))$. We remark that near-field interactions between boxes containing only 1 data point have linear time complexity, hence the results hold.

Theorem 1. *Given a KMVM with edge* $\mathcal{E}$ *(dependent on data* $\mathcal{X}, \mathcal{Y}$*), lengthscale* γ, *effective variance limit* η, n *data points and data dimension* D, *F^{3}M has time complexity* $\mathcal{O}(n \cdot \log_2\left(\frac{D\cdot\mathcal{E}^2}{\gamma^2\cdot 4\cdot\eta}\right))$, *which can be taken as* $\mathcal{O}(n \cdot \log_2\left(\frac{C}{\eta}\right))$ *where* $C \propto \frac{D\cdot\mathcal{E}^2}{\gamma^2}$.

Memory footprint As our implementation uses the same partitioning strategy as FFM, the *theoretical* memory complexity remains $\mathcal{O}(n)$ for F^{3}M (see [3] for proof). However, this does not accurately reflect the memory footprint of the actual implementations, whose memory mostly depends

on the number of interactions stored. We summarize these memory footprints for FFM and F^{3}M in Theorem 2 below.

Theorem 2. *The number of interactions M_i against tree depth i of FFM and F^{3}M grows as $\mathcal{O}\left(M_{i-1}2^{2\cdot D} - m_i^{far}\right)$ and*

$$\mathcal{O}\left(M_{i-1}2^{2\cdot D} - (m_i^{empty})^2 - m_i^{far} - m_i^{smooth} - m_i^{small}\right)$$

respectively. Here $M_{-1} = \frac{1}{2^{2D}}$ and $m_0^{far} = m_0^{smooth} = m_0^{small} = m_0^{empty} = 0$ and $m_i^{far}, m_i^{smooth}, m_i^{small}, m_i^{empty}$ denotes the number of far-field, smooth field, small field interactions and the number of empty boxes respectively at depth $i > 0$.

We see that the additional approximations presented in F^{3}M also impacts memory footprint, as the additional $(m_i^{\text{empty}})^2, m_i^{\text{smooth}}, m_i^{\text{small}}$ terms removes a substantial amount of interactions at each i, significantly slowing down the growth of interactions, reducing memory growth. The efficacy of $(m_i^{\text{empty}})^2, m_i^{\text{smooth}}, m_i^{\text{small}}$ is widely dependent on data. As an example, data with points very close to each other would significantly benefit m_i^{smooth} more, as the closeness of points would imply more smooth interactions. If points are sparsely spread out, $(m_i^{\text{empty}})^2, m_i^{\text{small}}$ would provide the most benefit as they remove empty boxes and stops boxes with few points to divide unnecessarily. All proofs can be found in Appendix E.

5 Experiments

We demonstrate the utility of F^{3}M over a variety of experiments using the Gaussian kernel $k(\mathbf{x}, \mathbf{y}) = \exp -(\frac{\|\mathbf{x}-\mathbf{y}\|^2}{2\gamma^2})$.[3] We generate data such that the EV (see section 4.3) varies between $0.1, 1, 10$ for data of sizes $n = 10^6, 10^7, 10^8, 10^9$. The parameters used for F^{3}M are $\eta = 0.1, 0.2, 0.3, 0.5$ and $r = 2^D, 3^D, 4^D$ with a cap at $r = 2048$. The error for the approximated KMVM product $\hat{\mathbf{v}}$ is calculated as Relative error $:= \frac{\|\hat{\mathbf{v}}-\mathbf{v}\|^2}{\|\mathbf{v}\|^2}$, where the true KMVM product $\mathbf{v}$ is obtained by calculating the full KMVM on a subset $\mathbf{X}'$ consisting of the first 5000 points in $\mathbf{X}$ against the entire dataset in double precision, i.e. $\mathbf{v} = k(\mathbf{X}', \mathbf{X}) \cdot \mathbf{b}$, where we fix $\mathbf{b} \sim \mathcal{N}(0, I_n)$. All experiments were run on NVIDIA V100-32GB cards, where the data is fitted entirely on the GPU. These cards were chosen since the extra graphic memory is necessary to fit the data on one card when $n = 10^9$. It should be noted that $n = 10^9$ can only be run up to $D = 3$, as $\mathbf{X}$ and $\mathbf{b}$ itself cannot fit in memory for higher dimensions with the GPUs we had available. For details on how F^{3}M scales across multiple GPUs, see Appendix G.

Table 2: F^{3}M (GPU) compared to results reported in [3] for FFM(CPU). F^{3}M achieves a $90\times$ speed up on a billion data points. F^{3}M used parameters $r = 64$ and $\eta = 0.5$

FFM (12 CPU cores)				**F^{3}M (GPU, Ours)**			
n	Time (s)	Error	Memory	Time (s)	Error	Memory	**Speedup**
10^6	33.4	$1.35 \cdot 10^{-4}$	100 MB	0.08 ± 0.00	$3 \cdot 10^{-4} \pm 7 \cdot 10^{-5}$	$\sim$ 28 MB	$417\times$
10^7	169	$1.98 \cdot 10^{-4}$	1GB	1.16 ± 0.04	$3 \cdot 10^{-4} \pm 1.2 \cdot 10^{-4}$	$\sim$ 280 MB	$145\times$
10^8	1499	$1.81 \cdot 10^{-4}$	10 GB	12.45 ± 0.06	$2 \cdot 10^{-4} \pm 5 \cdot 10^{-5}$	$\sim$ 2.8 GB	$120\times$
10^9	11340	$3.11 \cdot 10^{-4}$	100 GB	125.90 ± 0.52	$3 \cdot 10^{-4} \pm 1.3 \cdot 10^{-5}$	$\sim$ 28 GB	$90\times$

KMVM experiments We consider a wide variation of generated datasets to simulate different real-world scenarios to test F^{3}M on. For the $k(\mathbf{X}, \mathbf{X})$-case we consider uniformly and normally distributed data ($D = 1, 2, 3, 4, 5, 6, 7$) together with data simulated from Brownian motion, fractional Brownian motion, and Clustered data ($D = 1, 2, 3$). For the $k(\mathbf{X}, \mathbf{Y})$-case we consider uniformly distributed $\mathbf{x}$ and normal distributed $\mathbf{y}$ ($D = 1, 2, 3, 4, 5, 6, 7$). See Appendix 10 for visualizations of data. We have to consider smaller n for the $k(\mathbf{X}, \mathbf{Y})$-case when $D \geq 3$, as twice the amount of data needs to be stored. For $D = 3/(4, 5)/(6, 7)$ we instead consider at most $n = 5 \cdot 10^8/2.5 \cdot 10^8/10^8$. It should be noted that $D = 7$ is a hard limit for geometric partitioning-based methods, since for $D = 8$, we would have $2^{8\cdot 2} \cdot 2^{8\cdot 2} \approx 4.3 \cdot 10^9$ interactions after only 2 divisions. This number of interactions cannot even be represented by a 32-bit integer. We summarize the runs in Table 3 and plot the error and time complexity in Figure 9 for each dataset when $D = 3$. We find that F^{3}M maintains sub-linear empirical complexity up to $D = 6$, where we have to set *small field* limit ρ to a larger number to

[3] F^{3}M is kernel agnostic, however we choose the Gaussian kernel for simplicity.

Table 3: Run time and relative error of all KMVM experiments for F^{3}M. The slope is computed by regressing $\log_{10}(\text{Time (s)})$ against $\log_{10}(n)$. A slope of 1 implies $\mathcal{O}(n)$ scaling.

	Time(s)							**Relative Error**						
n/D	1	2	3	4	5	6	7	1	2	3	4	5	6	7
10^6	0.2 ±0.1	0.2 ±0.2	0.2 ±0.1	0.4 ±0.2	0.9 ±0.9	3.0 ±2.4	3.1 ±2.1	0.0018 ±0.0021	0.0023 ±0.0028	0.0005 ±0.0009	0.0014 ±0.0013	0.0022 ±0.0022	0.0303 ±0.0314	0.0294 ±0.0282
10^7	0.7 ±0.3	1.1 ±0.4	1.7 ±0.7	4.0 ±2.4	12.7 ±10.0	79.9 ±88.6	88.2 ±90.2	0.0016 ±0.0021	0.0019 ±0.0023	0.0006 ±0.0013	0.0017 ±0.0019	0.0057 ±0.0032	0.0325 ±0.0261	0.0273 ±0.0179
10^8	3.5 ±1.1	7.4 ±2.1	17.1 ±7.3	40.6 ±28.7	76.0 ±83.2	525.1 ±410.9	512.9 ±471.4	0.002 ±0.0026	0.0025 ±0.0032	0.0007 ±0.0013	0.0023 ±0.0021	0.0051 ±0.0032	0.0376 ±0.0239	0.0444 ±0.017
$2.5 \cdot 10^8$	N/A	N/A	N/A	76.7 ±25.9	340.7 ±247.6	OOM	OOM	N/A	N/A	N/A	0.0034 ±0.0029	0.0076 ±0.0012	N/A	N/A
$5 \cdot 10^8$	N/A	N/A	74.9 ±28.5	289.4 ±222.8	631.5 ±997.0	OOM	OOM	N/A	N/A	0.0012 ±0.0013	0.0023 ±0.0026	0.0041 ±0.003	N/A	N/A
10^9	29.4 ±8.9	73.8 ±23.4	174.0 ±76.6	OOM	OOM	OOM	OOM	0.0024 ±0.0024	0.0025 ±0.0027	0.0009 ±0.0018	N/A	N/A	N/A	N/A
Slope	**0.78**	**0.85**	**0.99**	**0.98**	**0.99**	**1.06**	**1.02**							
$\mathcal{O}(n\log(n))$				1.11										

Table 4: FALKON using default KMVM vs FALKON with F^{3}M.

