Doctoral Dissertation

Integrating Kernel Conditional Mean Embeddings with Deep Learning

Eiki Shimizu

The Graduate University for Advanced Studies,
SOKENDAI

ABSTRACT

Conditional density estimation is a foundational task in reliable decision making and probabilistic inference. The main focus of this thesis is Kernel Conditional Mean Embeddings (CMEs), which provide a framework for representing conditional distributions in reproducing kernel Hilbert spaces. CMEs inherit the powerful properties of positive definite kernels, for instance, allowing empirical quantities and operations on distributions to be computed via linear algebraic methods. Yet, their practical adoption in large-scale, high-dimensional applications has been limited due to challenges in scalability, restricted expressiveness, and the difficulty of tuning hyperparameters especially for the output-variable kernel.

This thesis aims to advance the existing CME framework by integrating deep learning, and proposes a novel approach called Neural-Kernel Conditional Mean Embeddings (NK-CMEs). It also explores strategies for optimizing hyperparameters of the output-variable kernel and extends into challenging applications where classical CMEs may not be well-suited but the proposed method can excel.

At its core, NK-CMEs replace the expensive Gram-matrix inversion in classical CME estimation with a deep learning model, while retaining the kernel for the output variable. This hybrid architecture combines the representational power and optimization efficiency of modern deep neural networks (DNNs) with the appealing properties of positive definite kernels for modeling conditional distributions. To address the challenge of selecting the output kernel hyperparameters, which renders standard cross-validation inapplicable, two strategies are introduced: (1) an iterative optimization method employing a squared error supplementary loss, and (2) a joint optimization scheme that simultaneously learns DNN parameters and kernel bandwidths using a theoretically grounded upper-bound formulation.

Empirical evaluations demonstrate the effectiveness of NK-CMEs in both synthetic and real-world conditional density estimation tasks. Together with these hyperparameter optimization strategies, the proposed method achieves competitive, and often superior, performance compared to alternatives such as methods based on deep feature representation, mixture density networks, normalizing flows, and even diffusion models.

Beyond standard density estimation, this thesis considers applying NK-CMEs to distributional reinforcement learning (RL): by modeling distributions of discounted cumulative rewards within the NK-CME framework, a new distributional Q-learning method naturally emerges. In classic control benchmarks, the proposed method consistently outperforms existing baselines and sheds light on how

the choice of distribution parameterization and loss function impacts learning in distributional RL.

ACKNOWLEDGEMENTS

First and foremost, I would like to thank my supervisor, Prof. Kenji Fukumizu, for his continuous support and encouragement. I began my journey as a technical staff member upon returning from London after completing masters, uncertain whether I wanted to pursue research. Meetings and discussions with Kenji convinced me to continue, and I chose to start my PhD under his guidance. Although it took time, the work we initiated together was ultimately published at a top conference—an achievement I owe to his steadfast support.

I would also like to thank Prof. Dino Sejdinovic for his invaluable contributions as a co-author. Without his insightful feedback on writing and rebuttals, the paper would not have reached its current form.

During my PhD, I was fortunate to work with many other reseachers. Particularity, I would like to thank Dr. Paavo Parmas for mentoring me during summer internship and submitting a paper together. I would also like to thank Prof. Emtiyaz Khan for hiring me as part-time researcher at Riken, and for his valuable advice.

My gratitude extends to all members of Fukumizu Lab, especially Itsumi-san and Akatsuka-san, for their help with many paper works and supports for business trips.

Finally, I am deeply grateful to my parents for their unwavering support throughout my entire PhD journey.

Contents

List of Tables

List of Figures

Chapter 1

Introduction

1 Motivations

When conditional distributions $P(Y \mid X)$ exhibit complex patterns—such as multimodality, skewness, heavy tails, or heteroscedastic noise—modeling only the conditional mean, as in standard regression, is insufficient. A rich representation of $P(Y \mid X)$ can reveal multiple likely outcomes or input dependent variability that a single point estimate would completely miss.

For example, consider autonomous driving at a Y-junction: if the car must predict a steering angle when the lane splits, there are two equally valid choices. A model that captures the full conditional density can represent both peaks—left and right turns—provided such scenarios are well represented in the training data. By contrast, a mean-based predictor would steer straight into the barrier between lanes, essentially "crashing" between the true possibilities. This example illustrates why, in many real-world problems, we need more than a mean prediction: we need the full conditional distribution for reliable decision making.

In fact, complex conditional distributions arise across many fields. For instance in social science, it is common to encounter heteroscedastic noise that varies with input variables. Likewise, in real-world scientific systems a single observation may correspond to multiple underlying causes—inverse problems with such one-to-many mappings give rise to multimodality.

We list existing applications where the representation of conditional distributions is essential.

- In finance, risk management often hinges on forecasting the full probability distribution of future returns to quantify value-at-risk and expected shortfall. Widely used models such as GARCH (Engle, 1982) provide a parametric form for the conditional return density, typically assuming time-varying volatility.

- Robotic systems frequently face one-to-many mappings. A classic example is robot arm inverse kinematics (Bishop, 1994): determining joint angles from a desired end effector position. Such inverse problems are inherently multivalued, since the arm can reach the same pose with multiple configurations. Similarly, transition or trajectory predictions in stochastic control can exhibit multimodality.

- In cosmology, galaxy redshifts are typically estimated from photometric inputs, like broadband magnitudes and colors, instead of spectroscopy (Dalmasso et al., 2019). Because different combinations of galaxy type, metallicity, and dust extinction can produce similar photometric signatures, this mapping is often multimodal.

Finally, conditional distributions appear in many of the probabilistic inference methods and tasks (Bishop, 2006), such as Bayesian inference, filtering and smoothing in state-space models, message passing in graphical models. Thus estimation of $P(Y|X)$ is also essential for inference and reasoning in complex systems.

2 Literature on Conditional Density Estimation Methods

Ideally, a method should capture multiple types of complex patterns, such as multimodality and heteroscedasticity, within a single framework. Nonparametric approaches provide flexible estimations of conditional distributions without assuming a fixed parametric form, and there has been extensive work in this area. Notably, one of the most popular and well-studied methods is based on the Kernel Density Estimator (KDE) (Rosenblatt, 1969). This method first estimates the joint distribution $P(X,Y)$ and the marginal distribution $P(X)$ by smoothing observed data points with a kernel function, and then computes $P(Y \mid X) = \frac{P(X,Y)}{P(X)}$. Note that this "kernel" in KDE refers to a smoothing window and should not be confused with the positive definite kernels used in reproducing kernel Hilbert spaces.

A related approach is localization methods (Fan et al., 1996), where local polynomial estimation is applied to handle varying smoothness and heteroscedasticity across the input domain.

Other popular approaches use mixture-of-experts models to represent conditional densities as weighted combinations of local regressors (Jacobs et al., 1991) or distributions (Bishop, 1994). Some methods employ tree-based techniques that partition the input space and fit simple densities within each leaf (Meinshausen, 2006). Another line of work directly estimates the density ratio $\frac{P(X,Y)}{P(X)}$ (Sugiyama et al., 2010) using classifiers or least-squares criteria. More recently, deep learning-based techniques like normalizing flows (Papamakarios et al., 2021; Rezende & Mohamed, 2015) and diffusion models (Ho et al., 2020) have been adapted for conditional density estimation, demonstrating strong performance in tasks such as simulation based inference (Gloeckler et al., 2024; Lueckmann et al., 2021).

The focus of this thesis, Kernel Conditional Mean Embeddings also provide nonparametric framework for representing conditional distributions. We provide brief review in the next section.

3 Representing Conditional Distributions with Positive Definite Kernels

Kernel Conditional Mean Embeddings (CMEs) provide nonparametric framework for representing conditional distributions. Introduced by Song et al. (2009) and reviewed in detail by Muandet et al. (2017), CMEs represent the conditional distribution $P(Y \mid X)$ as a mean element of reproducing kernel Hilbert Space (RKHS) $\mathscr{H}_Y$ associated with a positive definite kernel. Concretely, for random variables $X \in \mathscr{X}$ and $Y \in \mathscr{Y}$, one embeds the distribution $P(Y \mid X = \cdot)$ as a mapping $\mu_{P(Y|X)}(\cdot) \in \mathscr{H}_Y$. This is done by projecting inputs X and outputs Y into potentially infinite-dimensional feature spaces via kernels and then computing a conditional expectation in that space. Intuitively, CME estimation can be viewed as performing regression in the RKHS (see Figure 1.1). The detailed formulations are provided in Chapter 2.

Similarly to other nonparametric methods, CMEs make minimal assumptions about the form of $P(Y \mid X)$ and instead leverage the rich function class of the RKHS to capture complex distributional features. In addition, CMEs offer several

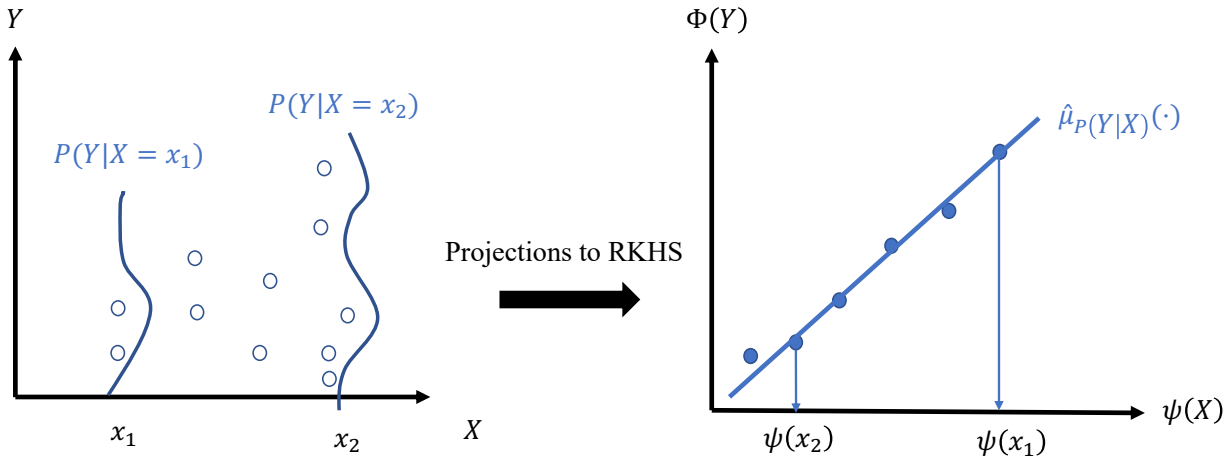

Figure 1.1: Illustration of Kernel Conditional Mean Embeddings.

advantages over alternative conditional density estimation methods by directly leveraging the properties of positive-definite kernels.

We summarize advantages of CMEs:

- CMEs do not directly estimate a density function as in KDE—in particular, the estimated conditional embedding need not be normalized or nonnegative. This flexibility helps CME-based methods handle high-dimensional conditional distributions more gracefully than KDE-based approaches, which are known to suffer severely from the curse of dimensionality. For instance, Fukumizu et al. (2013) demonstrate an empirical advantage of CMEs over KDE in a Bayesian-inference task. From a theoretical perspective, several works have shown optimality of CME estimators in terms of convergence rates. (Li et al., 2022; Talwai et al., 2022).

- Basic operations on distributions can be performed analytically. For example, for any $g \in \mathscr{H}_Y$,

$$\mathbb{E}\big[g(Y) \mid X = x\big] \;=\; \big\langle g, \mu_{P(Y|X)}(x) \big\rangle_{\mathscr{H}_Y},$$

  where, as we will see in Chapter 2, the empirical estimate reduces to simple linear-algebraic operations. Moreover, both sum rule (marginalization) $P(X) = \mathbb{E}_Y\big[P(X,Y)\big]$ and the product rule $P(X,Y) = P(Y \mid X)\,P(X)$ can be computed analytically (Song et al., 2013). These analytic operations make

CMEs a natural fit for a wide range of inference tasks: probabilistic inference including Bayesian updates, filtering and smoothing in state-space models, belief propagation in graphical models (Fukumizu et al., 2013; Song et al., 2013), and for causal inference problems, including causal discovery (Z. Chen et al., 2014), treatment effect estimation (Muandet et al., 2021), and inference with causal graphs (Chau et al., 2021; Singh et al., 2019; Xu & Gretton, 2023, 2025).

- By representing conditional distributions as mean elements in an RKHS, one can treat each $\mu_{P(Y|X)}(x)$ as a point in $\mathscr{H}_Y$, and measure distances via the RKHS norm between $P(Y \mid X = x)$ and $P(Y \mid X = x')$. All of these computations admit closed-form solutions, yielding powerful two sample tests (Gretton et al., 2012) and conditional independence tests (Gretton et al., 2005).

- By using specialized kernels, the method can be applied directly to non-vectorial data, such as graphs and text (Gärtner, 2003; Schölkopf & Smola, 2001).