				FALKON with default KMVM		**FALKON with F^{3}M**			
Dataset	n	D	M	R^2	Time (s)	R^2	Time (s)	Error diff	**Speedup**
Uniform	10^9	3	10^5	0.975 ± 0.034	7631 ± 2	0.976 ± 0.038	2234 ± 429	0%	5.31
Normal	10^9	3	10^5	0.893 ± 0.118	7631 ± 2	0.902 ± 0.114	2234 ± 429	1%	3.41
OSM	10^9	2	10^5	0.932 ± 0.056	6752 ± 13	0.943 ± 0.043	1670 ± 48	1.2%	4.04
NYC Taxi	10^9	3	10^5	0.526 ± 0.029 (AUC)	6963 ± 69	0.526 ± 0.030 (AUC)	4535 ± 7	0%	1.53

not run out of memory. Further, the error increases in the higher dimensions since we use fewer nodes per dimension when interpolating, owing to the *sparse grid* technique. We note that $D = 7$ has faster run times than $D = 6$ which is explained by that for some values of EV, $D = 7$ doesn't run with acceptable errors which skew the run time to datasets where a larger portion of the data can be interpolated.

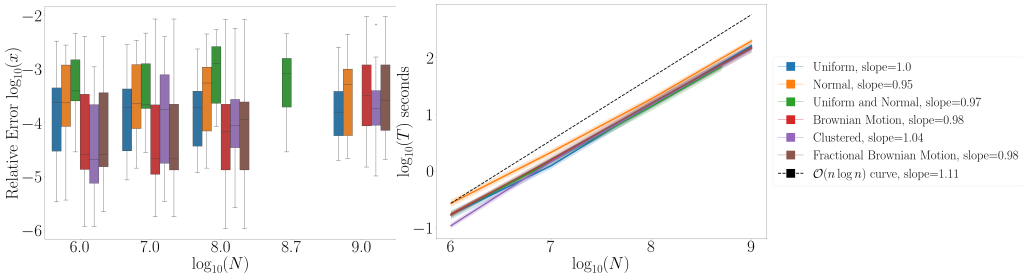


Figure 9: Relative error and time complexity for each 3D dataset.

We further replicate the data used in the first experiment in [3] and compare F^{3}M against FFM (CPU) in Table 2.

Kernel Ridge Regression experiment We apply F^{3}M to FALKON [24], where we replace their KMVM operation with F^{3}M and compare performance and speed in solving Kernel Ridge Regression (KRR). The KMVM operation currently used for smaller dimensions is KeOps [11]. Given some data $\mathbf{X} \in \mathbb{R}^{N \times d}$ we want to find the solution $\alpha = (k(\mathbf{X}, \mathbf{X}) + \lambda I)^{-1}\, \mathbf{b}$ where λ is the ridge parameter that stabilizes the inverse. FALKON is a Nyström approximation based solver that requires a subsample $\mathbf{X}' \in \mathbb{R}^{M \times d}$ of $\mathbf{X}$ to approximate the inverse computation. We focus the experiments on tall and skinny data and take $n = 10^9, d \leq 3$ with $M = 10^5$ for all experiments. We consider uniformly and normally sampled data, the Open Street Map (OSM) dataset [1] and a classification task on the NYC Taxi dataset [2], where we predict whether the customer will tip based on trip distance, trip time and fare cost. To construct $\mathbf{b}$ on synthetic problems, we first take a subset $\mathcal{D} \in \mathbb{R}^{1000 \times d}$ of $\mathbf{X}$ and sample $\alpha \sim \mathcal{N}(0, I_{1000 \times 1000})$. We then calculate $\mathbf{b} = k(\mathbf{X}, \mathcal{D}) \cdot \alpha + \varepsilon$, where $\varepsilon \sim \mathcal{N}(0, 0.1)$. We run KRR for $EV = 0.1, 1, 10$ on synthetic data, and report the average R^2 (AUC for NYC Taxi) and training time in Table 4. For the real world datasets OSM and NYC Taxi, we fix the lengthscale using the median heuristic proposed in [16] averaged our results over the 3 runs.

Ablation study between FFM(GPU) and F^{3}M As much of the improved performance can be attributed to our GPU implementation, we conduct an ablation study of FFM(GPU) against F^{3}M and KeOps in Table 5. We first present KMVM run times averaged over $D = 3$ and real-world datasets OSM and NYC Taxi. For KeOps, we only computed the KMVM on uniform data. Since KeOps is an exact method, the dataset distribution has no effect on computational time. Here, the *smoothness*

criteria and *adaptive far-field* technique improve computational time. We find that F^{3}M achieves a speed-up between $2.0 - 33.3\times$ against FFM(GPU) and $8.0 - 8500\times$ speed-up against KeOps.

Table 5: Comparison between F^{3}M, FFM(GPU) and KeOps. It should be noted that KeOps is only run up to $n = 10^8$ for all experiments (a run for $n = 10^9$ would take weeks). The times for $n > 10^8$ are extrapolated for KeOps. FFM(GPU) could only run on uniform data for $n = 10^9$.

	F^{3}M time (s)			**FFM(GPU) time (s)**			**KeOps time (s)**	**Speedup vs KeOps**	**Speedup vs (GPU)**		
n	OSM ($D=2$)	Taxi ($D=3$)	$D=3$	OSM ($D=2$)	Taxi ($D=3$)	$D=3$	$D=3$		OSM ($D=2$)	Taxi ($D=3$)	$D=3$
10^6	0.1 ± 0.0	0.5 ± 0.2	0.2 ± 0.1	0.7 ± 0.2	1.2 ± 0.2	0.4 ± 0.2	1.56	8.0	**7.0**	**2.4**	**2.0**
10^7	1.1 ± 0.4	3.3 ± 0.6	1.7 ± 0.7	10.1 ± 0.8	110.0 ± 1.9	3.7 ± 1.0	148.6	87.0	**9.2**	**33.3**	**2.2**
10^8	10.8 ± 3.9	30.4 ± 6.9	17.1 ± 7.3	OOM	769.5 ± 27.9	43.7 ± 12.2	1.49e4	873.0	N/A	**25.3**	**2.6**
10^9	101.8 ± 36.6	290.0 ± 69.4	174.0 ± 76.6	OOM	OOM	517.4 ± 60.8	1.49e6	8583.0	N/A	N/A	**3.0**
Error	0.0005	0.0012	0.0008	0.0005	0.0002	0.0003	0.0				
Theoretical Complexity	$\mathcal{O}\left(n \cdot \log_2\left(\frac{D \cdot \xi^2}{\gamma^2 \cdot 4 \cdot \eta}\right)\right)$			$\mathcal{O}\left(n \log\left(n\right)\right)$			$\mathcal{O}\left(n^2\right)$				

Ablation study between F$^{2.5}$M and F^{3}M We provide and additional ablation study between F$^{2.5}$M and F^{3}M in Table 6. The results are quite similar to the comparison between F^{3}M and FFM(GPU). Here, we see that smooth field and adaptive far-field approximation (F^{3}M) both improve speed and also memory usage as smooth field helps approximate more interactions. We can thus infer empirically that the m_i^{smooth} term in Theorem 2 has a significant impact on reducing the memory footprint of interactions.

Table 6: Comparison between F^{3}M, F$^{2.5}$M and KeOps. It should be noted that KeOps is only run up to $n = 10^8$ for all experiments (a run for $n = 10^9$ would take weeks). The times for $n > 10^8$ are extrapolated for KeOps.

	F^{3}M time (s)			**F$^{2.5}$M time (s)**			**KeOps time (s)**	**Speedup vs KeOps**	**Speedup vs F$^{2.5}$M**		
n	OSM ($D=2$)	Taxi ($D=3$)	$D=3$	OSM ($D=2$)	Taxi ($D=3$)	$D=3$	$D=3$		OSM ($D=2$)	Taxi ($D=3$)	$D=3$
10^6	0.1 ± 0.0	0.5 ± 0.2	0.2 ± 0.1	1.4 ± 0.3	2.0 ± 0.3	0.3 ± 0.2	1.56	8.0	**14.0**	**4.0**	**1.5**
10^7	1.1 ± 0.4	3.3 ± 0.6	1.7 ± 0.7	11.3 ± 0.4	110.8 ± 2.0	4.5 ± 2.1	148.6	87.0	**10.3**	**33.6**	**2.6**
10^8	10.8 ± 3.9	30.4 ± 6.9	17.1 ± 7.3	OOM	777.4 ± 21.5	42.5 ± 13.1	1.49e4	873.0	N/A	**25.6**	**2.5**
10^9	101.8 ± 36.6	290.0 ± 69.4	174.0 ± 76.6	OOM	OOM	488.4 ± 55.3	1.49e6	8583.0	N/A	N/A	**2.8**
Error	0.0005	0.0012	0.0008	0.0004	0.0002	0.0016	0.0				
Theoretical Complexity	$\mathcal{O}\left(n \cdot \log_2\left(\frac{D \cdot \xi^2}{\gamma^2 \cdot 4 \cdot \eta}\right)\right)$			$\mathcal{O}\left(n \log\left(n\right)\right)$			$\mathcal{O}\left(n^2\right)$				

Gaussian process regression experiments We further compare F^{3}M as a drop-in KMVM operation applied to Black-box Matrix Multiplication [15] for Gaussian Processes, compared to KISS-GP [35], an approximate Gaussian process using cubic interpolation for kernel approximation. We mimic the setup in [32] and consider the datasets 3DRoad, Song, Buzz and House Electric, where we apply PCA to the last three datasets and take the 3 first principal components for a fair comparison against KISS-GP, which is limited by $D \leq 3$. We demonstrate the results in Table 7. As exact GP using F^{3}M demonstrates competitive results even when compared to SVGP [18] and SGPR [30], we hypothesize that many high-dimensional datasets conform to the *manifold hypothesis* [13], allowing F^{3}M to be widely applicable out-of-the-box even in high-dimensional settings.

Table 7: Gaussian process regression results. Exact GP using F^{3}M shows improved results and scaling compared to KISS-GP. SGPR and KISS-GP could not scale to the HouseElectric dataset.