4 Limitations of Kernel Conditional Mean Embeddings

Despite aforementioned strengths, the use of traditional CMEs in large-scale, high-dimensional settings have been limited due to challenges in **scalability**, **expressiveness**, and **hyperparameter selection**. We summarize each components below, and further details are discuss in Chapter 3.

1. **Scalability**

   Standard CME estimation requires solving a linear system involving an $n \times n$ Gram matrix, where n is the number of training samples. Computing and inverting this kernel matrix is an $\mathscr{O}(n^3)$ operation, and storing it takes $\mathscr{O}(n^2)$ memory—both of which become prohibitive as n grows. Kernel approximation techniques like random Fourier features (Rahimi & Recht, 2007) or Nyström methods (Drineas & Mahoney, 2005) can reduce these costs, but they often sacrifice accuracy.

2. **Expressiveness**

In the classic kernel-based CME, the feature map $\psi(x)$ is fixed by the choice of kernel (Gaussian RBF, polynomial, etc.). For high-dimensional or highly nonlinear cases, these predefined features may fail to capture essential structure, leading to suboptimal performance. In contrast, deep neural networks learn task-specific features and can adapt their representations to complex data.

3. **Hyperparameter selection**

   Selecting the hyperparameter of the output kernel $k_{\mathscr{Y}}$ (e.g. its bandwidth or shape parameters) is non-trivial because the CME objective itself depends on the RKHS norm induced by $k_{\mathscr{Y}}$. Changing the kernel parameters thus alters the very objective function being optimized, making standard model-selection tools like cross-validation inapplicable. In practice, one resorts to tricks like the median heuristics for Gaussian kernel bandwidth, or expensive trial-and-error search for these hyperparameters, neither of which is fully satisfactory.

To summarize, in its current form the CME framework is not suited for modern machine learning tasks, which are increasingly large-scale and high-dimensional. This is in contrast with standard deep learning models, which typically scale efficiently and handle high-dimensional data, making them the de facto choice for many applications.

5 Research Questions and Contributions

The aim of this thesis is to extend CMEs for greater effectiveness in challenging machine learning tasks. To achieve this, we must overcome the limitations of conventional CMEs and advance the existing framework. To be more specific, we pose the following research questions:

1. Is it possible to combine the scalability and expressiveness of deep learning with the desirable properties of CMEs to manipulate distributions?
2. What efficient strategies can be devised for optimizing the hyperparameters of the output-variable kernel?
3. Can CMEs be extended to applications where classical CMEs are not suited but the proposed method can excel?

For the first question, we introduce Neural-Kernel Conditional Mean Embeddings (NK-CMEs), a methodology that **integrates deep learning with kernel CMEs**. The central idea is to replace the costly Gram-matrix inversion with a trainable deep neural network (DNN), while preserving the output-variable kernel and thus the representation of conditional distributions in the RKHS $\mathscr{H}_\mathscr{Y}$. Specifically, we propose a DNN-based approach that directly learns the input-dependent CME weights $\boldsymbol{\beta}(x) = (K_X + \lambda I)^{-1}\boldsymbol{k}_X)$ via end-to-end training with a kernel-based loss. This eliminates explicit matrix inversion that the kernel solution would otherwise compute analytically. At the same time, the use of an RKHS-based loss, derived from the CME objective, ensures that the learned model retains the nonparametric flexibility and statistical grounding of kernel mean embeddings.

For the second question, we introduce a novel strategy for tuning the output-kernel hyperparameter: we define a specialized "Gaussian density kernel" and derive a *supplemental objective* that lets us learn its bandwidth alongside the DNN. We present two procedures:

1. An **iterative two-step** approach, alternating updates of NN parameters and kernel bandwidth.
2. A **fully joint** optimization of both parameter sets under a single upper-bound objective.

By learning the kernel parameter together with the DNN, we ensure the output feature space is well-suited to the data. This not only address the non-trivial hyperparameter selection of the output kernel, but also provides computationally efficient algorithm. The effectiveness of NK-CME, combined with these hyperparameter optimization strategies, is demonstrated on both synthetic and real-world datasets. We show that our method achieves competitive, and often superior, performance compared to alternative approaches, including diffusion model-based methods.

For the third question, we demonstrate that NK-CMEs enable new applications of CMEs in distributional reinforcement learning (RL). In distributional RL, one models the full distribution of discounted cumulative returns rather than just its mean. CMEs are especially suitable because they not only represent conditional distributions but also allow construction of a loss based on a conditional Maximum Mean Discrepancy (MMD) (Gretton et al., 2012). By incorporating NK-CME as a drop-in distributional estimator within a Q-learning framework, we naturally arrive at a novel distributional RL method. Importantly, NK-CME parameterization excels by bypassing the need for matrix inversion, enabling efficient action selection. In contrast, applying existing CME in this setting would necessitate matrix

inversion on the entire replay buffer, which can be prohibitively expensive, especially when performed repeatedly. Our approach also shares conceptual links with popular categorical DQN (Bellemare et al., 2017) and MMD-based distributional Q-networks (Nguyen-Tang et al., 2021), while benefiting from the principled CME foundation. Empirical results on several benchmark environments show consistent performance improvements, highlighting the versatility of NK-CME.

Contributions. Finally, we summarize the main contributions of this thesis. This thesis is based on the following conference paper (Shimizu et al., 2024):

Eiki Shimizu, Kenji Fukumizu and Dino Sejdinovic. Neural-Kernel Conditional Mean Embeddings. In *Proceedings of the 41st International Conference on Machine Learning*, pages 45040–45059, volume 235. Proceedings of Machine Learning Research, 2024.

- We propose a DNN-based CME framework that addresses the major limitations of traditional kernel CMEs. By training a DNN with a conditional kernel mean embedding loss, our approach circumvents expensive matrix inversions and leverages learned feature representations to handle high-dimensional, highly nonlinear data. This significantly improves the scalability and expressiveness of classical CMEs.

- We introduce a solution to the challenge of tuning hyperparameters of the output kernel. In particular, we define a Gaussian density kernel for the output space and derive a supplemental objective loss function that can be used for iterative optimization of the kernel bandwidth and the DNN. We also provide theoretical result that the supplemental objective provides an upper bound on the primary RKHS-based loss, and we develop an efficient algorithm to jointly train the DNN parameters and the kernel hyperparameter.

- We develop a distributional RL method that integrates our NK-CME into the deep Q-learning paradigm. This method inherits advantages from both deep distributional RL and CMEs, and we demonstrate its effectiveness on multiple control benchmark tasks.

6 Outline

The thesis is organized as follows: Chapter 2 introduces necessary backgrounds, including kernel methods in general, kernel mean embeddings, CMEs, deep learning and NN-parameterized deep feature approach. Chapter 3 discusses three limitations of classical CMEs in details. Chapter 4 presents our NK-CME approach and two hyperparameter optimization strategies. Chapter 5 demonstrates the performance of our method on both synthetic and real-world datasets. Chapter 6 delves into RL, introducing our new distributional RL method and assessing its performance in three classic control environments. Chapter 7 summarizes the thesis and discuss limitations and future works.

Chapter 2

Background

In this chapter, we provide the necessary background on kernel methods and deep learning to contextualize this thesis. We begin with a brief overview of positive definite kernels, reproducing kernel Hilbert space, and the concept of kernel mean embeddings for probability distributions. We then discuss kernel *conditional* mean embeddings in detail, including their formulation and computation. Next, we give a brief introduction to deep learning, covering their mathematical structure and standard training procedures. Finally, we discuss existing kernel mean embedding method that leverages features learned by deep learning models. In this thesis, we use the following notations:

We consider random variables X and Y, residing in domains $\mathscr{X} \subseteq \mathbb{R}^{d_x}$ and $\mathscr{Y} \subseteq \mathbb{R}^{d_y}$, respectively, with realizations x and y, joint distribution P, and density function $p(x,y)$. Measurable positive definite kernels $k_{\mathscr{X}} : \mathscr{X} \times \mathscr{X} \to \mathbb{R}$ and $k_{\mathscr{Y}} : \mathscr{Y} \times \mathscr{Y} \to \mathbb{R}$, associated with RKHSs $\mathscr{H}_{\mathscr{X}}$ and $\mathscr{H}_{\mathscr{Y}}$, induce features $\psi(x) = k_{\mathscr{X}}(x,\cdot)$ and $\phi(y) = k_{\mathscr{Y}}(y,\cdot)$. The inner product is denoted by $\langle \cdot,\cdot \rangle_{\mathscr{H}}$ and the RKHS norm by $\| \cdot \|_{\mathscr{H}}$.

1 Positive Definite Kernels and Reproducing Kernel Hilbert Spaces

This section reviews the mathematical foundations of kernel-based methods used later in the thesis. We first give formal definitions, then state classical theorems and finally survey widely used kernel families.

1.1 Definitions

Definition 1.1 (Positive definite (PD) kernel) *Let $\mathscr{X}$ be a non-empty set. A symmetric function $k_{\mathscr{X}} : \mathscr{X} \times \mathscr{X} \to \mathbb{R}$ is* positive-definite *if, for any $n \in \mathbb{N}$, any choice of points $x_1, \ldots, x_n \in \mathscr{X}$, and any coefficient vector $c = (c_1, \ldots, c_n)^\top \in \mathbb{R}^n$,*

$$c^\top K_X c \;=\; \sum_{i=1}^{n} \sum_{j=1}^{n} c_i c_j k_{\mathscr{X}}(x_i, x_j) \;\geq\; 0,$$

where $K_X = [k_{\mathscr{X}}(x_i, x_j)]_{i,j=1}^n$ is the Gram matrix of $k_{\mathscr{X}}$ over the sample $\{x_i\}_{i=1}^n$. Equivalently, the $n \times n$ Gram matrix $K_X = [k_{\mathscr{X}}(x_i, x_j)]_{i,j=1}^n$ is symmetric positive semidefinite for all finite samples.

Definition 1.2 (Reproducing kernel Hilbert space) *The Hilbert space $\mathscr{H}_{\mathscr{X}}$ associated with a PD kernel $k_{\mathscr{X}}$ is known as a reproducing kernel Hilbert space (RKHS). It is a space of functions on $\mathscr{X}$ satisfying:*

1. *$k_{\mathscr{X}}(\cdot, x) \in \mathscr{H}_{\mathscr{X}}$ for all $x \in \mathscr{X}$*
2. *Reproducing Property: $f(x) \;=\; \langle f,\, k_{\mathscr{X}}\big(x, \cdot\big) \rangle_{\mathscr{H}_{\mathscr{X}}}$, for all $f \in \mathscr{H}_{\mathscr{X}}$ and $x \in \mathscr{X}$.*

Therefore, we can view $k_{\mathscr{X}}\big(x, \cdot\big)$ itself as a function in $\mathscr{H}_{\mathscr{X}}$, and the inner product in $\mathscr{H}_{\mathscr{X}}$ "reproduces" the function values $f(x)$. This property allows one to regard $k_{\mathscr{X}}(\cdot, \cdot)$ as the canonical feature map from $\mathscr{X}$ into $\mathscr{H}_{\mathscr{X}}$:

$$\psi(x) \;=\; k_{\mathscr{X}}\big(x, \cdot\big) \;\in\; \mathscr{H}_{\mathscr{X}}.$$

1.2 Classical Theorems

Mercer's theorem (Mercer, 1909; Steinwart & Scovel, 2012). Suppose $\mathscr{X} \subset \mathbb{R}^d$ is compact, $k_{\mathscr{X}}$ is continuous, and define the positive definite integral operator $\mathscr{T} : L_2(\mathscr{X}, \mu) \to L_2(\mathscr{X}, \mu)$ by $(\mathscr{T} f)(x) = \int_{\mathscr{X}} k_{\mathscr{X}}(x, x') f(x') \, \mathrm{d}\mu(x')$. Then there exist orthonormal eigenfunctions $\{\psi_\ell\}_{\ell=1}^\infty$ of operator $\mathscr{T}$ and non-negative eigenvalues $\{\lambda_\ell\}_{\ell=1}^\infty$ such that:

$$k_{\mathscr{X}}(x, x') \;=\; \sum_{\ell=1}^{\infty} \lambda_\ell \, \psi_\ell(x) \, \psi_\ell(x'), \qquad \forall x, x' \in \mathscr{X}.$$

. where the convergence is absolute and uniform over $x, x' \in \mathscr{X}$.

Note in practice, one often does not compute this expansion explicitly. Instead, one exploits the reproducing property, performing computations through $k_{\mathscr{X}}(x,x')$ directly rather than through the explicit mapping $\psi(x)$.