			RMSE				**Training time (s)**			
Dataset	n	d	**Exact GP (F^{3}M)**	**KISS-GP**	**SGPR** ($m = 512$)	**SVGP** ($m = 1024$)	**Exact GP (F^{3}M)**	**KISS-GP**	**SGPR** ($m = 512$)	**SVGP** ($m = 1024$)
3DRoad	278,319	3	**0.297 ±0.036**	0.314 ±0.01	0.661 ± 0.010	0.481 ± 0.002	**27.8 ±18.0**	312.9 ±10.8	720.5 ± 330.4	2045.1 ± 191.4
Song	329,820	90	**0.369 ±0.029**	0.57 ±0.298	0.803 ± 0.002	0.998 ± 0.000	**7.2 ±3.1**	1705.2 ±115.6	473.3 ± 187.5	2373.3 ± 184.9
Buzz	373,280	77	0.967 ±0.002	0.997 ±0.05	**0.300± 0.004**	0.304 ± 0.012	**33.5 ±9.0**	542.7 ±0.8	1754.8 ± 1099.6	2780.8 ± 175.6
HouseEletric	1,311,539	9	0.308 ±0.006	OOM	OOM	**0.084 ± 0.005**	**79.8 ±23.1**	N/A	N/A	22062.6 ± 282.0

6 Limitations and Further Research

This work has introduced and implemented F^{3}M on GPU, which enables fast KMVM for tall and skinny data up to $n = 10^9$. F^{3}M has improved complexity which also is controllable through η, and retains linear memory. Experiments in higher dimensions also exhibit linear complexity, however requiring more nodes for lower errors. F^{3}M can further be directly used as a drop-in KMVM operation, as demonstrated with FALKON and Gaussian process regression, achieving significant speedups and competitive performance on both tasks. As an interpolation based approximation method, F^{3}M is still limited by the exponential growth of interpolation nodes with respect to D, although removing empty boxes, *small field* and sparse grids allow KMVM for $D \leq 7$. A fruitful direction would be to extend ideas in F^{3}M to accommodate higher-dimensional data by considering randomized partitioning [4], decoupling the dependency on D in geometry based partitioning. Further, an exact characterization of how $m_i^{\text{far}}, m_i^{\text{smooth}}, m_i^{\text{small}}, m_i^{\text{empty}}$ grows is left to future work.

Acknowledgments

The authors sincerely thank Lood van Niekerk and Jean-François Ton for their helpful comments.

References

[1] https://examples.pyviz.org/osm/osm-1billion.html. `https://examples.pyviz.org/osm/osm-1billion.html`. Accessed: 2021-05-19.

[2] https://www1.nyc.gov/site/tlc/about/tlc-trip-record-data.page. `https://www1.nyc.gov/site/tlc/about/tlc-trip-record-data.page`. Accessed: 2022-05-14.

[3] M. Aussal and M. Bakry. The fast and free memory method for the efficient computation of convolution kernels, 2019.

[4] A. Backurs, P. Indyk, C. Musco, and T. Wagner. Faster kernel matrix algebra via density estimation. In M. Meila and T. Zhang, editors, *Proceedings of the 38th International Conference on Machine Learning*, volume 139 of *Proceedings of Machine Learning Research*, pages 500–510. PMLR, 18–24 Jul 2021.

[5] J. H. Barnes and P. Hut. A hierarchical o(n log n) force-calculation algorithm. *Nature*, 324:446–449, 1986.

[6] J.-M. Belley, P. Belley, F. Colin, and R. Egli. Non-smooth kernels for meshfree methods in fluid dynamics. *Computers |& Mathematics with Applications*, 58(6):1253–1272, 2009.

[7] J.-P. Berrut and L. N. Trefethen. Barycentric lagrange interpolation. *SIAM Review*, 46(3):501–517, 2004.

[8] S. Börm, M. Lopez-Fernandez, and S. Sauter. Variable order, directional h2-matrices for helmholtz problems with complex frequency, 2019.

[9] D. Cai, E. Chow, Y. Saad, and Y. Xi. Smash: Structured matrix approximation by separation and hierarchy, 2017.

[10] J. Carrier, L. Greengard, and V. Rokhlin. A fast adaptive multipole algorithm for particle simulations. *SIAM J. Sci. Stat. Comput.*, 9(4):669–686, July 1988.

[11] B. Charlier, J. Feydy, J. A. Glaunès, F.-D. Collin, and G. Durif. Kernel operations on the GPU, with autodiff, without memory overflows. *arXiv preprint arXiv:2004.11127*, 2020.

[12] J. F. Epperson. On the runge example. *Am. Math. Monthly*, 94(4):329–341, Apr. 1987.

[13] C. Fefferman, S. Mitter, and H. Narayanan. Testing the manifold hypothesis. *Journal of the American Mathematical Society*, 29, 10 2013.

[14] C. for F3M. https://github.com/MrHuff/F3M.

[15] J. R. Gardner, G. Pleiss, D. Bindel, K. Q. Weinberger, and A. G. Wilson. Gpytorch: Blackbox matrix-matrix gaussian process inference with gpu acceleration, 2018.

[16] D. Garreau, W. Jitkrittum, and M. Kanagawa. Large sample analysis of the median heuristic, 2017.

[17] L. Greengard, M. O'Neil, M. Rachh, and F. Vico. Fast multipole methods for evaluation of layer potentials with locally-corrected quadratures, 2020.

[18] J. Hensman, N. Fusi, and N. D. Lawrence. Gaussian processes for big data. In *Proceedings of the Twenty-Ninth Conference on Uncertainty in Artificial Intelligence*, UAI'13, page 282–290, Arlington, Virginia, USA, 2013. AUAI Press.

[19] G. W. Howell. Derivative error bounds for lagrange interpolation: An extension of cauchy's bound for the error of lagrange interpolation. *Journal of Approximation Theory*, 67(2):164–173, 1991.

[20] W. Kang and L. C. Wilcox. Mitigating the curse of dimensionality: Sparse grid characteristics method for optimal feedback control and hjb equations, 2015.

[21] B. Kohnke, C. Kutzner, A. Beckmann, G. Lube, I. Kabadshow, H. Dachsel, and H. Grubmüller. A cuda fast multipole method with highly efficient m2l far field evaluation. *The International Journal of High Performance Computing Applications*, 35:109434202096485, 10 2020.

[22] G. K. Leaf and H. G. Kaper. l^{∞}-error bounds for multivariate lagrange approximation. *SIAM Journal on Numerical Analysis*, 11(2):363–381, 1974.

[23] D. Meagher. Octree encoding: A new technique for the representation, manipulation and display of arbitrary 3-d objects by computer. 10 1980.

[24] G. Meanti, L. Carratino, L. Rosasco, and A. Rudi. Kernel methods through the roof: Handling billions of points efficiently. In H. Larochelle, M. Ranzato, R. Hadsell, M. F. Balcan, and H. Lin, editors, *Advances in Neural Information Processing Systems*, volume 33, pages 14410–14422. Curran Associates, Inc., 2020.

[25] A. Paszke, S. Gross, F. Massa, A. Lerer, J. Bradbury, G. Chanan, T. Killeen, Z. Lin, N. Gimelshein, L. Antiga, A. Desmaison, A. Kopf, E. Yang, Z. DeVito, M. Raison, A. Tejani, S. Chilamkurthy, B. Steiner, L. Fang, J. Bai, and S. Chintala. Pytorch: An imperative style, high-performance deep learning library. In H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox, and R. Garnett, editors, *Advances in Neural Information Processing Systems 32*, pages 8024–8035. Curran Associates, Inc., 2019.

[26] A. Rudi, L. Carratino, and L. Rosasco. Falkon: An optimal large scale kernel method, 2018.

[27] B. Scholkopf and A. J. Smola. *Learning with Kernels: Support Vector Machines, Regularization, Optimization, and Beyond*. MIT Press, Cambridge, MA, USA, 2001.

[28] C. Schwab and W. L. Wendland. Kernel properties and representations of boundary integral operators. *Mathematische Nachrichten*, 156(1):187–218, 1992.

[29] S. A. Smolyak. Quadrature and interpolation formulas for tensor products of certain class of functions. *Dokl. Akad. Nauk SSSR*, 148(5):1042–1053, 1963. Transl.: Soviet Math. Dokl. 4:240-243, 1963.

[30] M. Titsias. Variational learning of inducing variables in sparse gaussian processes. In D. van Dyk and M. Welling, editors, *Proceedings of the Twelth International Conference on Artificial Intelligence and Statistics*, volume 5 of *Proceedings of Machine Learning Research*, pages 567–574, Hilton Clearwater Beach Resort, Clearwater Beach, Florida USA, 16–18 Apr 2009. PMLR.

[31] D. Tward, T. Brown, Y. Kageyama, J. Patel, Z. Hou, S. Mori, M. Albert, J. Troncoso, and M. Miller. Diffeomorphic registration with intensity transformation and missing data: Application to 3d digital pathology of alzheimer's disease. *Frontiers in Neuroscience*, 14, 2020.

[32] K. A. Wang, G. Pleiss, J. R. Gardner, S. Tyree, K. Q. Weinberger, and A. G. Wilson. Exact gaussian processes on a million data points, 2019.

[33] L. Wang, R. Krasny, and S. Tlupova. A kernel-independent treecode based on barycentric lagrange interpolation, 2019.

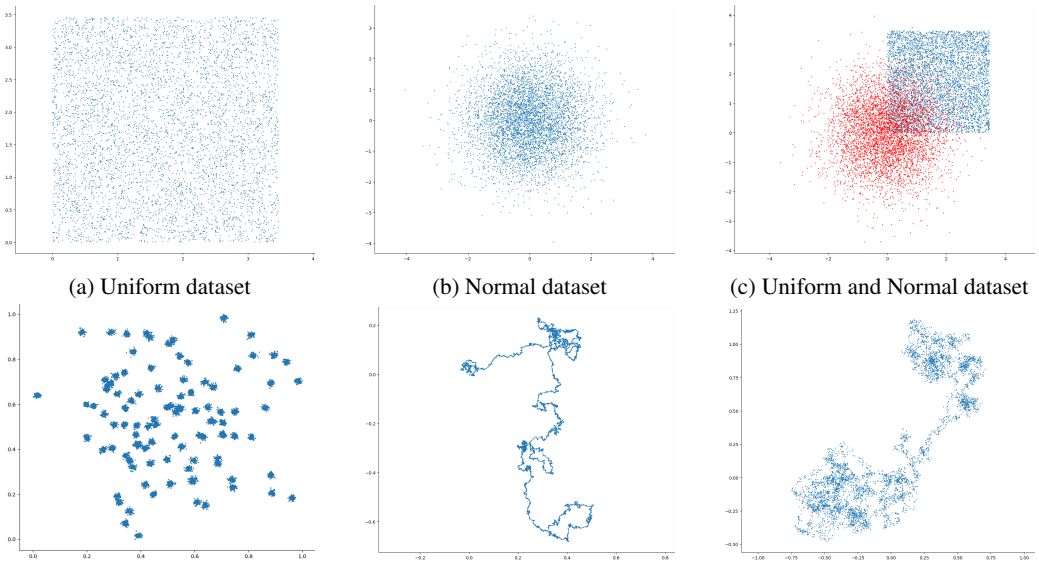
[34] A. Wilson and H. Nickisch. Kernel interpolation for scalable structured gaussian processes (kiss-gp). In F. Bach and D. Blei, editors, *Proceedings of the 32nd International Conference on Machine Learning*, volume 37 of *Proceedings of Machine Learning Research*, pages 1775–1784, Lille, France, 07–09 Jul 2015. PMLR.