Bochner's theorem. (Bochner, 1933) A continuous function $k_{\mathscr{X}} : \mathbb{R}^d \times \mathbb{R}^d \to \mathbb{C}$ of the form $k_{\mathscr{X}}(x,x') = \psi(x-x')$ is called translation-invariant kernel. This is PD iff ψ is the inverse Fourier transform of a finite non-negative Borel measure Λ on $\mathbb{R}^d$:

$$k_{\mathscr{X}}(x,x') \;=\; \psi(x-x') \;=\; \int_{\mathbb{R}^d} e^{\mathrm{i}\,\omega^\top (x-x')}\, \mathrm{d}\Lambda(\omega).$$

Bochner's theorem thus characterizes translation–invariant kernel $\psi(x-x')$ by its *spectral density / measure* Λ. Two important consequences are worth highlighting.

- Random Fourier Features (Rahimi & Recht, 2007)

  Sampling $\omega_1, \ldots, \omega_D \overset{\text{iid}}{\sim} \Lambda$ and defining the real feature map

  $$z_D(x) \;=\; \frac{1}{\sqrt{D}} \Big[\cos(\omega_1^\top x), \sin(\omega_1^\top x), \;\ldots,\; \cos(\omega_D^\top x), \sin(\omega_D^\top x) \Big]^\top$$

  yields the unbiased Monte-Carlo estimate

  $$k(x,x') \;\approx\; z_D(x)^\top z_D(x'),$$

  This converts kernel algorithms into scalable linear models whose time and memory costs depend on the chosen feature dimension D rather than on the number of data points n.

- Characteristic kernels for kernel mean embeddings (Fukumizu et al., 2008; Sriperumbudur et al., 2010)

  As we will see later, kernel mean embedding is defined as $\mu_{P(X)} = \mathbb{E}_X[\psi(X)] \in \mathscr{H}_{\mathscr{X}}$. The kernel $k_{\mathscr{X}}$ is said to be *characteristic* if the map $P \mapsto \mu_P$ is injective, implying $\mu_P = \mu_Q \Rightarrow P = Q$. Bochner's theorem implies that a translation–invariant kernel is characteristic iff its spectral measure has full support: $\operatorname{supp}\Lambda = \mathbb{R}^d$.

1.3 Popular Kernels

Many kernel-based algorithms, such as Support Vector Machines (Schölkopf & Smola, 2001) and Gaussian Processes (Rasmussen & Williams, 2006), use PD kernels as a way to measure similarity among inputs in $\mathscr{X}$. We list some of the frequently used PD kernels below (For more examples see Rasmussen and Williams (2006) and Schölkopf and Smola (2001)):

- **Polynomial kernel of degree p:**

$$k_{\mathscr{X}}(x,x') \;=\; \big(1+x^{\top}x'\big)^{p}.$$

- **Gaussian / RBF kernel:**

$$k_{\mathscr{X}}(x,x') \;=\; \exp\Big(-\tfrac{\|x-x'\|_2^2}{2\sigma^2}\Big).$$

- **Laplace kernel:**

$$k_{\mathscr{X}}(x,x') \;=\; \exp\Big(-\tfrac{\|x-x'\|_1}{\sigma}\Big).$$

- **Matérn kernel:**

$$k_{\mathscr{X}}(x,x') \;=\; \frac{2^{1-\nu}}{\Gamma(\nu)}\Big(\sqrt{2\nu}\,\tfrac{\|x-x'\|_2}{\sigma}\Big)^{\nu} K_{\nu}\Big(\sqrt{2\nu}\,\tfrac{\|x-x'\|_2}{\sigma}\Big).$$

  K_{ν} is the modified Bessel function of the second kind. Special cases: $\nu=\frac{1}{2}$ gives the Laplace kernel; $\nu\to\infty$ yields the Gaussian kernel.

These kernels differ in the *spectral decay* of their associated power spectra, so choosing a kernel amounts to encoding prior assumptions about function smoothness and regularity.

- **Gaussian / RBF kernel.**

  The spectral density is itself Gaussian, $\Lambda_{\text{Gaussian}}(\omega)\propto\exp\big(-\frac{\sigma^2}{2}\|\omega\|^2\big)$, which suppresses high–frequency components exponentially as ω increases and yields functions that are infinitely differentiable.

- **Laplace kernel.**

  The spectral density, $\Lambda_{\text{Lapace}}(\omega)=\frac{1}{2\pi(\omega^2+\beta^2)}$ with $\beta=1/\sigma$, is heavier tailed compared to the Gaussian kernel case, producing rougher functions.

- **Matérn kernel.**

  The spectral density $\Lambda_{\text{Matérn}}(\omega) \propto \left(\frac{2\nu}{\sigma^2} + 4\pi^2\|\omega\|^2\right)^{-\nu-d/2}$ decays polynomially as ω increases. This is also norm-equivalent to the Soblev space of order $\nu + d/2$.

The flexibility of PD kernels, combined with the ability to conduct all computations via $k_{\mathscr{X}}(x,x')$ rather than explicit feature vectors, has historically made them a standard tool in machine learning. However, when data size n is large, naive kernel approaches can become computationally expensive. This is because the resulting Gram matrices are of size $n \times n$, and operations such as matrix inversion become prohibitively costly. Throughout this thesis, we explore how such kernel methods can be integrated with *deep learning* to enhance scalability and expressive power, while still retaining the theoretical grounding provided by RKHS-based methods.

2 Kernel Mean Embeddings

Definitions For a given marginal distribution $P(X)$, the kernel mean embedding (KME) $\mu_{P(X)}$ is defined as the expectation of the feature $\psi(X)$:

$$\mu_{P(X)} = \mathbb{E}_X\left[\psi(X)\right] \in \mathscr{H}_{\mathscr{X}},$$

and always exists for bounded kernels. The reproducing property of RKHSs equips KMEs with the useful ability to estimate function expectations:

$$\left\langle f, \mu_{P(X)} \right\rangle_{\mathscr{H}_{\mathscr{X}}} = \mathbb{E}_X\left[f(X)\right],$$

for any $f \in \mathscr{H}_{\mathscr{X}}$. Furthermore, for *characteristic kernels*, these embeddings are injective, uniquely defining the probability distribution (Fukumizu et al., 2008; Sriperumbudur et al., 2010). For instance, popular kernels such as Gaussian and Laplace kernels possess this property.

The second order mean embeddings, also known as covariance operators (Fukumizu et al., 2004), are defined as the expectation of tensor products between features, for instance:

$$C_{XY} = \mathbb{E}_{XY}\left[\psi(X) \otimes \phi(Y)\right],$$

where $\otimes$ is the tensor product, and always exist for bounded kernels. Covariance operators generalize the familiar notion covariance matrices to accommodate infinite-dimensional feature spaces.

The covariance operator can also be regarded as a Hilbert–Schmidt operator $C_{XY} : \mathscr{H}_{\mathscr{Y}} \longrightarrow \mathscr{H}_{\mathscr{X}}$. For any $g \in \mathscr{H}_{\mathscr{Y}}$ the action of C_{XY} on g is

$$\big(C_{XY}\, g\big) \;=\; \mathbb{E}_{XY}\big[\psi(X)\,\langle \phi(Y), g\rangle_{\mathscr{H}_{\mathscr{Y}}}\big].$$

Empirical Estimates of KMEs Given an *i.i.d.* sample $\{x_i\}_{i=1}^n \sim P(X)$ the KME can be approximated by the sample average in feature space

$$\widehat{\mu}_{P(X)} \;=\; \frac{1}{n}\sum_{i=1}^{n} \psi(x_i) \;\in\; \mathscr{H}_{\mathscr{X}}.$$

For bounded kernels this estimator is unbiased and converges in $\mathscr{H}_{\mathscr{X}}$-norm at the rate $\mathscr{O}_P(n^{-1/2})$.

For an i.i.d. sample of pairs $\{(x_i, y_i)\}_{i=1}^n \sim P(X,Y)$ and feature maps $\psi : \mathscr{X} \to \mathscr{H}_{\mathscr{X}}$ and $\phi : \mathscr{Y} \to \mathscr{H}_{\mathscr{Y}}$, the (cross-)covariance operator $C_{XY} = \mathbb{E}[\psi(X) \otimes \phi(Y)]$ is estimated by

$$\widehat{C}_{XY} \;=\; \frac{1}{n}\sum_{i=1}^{n} \psi(x_i) \otimes \phi(y_i) \;\in\; \mathscr{H}_{\mathscr{X}} \otimes \mathscr{H}_{\mathscr{Y}}.$$

With the Hilbert–Schmidt operator formulation, we can also write:

$$\widehat{C}_{XY} = \frac{1}{n}\sum_{i=1}^{n} \psi(x_i)\,\langle \phi(y_i), \cdot\rangle_{\mathscr{H}_{\mathscr{Y}}}.$$

When $k_{\mathscr{X}}$ and $k_{\mathscr{Y}}$ are bounded, $\widehat{C}_{XY}$ is an unbiased estimator that converges in operator norm at the $\mathscr{O}_P(n^{-1/2})$ rate. Auto-covariance operators $\widehat{C}_{XX}$ and $\widehat{C}_{YY}$ are obtained by setting $Y = X$ or $X = Y$ respectively, and a centered version is formed by replacing each feature $\psi(x_i)$ with $\psi(x_i) - \widehat{\mu}_{P(X)}$.

Maximum Mean Discrepancy (MMD). For two probability densities P and Q on $\mathscr{X}$, Maximum Mean Discrepancy (MMD) (Gretton et al., 2012) is defined as:

$$\mathrm{MMD}(P, Q; k_{\mathscr{X}}) \;:=\; \sup_{\substack{f \in \mathscr{H}_{\mathscr{X}} \\ \|f\|_{\mathscr{H}_{\mathscr{X}}} \le 1}} \Big\{ \mathbb{E}_{X\sim P}\big[f(X)\big] \;-\; \mathbb{E}_{Y\sim Q}\big[f(Y)\big] \Big\},$$

which measures the largest mean discrepancy attainable by the function f of RKHS norm at most one. This function can be written $f = (\mu_P - \mu_Q)/\|\mu_P - \mu_Q\|_{\mathscr{H}_\mathscr{X}}$, and evaluating the supremum, we obtain:

$$\mathrm{MMD}^2(P, Q; k_\mathscr{X}) = \|\mu_P - \mu_Q\|^2_{\mathscr{H}_\mathscr{X}}.$$

Hence $\mathrm{MMD}(P, Q; k_\mathscr{X}) = 0$ if and only if $P = Q$ whenever $k_\mathscr{X}$ is characteristic.

With samples $\{x_i\}_{i=1}^m \sim P$ and $\{y_j\}_{j=1}^n \sim Q$, an unbiased U-statistic estimator:

$$\widehat{\mathrm{MMD}}^2_u = \frac{1}{m(m-1)} \sum_{i \neq i'} k_\mathscr{X}(x_i, x_{i'}) + \frac{1}{n(n-1)} \sum_{j \neq j'} k_\mathscr{X}(y_j, y_{j'}) - \frac{2}{mn} \sum_{i=1}^m \sum_{j=1}^n k_\mathscr{X}(x_i, y_j),$$

while the biased V-statistic uses all pairs in the first two sums.

Kernel herding. Kernel herding (Y. Chen et al., 2010) is a *supersampling* procedure that iteratively constructs a deterministic point set $\{x_t\}_{t=1}^T$ whose empirical KME matches μ_P. Starting from $x_1 = \arg\max_{x \in \mathscr{X}} \langle \psi(x), \mu_P \rangle$ and given the current average $\bar{\mu}_{t-1} = \frac{1}{t-1} \sum_{s=1}^{t-1} \psi(x_s)$, choose

$$x_t = \arg\max_{x \in \mathscr{X}} \left\langle \psi(x),\ \mu_P - \bar{\mu}_{t-1} \right\rangle_{\mathscr{H}_\mathscr{X}}.$$

The resulting empirical embedding $\bar{\mu}_T = \frac{1}{T} \sum_{t=1}^T \psi(x_t)$ converges to μ_P at rate $\mathscr{O}(T^{-1})$. The algorithm is shown in Algorithm 1.

Algorithm 1 Kernel Herding for Supersampling a Distribution P

Input: kernel $k_{\mathscr{X}}$ with feature map ψ, target distribution P (accessible via samples), number of iterations T

Output: Deterministic point set $\{x_t\}_{t=1}^{T}$ whose empirical KME approximates μ_P

(1) Compute/estimate the population mean embedding

$$\mu_P \;=\; \mathbb{E}_{X\sim P}\big[\psi(X)\big] \quad \text{or} \quad \widehat{\mu}_P = \tfrac{1}{n}\sum_{i=1}^{n}\psi(x_i) \text{ if only samples are available.}$$

(2) Initialize

$$x_1 = \arg\max_{x\in\mathscr{X}} \big\langle \psi(x), \mu_P \big\rangle_{\mathscr{H}_{\mathscr{X}}}$$

(3) Iterative herding step (for $t = 2,\ldots,T$)

1. Update the empirical mean: $\bar{\mu}_{t-1} = \dfrac{1}{t-1}\displaystyle\sum_{s=1}^{t-1}\psi(x_s)$.
2. Select the next point: $x_t = \arg\max_{x\in\mathscr{X}} \big\langle \psi(x), \mu_P - \bar{\mu}_{t-1} \big\rangle_{\mathscr{H}_{\mathscr{X}}}$.
3. Update: $\bar{\mu}_t = \dfrac{t-1}{t}\,\bar{\mu}_{t-1} + \dfrac{1}{t}\,\psi(x_t)$.