[35] A. G. Wilson and H. Nickisch. Kernel interpolation for scalable structured gaussian processes (kiss-gp). In *Proceedings of the 32nd International Conference on International Conference on Machine Learning - Volume 37*, ICML'15, page 1775–1784. JMLR.org, 2015.

[36] Yang, Duraiswami, Gumerov, and Davis. Improved fast gauss transform and efficient kernel density estimation. In *Proceedings Ninth IEEE International Conference on Computer Vision*, pages 664–671 vol.1, 2003.

Checklist

1. For all authors...
   (a) Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope? [Yes]
   (b) Did you describe the limitations of your work? [Yes]
   (c) Did you discuss any potential negative societal impacts of your work? [Yes]
   (d) Have you read the ethics review guidelines and ensured that your paper conforms to them? [Yes]
2. If you are including theoretical results...
   (a) Did you state the full set of assumptions of all theoretical results? [Yes]
   (b) Did you include complete proofs of all theoretical results? [Yes] See Appendix B, D, E.
3. If you ran experiments...
   (a) Did you include the code, data, and instructions needed to reproduce the main experimental results (either in the supplemental material or as a URL)? [Yes]
   (b) Did you specify all the training details (e.g., data splits, hyperparameters, how they were chosen)? [Yes]
   (c) Did you report error bars (e.g., with respect to the random seed after running experiments multiple times)? [Yes]
   (d) Did you include the total amount of compute and the type of resources used (e.g., type of GPUs, internal cluster, or cloud provider)? [Yes]
4. If you are using existing assets (e.g., code, data, models) or curating/releasing new assets...
   (a) If your work uses existing assets, did you cite the creators? [N/A]
   (b) Did you mention the license of the assets? [N/A]
   (c) Did you include any new assets either in the supplemental material or as a URL? [N/A]
   (d) Did you discuss whether and how consent was obtained from people whose data you're using/curating? [N/A]
   (e) Did you discuss whether the data you are using/curating contains personally identifiable information or offensive content? [N/A]
5. If you used crowdsourcing or conducted research with human subjects...
   (a) Did you include the full text of instructions given to participants and screenshots, if applicable? [N/A]
   (b) Did you describe any potential participant risks, with links to Institutional Review Board (IRB) approvals, if applicable? [N/A]
   (c) Did you include the estimated hourly wage paid to participants and the total amount spent on participant compensation? [N/A]



(a) Uniform dataset (b) Normal dataset (c) Uniform and Normal dataset

(d) Clustered dataset (e) Fractional Brownian Motion (f) Brownian motion

Figure 10: 2D illustrations of the synthetic datasets.

A Synthetic data

Note on synthetic datasets We generated synthetic datasets of different types to measure the ability of F^{3}M to deal with dense or sparse data. Dense datasets were generated as independent samples with either uniform or normal distributions. Clustered datasets were generated by sampling cluster centers from a normal distribution, and then recursively sampling sub-cluster centers from a normal distribution with reduced standard deviation and centered at each cluster center, until the desired number of points is attained. Fractional Brownian Motion and Brownian Motion samples were generated as samplings of Fractional Brownian Motion paths with respective Hurst index 0.75 and 0.5. Figure 10 shows samples of each dataset type in the 2D case.

B Note on maximal variance on an interval

Proposition 4. *Consider a random variable $X \in \mathbb{R}$ with finite variance with $m = \inf X$ and $M = \sup X$. Then Var$(X) \leq \frac{(M-m)^2}{4}$.*

Proof. Define a function g by $g(t) = \mathbb{E}\left[(X-t)^2\right]$. Computing the derivative g', and solving $g'(t) = -2\mathbb{E}[X] + 2t = 0$ yields that g achieves its minimum at $t = \mathbb{E}[X]$ (note that $g'' > 0$). Now, consider the value of the function g at the special point $t = \frac{M+m}{2}$. It must be the case that $\mathrm{Var}[X] = g(\mathbb{E}[X]) \leq g\left(\frac{M+m}{2}\right)$. Evaluating yields the expression

$$g\left(\frac{M+m}{2}\right) =$$

$$\mathbb{E}\left[\left(X - \frac{M+m}{2}\right)^2\right] = \frac{1}{4}\mathbb{E}\left[((X-m)+(X-M))^2\right]$$

Since $X - m \geq 0$ and $X - M \leq 0$, we have

$$((X-m)+(X-M))^2 \leq ((X-m)-(X-M))^2 = (M-m)^2$$

implying that

$$\frac{1}{4}\mathbb{E}\left[((X-m)+(X-M))^2\right] \leq$$

$$\frac{1}{4}\mathbb{E}\left[((X-m)-(X-M))^2\right] = \frac{(M-m)^2}{4}$$

Hence

$$\mathrm{Var}[X] \leq \frac{(M-m)^2}{4}$$

□

C Details on barycentric Lagrange interpolation

The barycentric lagrange interpolation is written as

$$L_i(t) = \frac{\frac{w_i}{t-s_i}}{\sum_{i=0}^{r} \frac{w_i}{t-s_i}}, w_i = \frac{1}{\prod_{j=0, j\neq i}^{r} (s_i - s_j)}, i = 0, \ldots, n$$

where w_i are known as the barycentric weights. In case of singularities, i.e. when $t = s_j$, we set $L_i(s_j) = \delta_{ij}$. In particular, [7] proposes Chebyshev nodes of the second kind $s_i = \cos\theta_i, \quad \theta_i = \frac{i\pi}{r}, \quad i = 0, \ldots, r$. This choice of nodes combined with the scale invariance property of the barycentric form makes the calculation of w_i particularly easy

$$w_i = (-1)^i \delta_i, \quad \delta_i = \begin{cases} 1/2, & i = 0 \text{ or } i = r \\ 1, & i = 1, \ldots, r-1 \end{cases}$$

and reduces the complexity of calculating w_i from $\mathcal{O}(r^2)$ to $\mathcal{O}(r)$. The Lagrange interpolation polynomial can then be expressed as $p_r(t) = \sum_{i=0}^{r} \frac{\frac{w_i}{t-s_i}}{\sum_{i=0}^{r} \frac{w_i}{t-s_i}} f_i$.

D Smooth field proof

Proposition. *Consider* $\mathbf{x}, \mathbf{y} \in \mathcal{X} \subset \mathbb{R}^d$ *such that* $d(\mathbf{x}, \mathbf{y}) := \frac{\|\mathbf{x}-\mathbf{y}\|^2}{2\gamma^2} = \frac{1}{2\gamma^2} \sum_i^d (x^{(i)} - y^{(i)})^2 \leq \eta < 1$ *for all* $\mathbf{x}, \mathbf{y}$. *When interpolating* $k(\mathbf{x}, \mathbf{y}) = \exp(-d(\mathbf{x}, \mathbf{y}))$ *using bivariate Lagrange interpolation* $\mathcal{L}_r(\mathbf{x}, \mathbf{y}) := \mathbf{L}_X^T \cdot \mathbf{K} \cdot (\mathbf{L}_Y \cdot \mathbf{b})$ *with degree* $r = 2p$, *for any* $p \in \mathbb{N}_{>0}$ *there exist nodes* $\mathbf{s}^{\mathbf{x}}, \mathbf{s}^{\mathbf{y}}$ *for* $\mathcal{L}_r(\mathbf{x}, \mathbf{y})$ *such that the pointwise interpolation error is bounded by* $\mathcal{O}(\eta^{p+1})$.

Proof. Note that $k(\mathbf{x}, \mathbf{y})$ is analytic in $d(\mathbf{x}, \mathbf{y})$ with Taylor expansion given by $T_p e^{-d(\mathbf{x},\mathbf{y})} = 1 - d(\mathbf{x}, \mathbf{y}) + \ldots + \mathcal{O}\left(d(\mathbf{x}, \mathbf{y})^{p+1}\right)$. With $d(\mathbf{x}, \mathbf{y}) \leq \eta$, it follows that $|T_p k(\mathbf{x}, \mathbf{y}) - k(\mathbf{x}, \mathbf{y})| \leq \mathcal{O}(\eta^{p+1})$. Using triangle inequality, we have $|\mathcal{L}_r(\mathbf{x}, \mathbf{y}) - k(\mathbf{x}, \mathbf{y})| \leq |T_p k(\mathbf{x}, \mathbf{y}) - k(\mathbf{x}, \mathbf{y})| + |T_p k(\mathbf{x}, \mathbf{y}) - \mathcal{L}_r(\mathbf{x}, \mathbf{y})| \leq \eta^{p+1} + |T_p k(\mathbf{x}, \mathbf{y}) - \mathcal{L}_r(\mathbf{x}, \mathbf{y})|$. We note that $\mathcal{L}_r(\mathbf{x}, \mathbf{y})$ contains all the terms of the Taylor expansion, and we can thus choose the nodes $\mathbf{s}^{\mathbf{x}}, \mathbf{s}^{\mathbf{y}}$ of $\mathcal{L}_r(\mathbf{x}, \mathbf{y})$ such that $|T_p k(\mathbf{x}, \mathbf{y}) - \mathcal{L}_r(\mathbf{x}, \mathbf{y})| = 0$ as long as $r = 2p$, meaning the polynomial orders are matched. □

Note that the same proof strategy can be applied to any kernel k that admits a Taylor expansion.

E Complexity

Proposition. *A far-field interaction between two boxes containing* n_x *and* n_y *points respectively has time complexity* $\mathcal{O}(n)$, *where* $n = \max(n_x, n_y)$.

Proof. Far-field interactions are calculated as

$$\mathbf{v} \approx \underbrace{\mathbf{L}_X^T \cdot}_{\mathcal{O}(n_x \cdot r_X)} \Big(\underbrace{\mathbf{K} \cdot}_{\mathcal{O}(r_X \cdot r_Y)} \underbrace{(\mathbf{L}_Y \cdot \mathbf{b})}_{\mathcal{O}(n_y \cdot r_Y)} \Big). \tag{3}$$

As r_X, r_Y are independent of n_x, n_y, the complexity becomes $\mathcal{O}(n)$. Further see [3] for alternative proof. □

Proposition. *Given* n *data points in dimension* D, *the maximum number of divisions* $\mathit{Tree}_{\texttt{max divisions}}$ *is given by*

$$\mathit{Tree}_{\texttt{max divisions}} = \log_{2^D}(n). \tag{4}$$

Proof. To see this, simply solve for

$$\frac{n}{2^{D\cdot \text{Tree}_{\texttt{max divisions}}}} = 1 \implies \text{Tree}_{\texttt{max divisions}} = \log_{2^D}(n).$$

Further see [3] for alternative proof. □

Theorem. *Given a KMVM with edge $\mathcal{E}$ (dependent on data $\mathcal{X}, \mathcal{Y}$), lengthscale γ, effective variance limit η, n data points and data dimension D, F^{3}M has time complexity $\mathcal{O}(n \cdot \log_2\left(\frac{D\cdot\mathcal{E}^2}{\gamma^2\cdot 4\cdot\eta}\right))$, which can be taken as $\mathcal{O}(n \cdot \log_2\left(\frac{C}{\eta}\right))$ where $C \propto \frac{D\cdot\mathcal{E}^2}{\gamma^2}$.*

Proof. Recall that near-field interactions can be smoothly interpolated when $D \cdot \frac{\mathcal{E}^2}{\gamma^2\cdot 4\cdot 2^{\texttt{tree_depth}}} \leq \eta$. Then all interactions will be interpolated when $\texttt{tree_depth} \geq \log_2\left(\frac{D\cdot\mathcal{E}^2}{\gamma^2\cdot 4\cdot\eta}\right)$, which implies we can take $\text{Tree}_{\text{max divisions}} = \log_2\left(\frac{D\cdot\mathcal{E}^2}{\gamma^2\cdot 4\cdot\eta}\right)$. Hence the complexity is $\mathcal{O}(n \cdot \log_2\left(\frac{D\cdot\mathcal{E}^2}{\gamma^2\cdot 4\cdot\eta}\right))$. □

Theorem. *The number of interactions M_i against tree depth i of FFM and F^{3}M grows as $\mathcal{O}\left(M_{i-1}2^{2\cdot D} - m_i^{far}\right)$ and*

$$\mathcal{O}\left(M_{i-1}2^{2\cdot D} - (m_i^{empty})^2 - m_i^{far} - m_i^{smooth} - m_i^{small}\right)$$

respectively. Here $M_{-1} = \frac{1}{2^{2D}}$ and $m_0^{far} = m_0^{smooth} = m_0^{small} = m_0^{empty} = 0$ and $m_i^{far}, m_i^{smooth}, m_i^{small}, m_i^{empty}$ denotes the number of far-field, smooth field, small field interactions and the number of empty boxes respectively at depth $i > 0$. Note that for $i > 0$, these are dependent on data.

Proof. We prove through induction that the recursion holds for F^{3}M. We start with the base case $M_0 = 1$, since at depth $i = 0$, we only have one box and hence only one interaction. $M_0 = M_{-1}2^{2\cdot D} - (m_0^{\text{empty}})^2 - m_0^{\text{far}} - m_0^{\text{smooth}} - m_0^{\text{small}} = 1$. Clearly at depth 0, there can not be any empty boxes or possible approximations. For the induction step, $M_i = M_{i-1}2^{2\cdot D} - (m_i^{\text{empty}})^2 - m_i^{\text{far}} - m_i^{\text{smooth}} - m_i^{\text{small}}$. To get M_{i+1}, each box at depth i is first divided into 2^D, hence the number of interactions grows by $2^{2\cdot D}$. At depth $i+1$, we can further remove $(m_{i+1}^{\text{empty}})^2$ interactions between empty boxes and further compute $m_{i+1}^{\text{far}} + m_{i+1}^{\text{smooth}} + m_{i+1}^{\text{small}}$ interactions. Then $M_{i+1} = (M_{i-1}2^{2\cdot D} - (m_i^{\text{empty}})^2 - m_i^{\text{far}} - m_i^{\text{smooth}} - m_i^{\text{small}})2^{2\cdot D} - (m_{i+1}^{\text{empty}})^2 - m_{i+1}^{\text{far}} - m_{i+1}^{\text{smooth}} - m_{i+1}^{\text{small}} = M_i 2^{2\cdot D} - (m_{i+1}^{\text{empty}})^2 - m_{i+1}^{\text{far}} - m_{i+1}^{\text{smooth}} - m_{i+1}^{\text{small}}$. Thus the base case and induction step holds which completes our proof. This proof also covers FFM, since FFM can be as a special case for F^{3}M without removing empty boxes, smooth field and small field computation. □

F Algorithm summary

We present a summary of the algorithm presented in FFM in Algorithm 1 and the **modifications F^{3}M does in boldface**.

Algorithm 1: FFM ($\mathbf{F}^3\mathbf{M}$)

Input: Datasets $\mathbf{X}, \mathbf{Y}, \mathbf{b}$, kernel k, average points threshold ζ
Result: $\mathbf{v} = k(\mathbf{X}, \mathbf{Y}) \cdot \mathbf{b}$
Initialize treecodes $\tau_x = T(\mathbf{X}), \quad \tau_y = T(\mathbf{Y})$
Initialize near-field interactions as $\mathcal{I}_{\text{near}} = \{0, 0\}$
Initialize output $\mathbf{v} = \mathbf{0}$
while $|\mathcal{I}| > 0$ *and* $\texttt{MaximumBoxSize}(\tau_x) > \zeta$ *and* $\texttt{MaximumBoxSize}(\tau_y) > \zeta$ **do**
  Divide τ_x, τ_y
  Calculate interactions left $\mathcal{I} := f(\mathcal{I}_{\text{near}})$
  Partition $\mathcal{I}$ to $\{\mathcal{I}_{\text{near}}, \mathcal{I}_{\text{far}}, \boldsymbol{\mathcal{I}}_{\text{smooth}}\}$
  Throw away interactions that are too far from eachother
  Compute far-field interactions $\mathbf{v}$ += FarFieldCompute$(\tau_x, \tau_y, \mathcal{I}_{\text{far}})$
  Compute smooth field interactions $\mathbf{v}$ += FarFieldCompute$(\tau_x, \tau_y, \mathcal{I}_{\textbf{smooth}})$;
end
Compute remaining near-field interactions $\mathbf{v}$ += NearFieldCompute$(\tau_x, \tau_y, \mathcal{I}_{\text{near}})$

Instead of comparing the average box size to ζ we compare the maximum box size. When points are non-uniformly distributed, taking the maximum ensures that we don't compute near-field interactions on boxes with many points, since it will be inefficient.

G Scalability Analysis

We conduct a scalability analysis over $N_{\text{GPU}} = 1, 2, 4, 8$. We parallelize the KMVM product by considering the $k(\mathbf{X}, \mathbf{X})$-case and divide the work onto multiple GPUs by partitioning each subproduct of the KMVM (see for Figure 11 an example when $N_{\text{GPU}} = 8$). We take $\mathbf{X}$ to be Uniform and 3 dimensional. We present results in Figure 12.

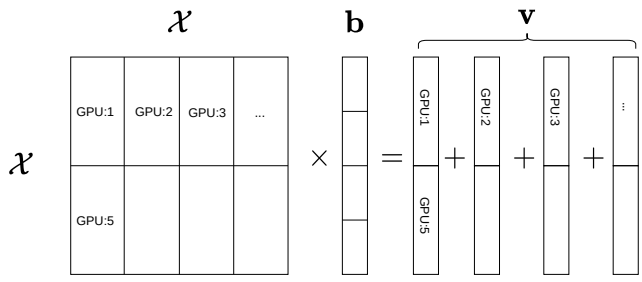


Figure 11: Partitioning a KMVM product into 8 jobs

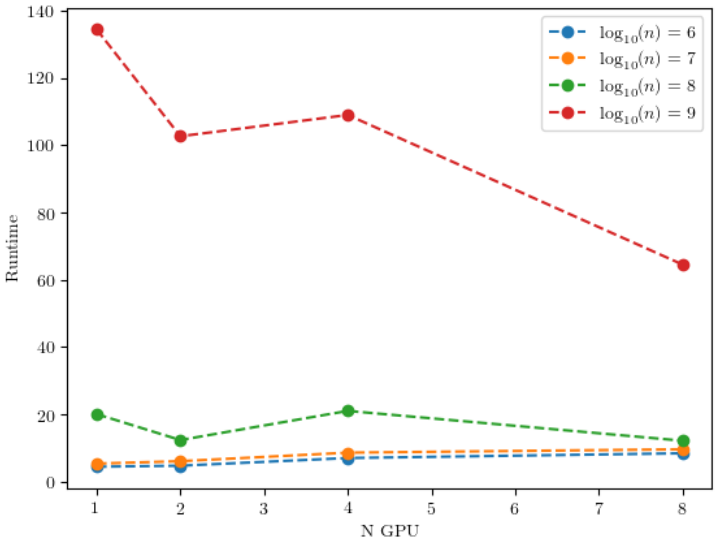


Figure 12: Since the V100 cards we use have very high throughput, we only get a performance boost when $n = 10^9$.

We also use `nvprof` to analyze the % of peak throughput of the V100 cards F^{3}M can utilize. We run `nvprof` for 3 dimensional uniform data for $n = 10^6, 10^7, 10^8, 10^9$. We present our results in Figure 13.