3 Kernel Conditional Mean Embeddings

Definition and the Empirical Estimate. With the aforementioned building blocks in place, we now extend KMEs to the case of conditional distributions, defining kernel conditional mean embeddings (CMEs) (Muandet et al., 2017; Song et al., 2009) as follows:

$$\mu_{P(Y|X)}(x) = \mathbb{E}_{Y|x}[\phi(Y)|X = x] \in \mathscr{H}_{\mathscr{Y}},$$

requiring an operator $C_{Y|X} : \mathscr{H}_{\mathscr{X}} \to \mathscr{H}_{\mathscr{Y}}$ that satisfies: (a) $\mu_{P(Y|X)}(x) = C_{Y|X}\,\psi(x)$ and (b) $\big\langle g, \mu_{P(Y|X)}(x)\big\rangle_{\mathscr{H}_{\mathscr{Y}}} = \mathbb{E}_{Y|x}[g(Y)|X = x]$ for $g \in \mathscr{H}_{\mathscr{Y}}$. Assuming

$$\mathbb{E}_{Y|x}[g(Y)|X = \cdot] \in \mathscr{H}_{\mathscr{X}},$$

the following operator satisfies the requirements:

$$C_{Y|X} = (C_{XX})^{-1}C_{XY}.$$

Given i.i.d samples $\{(x_i, y_i)\}_{i=1}^n \sim P$, an empirical estimate of the operator can be obtained as follows:

$$\begin{aligned}\widehat{C}_{Y|X} &= (\hat{C}_{XX} + \lambda I)^{-1} \hat{C}_{XY} \\ &= \Phi(K_X + \lambda I)^{-1} \Psi^\top,\end{aligned}$$

where $\lambda > 0$ is a regularization parameter, K_X is the Gram matrix $(K_X)_{ij} = k_{\mathscr{X}}(x_i, x_j)$, and Ψ and Φ are the feature matrices stacked by columns: $\Psi = [\psi(x_1), \ldots, \psi(x_n)]$ and $\Phi = [\phi(y_1), \ldots, \phi(y_n)]$. Alternatively, this empirical estimate can be obtained by solving following function-valued regression problem (Grünewälder et al., 2012):

$$\underset{C:\mathscr{H}_{\mathscr{X}} \to \mathscr{H}_{\mathscr{Y}}}{\arg\min} \frac{1}{n} \sum_{i=1}^{n} \|\phi(y_i) - C\psi(x_i)\|^2_{\mathscr{H}_{\mathscr{Y}}} + \lambda \|C\|^2_{HS}, \tag{2.1}$$

where $\|C\|_{HS}$ is the Hilbert-Schmidt norm. Putting together these elements, we arrive at the empirical estimator:

$$\hat{\mu}_{P(Y|X)}(x) = \widehat{C}\psi(x) = \sum_{i=1}^{n} \beta_i(x)\phi(y_i) = \Phi\boldsymbol{\beta}(x), \tag{2.2}$$

where:

$$\boldsymbol{\beta}(x) = (K_X + \lambda I)^{-1} \boldsymbol{k}_X,$$

with $\boldsymbol{k}_X = [k_{\mathscr{X}}(x_1, x), \ldots, k_{\mathscr{X}}(x_n, x)]^\top$. Intuitively, $\beta_i(x)$ corresponds to "weights" on the particles represented by $\phi(y_i)$. In contrast to KMEs for marginal distributions, CMEs employ non-uniform weights $\beta_i(x)$, which are not constrained to be positive or sum up to one.

Derivation. Here, we derive the empirical estimate of CME shown previously.

Step 1: We first show the following lemma.:

Lemma 3.1 (Right-multiplied Woodbury Identity) *For any invertible square matrix M and compatible matrices U and V,*

$$(M + UV)^{-1} U \;=\; M^{-1} U \big(I + VM^{-1}U\big)^{-1}.$$

To prove this, we start from the standard Woodbury identity

$$(M + UV)^{-1} = M^{-1} - M^{-1} U \big(I + VM^{-1}U\big)^{-1} VM^{-1}.$$

Right-multiply both sides by U:

$$(M+UV)^{-1}U = M^{-1}U - M^{-1}U\big(I+VM^{-1}U\big)^{-1}VM^{-1}U.$$

Set $X := VM^{-1}U$ and factor out $M^{-1}U$:

$$(M+UV)^{-1}U = M^{-1}U\big[I-(I+X)^{-1}X\big].$$

Note that $I-(I+X)^{-1}X = (I+X)^{-1}(I+X)-(I+X)^{-1}X = (I+X)^{-1}$. Substituting this identity yields

$$(M+UV)^{-1}U = M^{-1}U\big(I+VM^{-1}U\big)^{-1}.$$

Step 2: In the previous lemma, by choosing

$$M = \lambda I, \qquad U = \tfrac{1}{\sqrt{n}}\Psi, \qquad V = \tfrac{1}{\sqrt{n}}\Psi^\top,$$

we have:

$$(\tfrac{1}{n}\Psi\Psi^\top + \lambda I)^{-1}\tfrac{1}{\sqrt{n}}\Psi \;=\; \tfrac{1}{\sqrt{n}}\Psi\big(\lambda I + \tfrac{1}{n}\Psi^\top\Psi\big)^{-1}.$$

Step 3:

$$\begin{aligned}(\widehat{C}_{XX}+\lambda I)^{-1}\widehat{C}_{XY} &= \big(\lambda I + \tfrac{1}{n}\Psi\Psi^\top\big)^{-1}\big(\tfrac{1}{n}\Psi\Phi^\top\big)\\ &\overset{\text{Step 2}}{=} \Psi\big(n\lambda I + K_X\big)^{-1}\Phi^\top.\end{aligned}$$

Re-ordering the factors and resetting $n\lambda \to \lambda$ yields representation

$$\widehat{C}_{Y|X} = \Phi\big(K_X + \lambda I_n\big)^{-1}\Psi^\top.$$

Measure Theoretic Formulation. In the above, we formulated CMEs based on the operator $C_{Y|X} : \mathscr{H}_\mathscr{X} \to \mathscr{H}_\mathscr{Y}$. This formulation requires the assumption $\mathbb{E}_{Y|x}[g(Y)|X=\cdot] \in \mathscr{H}_\mathscr{X}$, which encodes a smoothness assumption on the operator. It intuitively means that for any function $g(Y)$, its expected value $\mathbb{E}[g(Y)]$ remains a smooth function of X within the $\mathscr{H}_x$. Since this condition may not always hold true in practice, a regularization parameter λ is introduced in the empirical estimate to circumvent this issue.