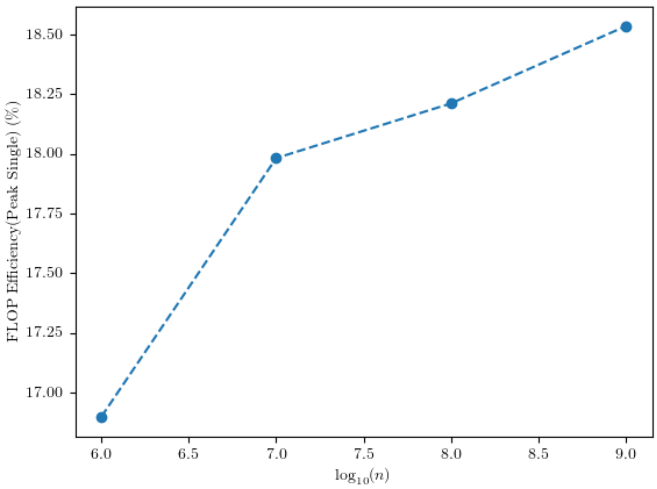


Figure 13: We used the `flop_sp_efficiency` metric in `nvprof` to generate the plot. One GPU was used for this experiment.

H Impact of η and r on performance

The performance of F^{3}M is tuned by choosing η and r to trade speed against accuracy. In Figure 14 we plot how different choices of η and r impacts computation time for F^{3}M on 3D data.

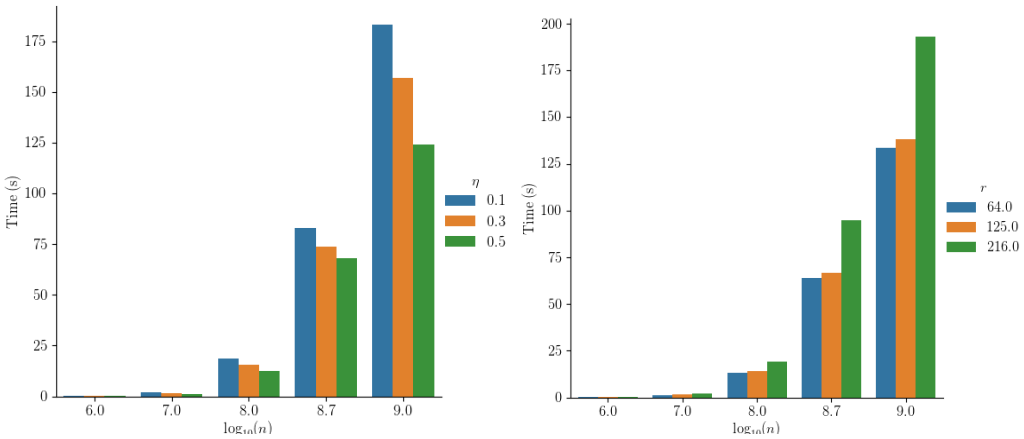


Figure 14: We see that larger η, more aggressive `smoothness criteria` and smaller number of interpolation nodes r improve speed.

I Implementation overview

We provide a skiss of how data is stored and used for F^{3}M in Figure 15.

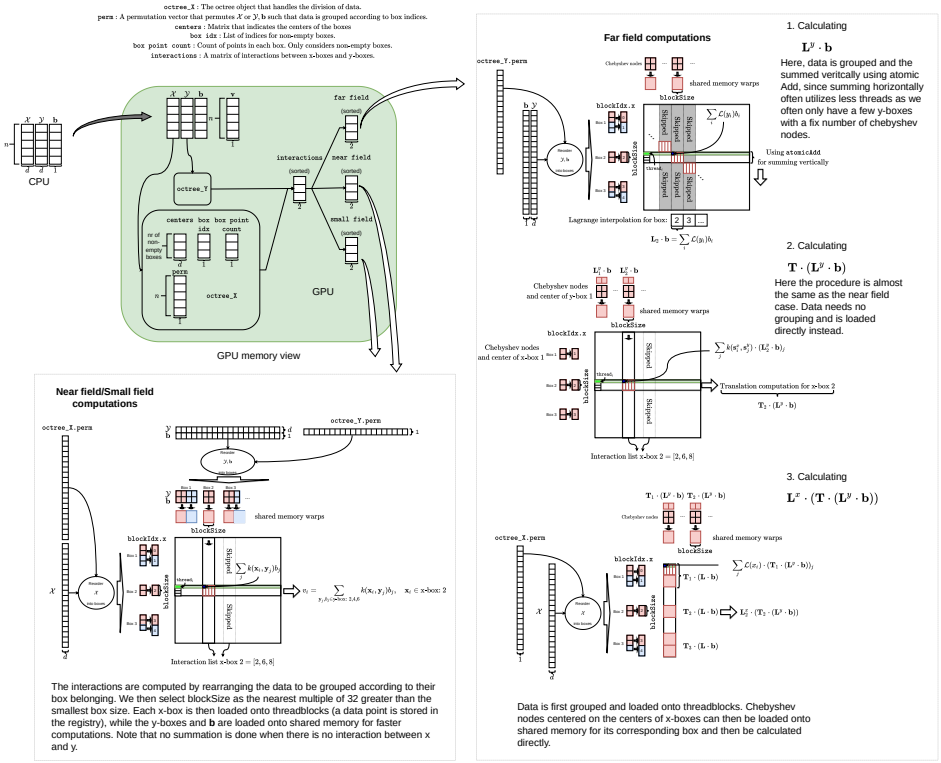


Figure 15: Skiss of how data is stored and used on GPU.

We provide an illustration on near field computations are carried out for F^{3}M in Figure 16.

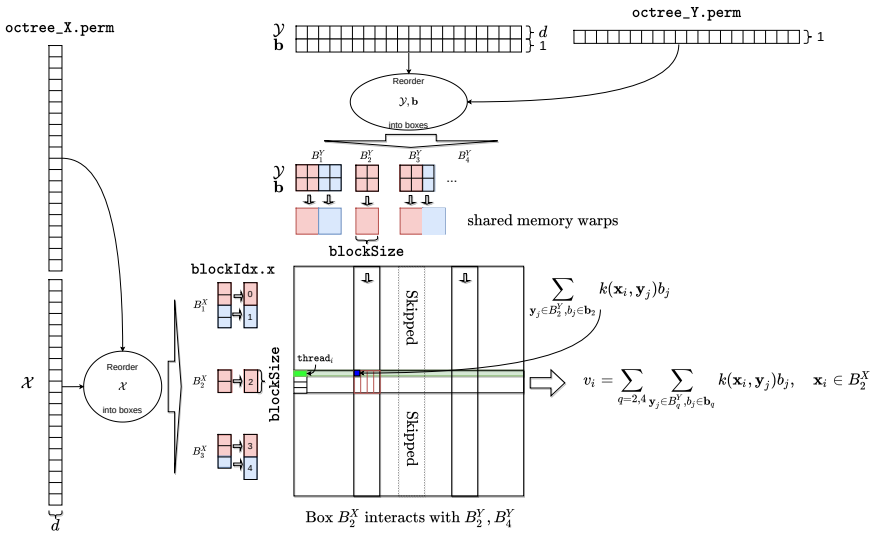


Figure 16: Illustration of how near-field interactions are computed in parallel on GPU. First data is reordered into their corresponding boxes using a permutation vector. Then each box is loaded in parallel into several thread blocks (indexed by `blockIdx.x`), wherein the challenge lies to execute this correctly. The computations are then parallelized across blocks, where only interactions are computed.

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	Giga-scale Kernel Matrix-Vector Multiplication on GPU
Publication Status	☐Published ☒ Accepted for Publication ☐Submitted for Publication ☐Unpublished and unsubmitted work written in a manuscript style
Publication Details	Hu, R. , Chau, S.L., Sejdinovic, D. and Glaunès, J.A. (2021) Giga-scale Kernel Matrix Vector Multiplication on GPU. To appear in the Thirty-sixth Conference on Neural Information Processing Systems (Neurips 2022)

Student Confirmation

Student Name:	Robert Hu		
Contribution to the Paper	I conceived methodology, implementation, experiments and did theory for the paper. Siu Lun Chau helped with polishing and presentation of the paper. During the meetings my supervisors helped out with helpful suggestions, feedback and even helped implement a few smaller functions. They also contributed a lot to the writing of the draft, checking the proofs, and proof-reading.		
Signature		Date	29/8-2022

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: Dino Sejdinovic			
Supervisor comments I am in agreement with the description of the contributions.			
Signature		Date	09/09/2022

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

7 Conclusion and Discussion

Contents

7.1 Contributions 197
7.2 Limitations and open questions 198
7.2.1 DIF 198
7.2.2 bd-HSIC 199
7.2.3 Kernel Fried Tensor 199
7.2.4 F^3M 200
7.3 Closing remarks 201

In this final chapter, we conclude the dissertation by summarizing the work presented together with outlining its limitations and potential directions for further study. Finally, we discuss the impact of our work and areas of application.

7.1 Contributions

In this thesis, we have presented different methods and models inspired by specific types of data and tasks encountered in an industrial context. We started with ideas from product development using image data, where we introduced DIF as an interpretable framework for binary image classification. Further, we leveraged kernel two-sample testing to ensure geometric separation of features in latent space corresponded to differences in image space, to understand feature importance and

effect. We then focused on hypothesis tests for causal association under high-dimensional confounding, introduced bd-HSIC and demonstrated that the proposed non-parametric test is calibrated and has good power on datasets with both linear and non-linear causal dependencies between treatment and outcome. Next, we considered forecasting problems and introduced KFT for modelling high-dimensional factor data in both a frequentist and Bayesian setting, demonstrating competitive performance against baselines and calibrated uncertainty quantification. Lastly, inspired by churn prediction, we proposed SuMo-net as a flexible survival regression framework on right-censored data with tractable likelihood and empirically showed that SuMo-net both achieves state-of-the-art likelihood scores on a variety of datasets and outperforms existing methods in terms of scalability. Finally, we scale kernel methods by proposing F^{3}M for KMVM applied to tall and skinny data to overcome the $\mathcal{O}(n^2)$ complexity and memory constraint of kernel methods. Besides strong empirical performance against KeOps [Charlier et al., 2021] and FFM [Aussal and Bakry, 2019], F^{3}M allows the user to compute a KMVM for $n = 10^9, D = 3$ in less than a minute on a single GPU with controllable error tolerance.

7.2 Limitations and open questions

In the beginner's mind there are many possibilities, but in the expert's there are few.

– Shunryu Suzuki [Suzuki, 2006]

As often in science, there is not a conclusive answer to which method or model is the best, but rather further pathologies to be discovered and improvements to be made. In this final section of this dissertation, we discuss the limitations of the proposed methods and possible extensions and questions that arise from the results.

7.2.1 DIF

Regression and multi-class tasks A natural extension of DIF would be to consider feature representation for multiple classes or regression. As DIF embeds class information into the latent variables, the extension retains this concept but

needs only to be extended beyond binary data, similar to [Joy et al., 2021] and [Yang et al., 2021]. A potential avenue would then be to embed multiclass labels into latent variables and ensure these variables retain a meaningful geometrical separation from each other.

7.2.2 bd-HSIC

Generalized kernel g-formula A limitation of bd-HSIC is that it only considers graphical models that satisfy the backdoor-criterion. A fruitful direction is to generalize bd-HSIC to handle a broader class of graphical models. This however requires careful formulation of the mean embeddings for the corresponding interventional distributions, which may require more advanced density ratio estimation than proposed in bd-HSIC.

7.2.3 Kernel Fried Tensor

Relationship with generic latent variable models KFT draws many parallels with latent variable models directly using side information proposed in Hu et al. [2017]. Here they propose latent variables passed through multiple weights and non-linear activations to predict a target $Y_{i,j} \in \mathbb{R}$. The proposed approach can be seen as a generalization of Deep matrix factorizations Xue et al. [2017], where latent variables corresponding to indices i, j are passed through a neural network architecture. Hu et al. [2017] can be considered a special parametrization of KFT when there only are two dimensions. The same parametrization can also be applied for each component of a KFT model, which acts as a general model for an arbitrary number of dimensions to factorize. Generally, for two dimensions KFT is likely to fall behind on predictive performance as it does not possess the same flexibility compared to latent variable models or deep matrix factorization models. However, for a higher number of factorization dimensions KFT is more viable as it allows a flexible and cheap way to model the multi-way interactions.

Uncertainty quantification While the variational inference model for KFT provides calibrated results, it adds additional parameters to the model and is somewhat dependent on the choice of the variational distribution. In particular, the multivariate normal choice of variational distribution can be particularly memory consuming. Conformal prediction [Shafer and Vovk, 2008] could be a viable alternative venue worth exploring for large scale uncertainty quantification.

Further computational optimization A current issue with KFT is that it inherently stores the entire tensor in memory during training, which implies the model size is limited by memory. To overcome this, efficient algorithms allowing for partial training of the tensor, meaning we load a part of the tensor on GPU (or any dedicated computing processor) to train at scale, must be developed to enable further scaling. As each part of the tensor can theoretically be trained independently, it would be both a practical and interesting challenge to consider KFT in a distributed setting.

7.2.4 F^3M

Higher dimensional data As F^3M targets tall and skinny data, a practical and natural extension is to consider the same divide and conquer strategy for higher dimensional data. However, this direction can be particularly challenging, as the partitioning and interpolation strategies previously proposed must be reworked to overcome the curse of dimensionality. A potential building block for this extension is [Adams et al., 2010], which proposes permutohedral lattices for kernel approximation which manages dimensionalities up to $D < 15$. While scaling further beyond dimensions of $D = 15$ is desirable, it is not immediately clear whether kernel interpolation strategies are even feasible for such data. A potential avenue is then to consider *Random Fourier Features* [Rahimi and Recht, 2008] for kernel approximation.

7.3 Closing remarks

Throughout this thesis, we introduced various machine learning methods inspired by industrial problems starting from product design, to marketing, and then into sales forecasting and going all the way to customer retention. Much of this can be seen as an entire life cycle of a business, where a product is conceived, marketed, sold and ultimately action is taken to retain existing customers. While the methods we proposed were inspired by this industrial setting, their application goes beyond this scope.

In Chapter 2, beyond analyzing popular garment features we further demonstrate that DIF also can be potentially used on x-ray images to understand what visual symptoms diseases might cause. For any binary classification task involving images, DIF can readily be used to extract important features. Next, in Chapter 3, bd-HSIC is a very general non-parametric test for any application that considers a graphical model satisfying the backdoor criteria. Examples in the paper include policy evaluation in economics. Further in Chapter 4, KFT can be used as a general prediction model for high-dimensional data, spanning applications areas from climate science [Kashinath et al., 2021] to medical imaging [Yaman et al., 2020]. SuMo-net in Chapter 5 is generally applicable as the enablement of a tractable right-censored likelihood in neural networks has wide applications ranging from epidemiology [Cole and Hudgens, 2010] to actuarial sciences [Barakat et al., 2019]. Finally, our proposal of F^3M in Chapter 6 has allowed KMVM to scale to data sets of $n = 10^9$ for skinny data. With KMVM scaling to such magnitudes, it potentially opens up kernel methods to be applicable to datasets beyond size $n = 10^9$, if combined with existing approximate methods such as FALKON [Rudi et al., 2017, Meanti et al., 2020].

It is our sincere hope that the open source codebase, together with this thesis, will become a valuable tool for any machine learning practitioner aiming to solve prediction and inference problems at scale.

Bibliography

A. Adams, J. Baek, and M. Davis. Fast high-dimensional filtering using the permutohedral lattice. *Comput. Graph. Forum*, 29:753–762, 05 2010. doi: 10.1111/j.1467-8659.2009.01645.x.

C. Aguilar-Palacios, S. Muñoz-Romero, and J. L. Rojo-Álvarez. Causal quantification of cannibalization during promotional sales in grocery retail. *IEEE Access*, 9: 34078–34089, 2021. doi: 10.1109/ACCESS.2021.3062222.

M. A. Aizerman. Theoretical foundations of the potential function method in pattern recognition learning. *Automation and Remote Control*, 25:821–837, 1964.

S. Ambikasaran, D. Foreman-Mackey, L. Greengard, D. W. Hogg, and M. O'Neil. Fast direct methods for gaussian processes, 2015.

L. Antolini, P. Boracchi, and E. Biganzoli. A time-dependent discrimination index for survival data. *Statistics in medicine*, 24(24):3927–3944, 2005.

N. Aronszajn. Theory of reproducing kernels. *Transactions of the American Mathematical Society*, 68(3):337–404, 1950. ISSN 00029947. URL `http://www.jstor.org/stable/1990404`.

M. Aussal and M. Bakry. The fast and free memory method for the efficient computation of convolution kernels, 2019.

A. Avati, T. Duan, S. Zhou, K. Jung, N. H. Shah, and A. Y. Ng. Countdown regression: sharp and calibrated survival predictions. In *Uncertainty in Artificial Intelligence*, pages 145–155. PMLR, 2020.

A. Barakat, A. Mittal, D. Ricketts, and B. A. Rogers. Understanding survival analysis: actuarial life tables and the kaplan–meier plot. *British Journal of Hospital Medicine*, 80(11):642–646, 2019. doi: 10.12968/hmed.2019.80.11.642. URL `https://doi.org/10.12968/hmed.2019.80.11.642`.

A. Belloni, V. Chernozhukov, and C. Hansen. Inference on Treatment Effects after Selection among High-Dimensional Controls. *The Review of Economic Studies*, 81(2):608–650, 04 2014. ISSN 0034-6527. doi: 10.1093/restud/rdt044. URL `https://doi.org/10.1093/restud/rdt044`.

A. Berlinet and C. Thomas-Agnan. *Reproducing Kernel Hilbert Spaces in Probability and Statistics*. Springer US, 2011. ISBN 9781441990969. URL `https://books.google.se/books?id=bX3TBwAAQBAJ`.

S. Bongers, P. Forré , J. Peters, and J. M. Mooij. Foundations of structural causal models with cycles and latent variables. *The Annals of Statistics*, 49(5), oct 2021. doi: 10.1214/21-aos2064. URL `https://doi.org/10.1214%2F21-aos2064`.

B. E. Boser, I. M. Guyon, and V. N. Vapnik. A training algorithm for optimal margin classifiers. In *Proceedings of the Fifth Annual Workshop on Computational Learning Theory*, COLT '92, page 144–152, New York, NY, USA, 1992. Association for Computing Machinery. ISBN 089791497X. doi: 10.1145/130385.130401. URL https://doi.org/10.1145/130385.130401.

S. Börm, M. Lopez-Fernandez, and S. Sauter. Variable order, directional h2-matrices for helmholtz problems with complex frequency, 2019.

J. Carrier, L. Greengard, and V. Rokhlin. A fast adaptive multipole algorithm for particle simulations. *SIAM J. Sci. Stat. Comput.*, 9(4):669–686, July 1988. ISSN 0196-5204. doi: 10.1137/0909044. URL https://doi.org/10.1137/0909044.

D. C. Castro, I. Walker, and B. Glocker. Causality matters in medical imaging. *Nature Communications*, 11(1), Jul 2020. ISSN 2041-1723. doi: 10.1038/s41467-020-17478-w. URL http://dx.doi.org/10.1038/s41467-020-17478-w.

B. Charlier, J. Feydy, J. A. Glaunès, F.-D. Collin, and G. Durif. Kernel operations on the gpu, with autodiff, without memory overflows. *Journal of Machine Learning Research*, 22(74):1–6, 2021. URL http://jmlr.org/papers/v22/20-275.html.

S. Cole and M. Hudgens. Survival analysis in infectious disease research: Describing events in time. *AIDS (London, England)*, 24:2423–31, 10 2010. doi: 10.1097/QAD.0b013e32833dd0ec.

D. R. Cox. Regression models and life-tables. *Journal of the Royal Statistical Society: Series B (Methodological)*, 34(2):187–202, 1972.

A. P. Dawid. Causal inference without counterfactuals. *Journal of the American Statistical Association*, 95(450):407–424, 2000. ISSN 01621459. URL http://www.jstor.org/stable/2669377.

L. H. Gilpin, D. Bau, B. Z. Yuan, A. Bajwa, M. Specter, and L. Kagal. Explaining explanations: An overview of interpretability of machine learning, 2018.

L. Greengard, M. O'Neil, M. Rachh, and F. Vico. Fast multipole methods for evaluation of layer potentials with locally-corrected quadratures, 2020.

A. Gretton, O. Bousquet, A. Smola, and B. Schölkopf. Measuring statistical dependence with hilbert-schmidt norms. In *Proceedings of the 16th International Conference on Algorithmic Learning Theory*, ALT'05, page 63–77, Berlin, Heidelberg, 2005. Springer-Verlag. ISBN 354029242X. doi: 10.1007/11564089_7. URL https://doi.org/10.1007/11564089_7.

F. L. Hitchcock. The expression of a tensor or a polyadic as a sum of products. *Journal of Mathematics and Physics*, 6(1-4):164–189, 1927. doi: https://doi.org/10.1002/sapm192761164. URL https://onlinelibrary.wiley.com/doi/abs/10.1002/sapm192761164.

C. Hu, P. Rai, and L. Carin. Deep generative models for relational data with side information. In *ICML*, 2017.

G. W. Imbens and J. D. Angrist. Identification and Estimation of Local Average Treatment Effects. *Econometrica*, 62(2):467–475, March 1994. URL https://ideas.repec.org/a/ecm/emetrp/v62y1994i2p467-75.html.

A. Jacot, F. Gabriel, and C. Hongler. Neural tangent kernel: Convergence and generalization in neural networks. In *Proceedings of the 32nd International Conference on Neural Information Processing Systems*, NIPS'18, page 8580–8589, Red Hook, NY, USA, 2018. Curran Associates Inc.

T. Joy, S. M. Schmon, P. H. S. Torr, N. Siddharth, and T. Rainforth. Capturing label characteristics in vaes. In *ICLR*, 2021.

Y. Juan, Y. Zhuang, W.-S. Chin, and C.-J. Lin. Field-aware factorization machines for ctr prediction. In *Proceedings of the 10th ACM Conference on Recommender Systems*, RecSys '16, page 43–50, New York, NY, USA, 2016. Association for Computing Machinery. ISBN 9781450340359. doi: 10.1145/2959100.2959134. URL https://doi.org/10.1145/2959100.2959134.

K. Kashinath, M. Mustafa, A. Albert, J.-L. Wu, C. Jiang, S. Esmaeilzadeh, K. Azizzadenesheli, R. Wang, A. Chattopadhyay, A. Singh, A. Manepalli, D. Chirila, R. Yu, R. Walters, B. White, H. Xiao, H. A. Tchelepi, P. Marcus, A. Anandkumar, P. Hassanzadeh, and n. Prabhat. Physics-informed machine learning: case studies for weather and climate modelling. *Philosophical Transactions of the Royal Society A: Mathematical, Physical and Engineering Sciences*, 379(2194):20200093, 2021. doi: 10.1098/rsta.2020.0093. URL https://royalsocietypublishing.org/doi/abs/10.1098/rsta.2020.0093.

J. L. Katzman, U. Shaham, A. Cloninger, J. Bates, T. Jiang, and Y. Kluger. Deepsurv: personalized treatment recommender system using a Cox proportional hazards deep neural network. *BMC Medical Research Methodology*, 18(1):24, 2018.

G. Ke, Q. Meng, T. Finley, T. Wang, W. Chen, W. Ma, Q. Ye, and T.-Y. Liu. Lightgbm: A highly efficient gradient boosting decision tree. *Advances in neural information processing systems*, 30:3146–3154, 2017.

H. Kvamme, Ø. Borgan, and I. Scheel. Time-to-event prediction with neural networks and Cox regression. *Journal of machine learning research*, 20(129):1–30, 2019.

S. L. Lauritzen. *Graphical Models*. Oxford University Press, 1996. ISBN 0-19-852219-3.

Y. LeCun, B. E. Boser, J. S. Denker, D. Henderson, R. E. Howard, W. E. Hubbard, and L. D. Jackel. Handwritten digit recognition with a back-propagation network. In *NIPS*, 1989.

C. Lee, W. R. Zame, J. Yoon, and M. van der Schaar. Deephit: A deep learning approach to survival analysis with competing risks. In *AAAI*, pages 2314–2321, 2018.

G. MacKenzie. The statistical analysis of failure time data. *The Statistician*, 31: 278–278, 1982.

G. Meanti, L. Carratino, L. Rosasco, and A. Rudi. Kernel methods through the roof: Handling billions of points efficiently. In H. Larochelle, M. Ranzato, R. Hadsell, M. Balcan, and H. Lin, editors, *Advances in Neural Information Processing Systems*, volume 33, pages 14410–14422. Curran Associates, Inc., 2020. URL https://proceedings.neurips.cc/paper/2020/file/a59afb1b7d82ec353921a55c579ee26d-Paper.pdf.

T. Moriyama and M. Kuwano. Causal inference for contemporaneous effects and its application to tourism product sales data. *Journal of Marketing Analytics*, 08 2021. doi: 10.1057/s41270-021-00130-x.

A. H. Nobari, M. F. Rashad, and F. Ahmed. Creativegan: Editing generative adversarial networks for creative design synthesis, 2021. URL https://arxiv.org/abs/2103.06242.

I. Oseledets. Tensor-train decomposition. *SIAM J. Scientific Computing*, 33: 2295–2317, 01 2011. doi: 10.1137/090752286.

J. Pearl. *Causality*. Cambridge University Press, Cambridge, UK, 2 edition, 2009a. ISBN 978-0-521-89560-6. doi: 10.1017/CBO9780511803161.

J. Pearl. *Causality*. Cambridge University Press, 2 edition, 2009b. doi: 10.1017/CBO9780511803161.

J. Peters, D. Janzing, and B. Schlkopf. *Elements of Causal Inference: Foundations and Learning Algorithms*. The MIT Press, 2017. ISBN 0262037319.

A. Rahimi and B. Recht. Weighted sums of random kitchen sinks: Replacing minimization with randomization in learning. In D. Koller, D. Schuurmans, Y. Bengio, and L. Bottou, editors, *Advances in Neural Information Processing Systems*, volume 21. Curran Associates, Inc., 2008. URL https://proceedings.neurips.cc/paper/2008/file/0efe32849d230d7f53049ddc4a4b0c60-Paper.pdf.

J. Ramsey, S. Hanson, C. Hanson, Y. Halchenko, R. Poldrack, and C. Glymour. Six problems for causal inference from fmri. *NeuroImage*, 49:1545–58, 09 2009. doi: 10.1016/j.neuroimage.2009.08.065.

C. E. Rasmussen and C. K. I. Williams. *Gaussian Processes for Machine Learning (Adaptive Computation and Machine Learning)*. The MIT Press, 2005. ISBN 026218253X.

J. M. Robins. A new approach to causal inference in mortality studies with a sustained exposure period—application to control of the healthy worker survivor effect. *Mathematical Modelling*, 7(9-12):1393–1512, 1986.

J. M. Robins, M. Ángel Hernán, and B. Brumback. Marginal structural models and causal inference in epidemiology. *Epidemiology*, 11:550–560, 2000.

K. J. Rothman and S. Greenland. Causation and causal inference in epidemiology. *American Journal of Public Health*, 95(S1):S144–S150, 2005. doi: 10.2105/AJPH.2004.059204. URL https://doi.org/10.2105/AJPH.2004.059204. PMID: 16030331.

D. B. Rubin. Causal inference using potential outcomes: Design, modeling, decisions. *Journal of the American Statistical Association*, 100(469):322–331, 2005. ISSN 01621459. URL http://www.jstor.org/stable/27590541.

A. Rudi, L. Carratino, and L. Rosasco. Falkon: An optimal large scale kernel method. In I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, editors, *Advances in Neural Information Processing Systems*, volume 30. Curran Associates, Inc., 2017. URL https://proceedings.neurips.cc/paper/2017/file/05546b0e38ab9175cd905eebcc6ebb76-Paper.pdf.

C. Rudin. Stop explaining black box machine learning models for high stakes decisions and use interpretable models instead. *Nature Machine Intelligence*, 1: 206–215, May 2019.

C. Saunders, A. Gammerman, and V. Vovk. Ridge regression learning algorithm in dual variables. In *ICML*, 1998.

R. Scheines. An introduction to causal inference. In *Causality in Crisis? University of Notre Dame*, pages 185–200. Press, 1997.

G. Shafer and V. Vovk. A tutorial on conformal prediction. *J. Mach. Learn. Res.*, 9:371–421, jun 2008. ISSN 1532-4435.

I. Steinwart and A. Christmann. *Support Vector Machines*. Springer Publishing Company, Incorporated, 1st edition, 2008. ISBN 0387772413.

G. A. Susto, A. Schirru, S. Pampuri, S. McLoone, and A. Beghi. Machine learning for predictive maintenance: A multiple classifier approach. *IEEE Transactions on Industrial Informatics*, 11(3):812–820, 2015.

S. Suzuki. Zen mind, beginner's mind, 2006.

L. R. Tucker. Some mathematical notes on three-mode factor analysis. *Psychometrika*, 31(3):279–311, Sep 1966. ISSN 1860-0980. doi: 10.1007/BF02289464. URL `https://doi.org/10.1007/BF02289464`.

H.-J. Xue, X.-Y. Dai, J. Zhang, S. Huang, and J. Chen. Deep matrix factorization models for recommender systems. In *Proceedings of the 26th International Joint Conference on Artificial Intelligence*, IJCAI'17, page 3203–3209. AAAI Press, 2017. ISBN 9780999241103.

B. Yaman, S. Weingärtner, N. Kargas, N. D. Sidiropoulos, and M. Akçakaya. Low-rank tensor models for improved multidimensional mri: Application to dynamic cardiac t_1 mapping. *IEEE Transactions on Computational Imaging*, 6:194–207, 2020. doi: 10.1109/TCI.2019.2940916.

M. Yang, F. Liu, Z. Chen, X. Shen, J. Hao, and J. Wang. Causalvae: Disentangled representation learning via neural structural causal models. *2021 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 9588–9597, 2021.

M. Ziehm and J. Thornton. Unlocking the potential of survival data for model organisms through a new database and online analysis platform: Survcurv. *Aging cell*, 12, 07 2013. doi: 10.1111/acel.12121.