This assumption can be removed by the recently established measure theoretic approach to CMEs (Park & Muandet, 2020). The function-valued regression problem (2.1) can be reformulated in the following way:

$$\operatorname*{arg\,min}_{F\in\mathscr{G}_{\mathscr{X}\mathscr{Y}}} \frac{1}{n}\sum_{i=1}^{n} \|\phi(y_i) - F(x_i)\|^2_{\mathscr{H}_\mathscr{Y}} + \lambda\,\|F\|^2_{\mathscr{G}_{\mathscr{X}\mathscr{Y}}},$$

where $\mathscr{G}_{\mathscr{X}\mathscr{Y}}$ is a vector-valued RKHS of functions $\mathscr{X} \to \mathscr{H}_{\mathscr{Y}}$. More formally, CME is defined based on the expression of Bochner conditional expectation. This formulation is also used to analyze the statistical learning rate of CMEs (Li et al., 2022). In the case of our NK-CMEs, we will be using NNs for $F \in \mathscr{G}_{\mathscr{X}\mathscr{Y}}$, and with this formulation it may be possible to prove the rate of convergence and the theoretical advantage of using NNs over kernels similarly to works such as Imaizumi and Fukumizu (2019) and Suzuki (2019).

4 Deep Learning: Structure and Training

Deep learning, or Deep Neural Networks (DNNs), (Goodfellow et al., 2016) have become a cornerstone of modern machine learning due to their ability to learn complex, hierarchical representations from data. Formally, a DNN is a parametric function:

$$f_\theta : \mathscr{X} \to \mathscr{Z},$$

where θ denotes all learnable parameters (weights and biases). For simplicity, here we consider a feed-forward network (sometimes called a multilayer perceptron) with L layers. We index the layers by $\ell = 1, \ldots, L$. Let $h^{(\ell)}(x)$ be the output of layer ℓ, with $h^{(0)}(x) = x$ denoting the input.

Then, the forward pass can be written as:

$$h^{(\ell)}(x) \;=\; \sigma\Big(W^{(\ell)}\, h^{(\ell-1)}(x) \;+\; b^{(\ell)}\Big), \quad \ell = 1, \ldots, L,$$

where $W^{(\ell)}$ is the weight matrix and $b^{(\ell)}$ is the bias vector for layer ℓ. The function $\sigma(z)$ is an elementwise nonlinear activation, such as the RELU: $\max\{0, z\}$, the logistic sigmoid: $\frac{1}{1+e^{-z}}$, or the hyperbolic tangent: $\tanh(z)$. The *depth* L is the number of nonlinear transformations, while the *width* at layer ℓ is the dimension of $h^{(\ell)}$. The network's final output is $h^{(L)}(x) \in \mathscr{Z}$. For example, if used for scalar regression, then $\dim \mathscr{Z} = 1$; if used for classification into C categories, we might set $\dim \mathscr{Z} = C$ and often employ a softmax to obtain probabilities.

Training of a DNN entails finding the parameter vector θ that minimizes a chosen loss function $\mathscr{L}(\theta)$. Typically, the loss is defined as an average of $\mathscr{L}\big(f_\theta(x_i), y_i\big)$ over training pairs $\{(x_i, y_i)\}_{i=1}^n$. The standard algorithmic approach is stochastic gradient descent (SGD) or one of its many variants. Rather than computing the gradient on all n samples at each iteration (full-batch gradient descent), *mini-batch*

SGD samples a subset $\mathscr{B} \subset \{1,\ldots,n\}$ of size B. Parameters are then updated via:

$$\theta \;\leftarrow\; \theta \;-\; \eta\,\nabla_\theta \left[\frac{1}{B}\sum_{i\in\mathscr{B}} \mathscr{L}\Big(f_\theta(x_i),\, y_i\Big)\right], \tag{2.3}$$

where $\eta > 0$ is the learning rate. This procedure iterates across many mini-batches, gradually refining θ to (locally) minimize the loss. Variants like Adam optimizer (Kingma & Ba, 2015) introduce adaptive learning rates and momentum terms, often improving practical performance.

5 Deep Feature Approach for CMEs

Empirical Estimate. Instead of relying on pre-specified RKHS feature maps, the Deep Feature approach (DF) (Xu et al., 2021) harnesses the adaptive capabilities of deep learning to learn tailored feature representations. This approach has demonstrated its efficacy in settings such as causal inference (Xu et al., 2021) and kernelized Bayes' rule (Xu et al., 2022), often outperforming classical CME methods.

By replacing ψ with a d-dimensional NN-parameterized feature map $\psi_\theta : \mathscr{X} \to \mathbb{R}^d$ (where θ denotes the NN's parameters) in (2.1), we can jointly optimize both the feature representation and the conditional operator C. This is achieved by optimizing θ with the following loss function:

$$\hat{\mathscr{L}}(\theta) = \operatorname{tr}\Big(K_Y(I - \Psi_\theta^\top(\Psi_\theta\Psi_\theta^\top + \lambda I)^{-1}\Psi_\theta)\Big),$$

where K_Y is the Gram matrix with elements $(K_Y)_{ij} = k_{\mathscr{Y}}(y_i, y_j)$, and gradient-based optimization methods can be applied. Given the learned parameter $\hat{\theta} = \arg\min_\theta \hat{\mathscr{L}}(\theta)$, we can express $\boldsymbol{\beta}(x)$ in (2.2) as follows:

$$\boldsymbol{\beta}(x) = \Psi_{\hat{\theta}}^\top \left(\Psi_{\hat{\theta}}\Psi_{\hat{\theta}}^\top + \lambda I\right)^{-1} \psi_{\hat{\theta}}(x).$$

Derivation of the Loss Function. The loss function can be derived by inserting $\widehat{C}_{Y|X} \;=\; \Phi\Psi_\theta^\top\big(\Psi_\theta\Psi_\theta^\top + n\lambda I_d\big)^{-1}$ into the following loss function:

$$n\,\mathscr{L}(\theta) = \underbrace{\big\|\Phi - \widehat{C}_{Y|X}\Psi_\theta\big\|^2_{\mathscr{H}_{\mathscr{Y}}}}_{\text{data-fit}} \;+\; \underbrace{n\lambda\,\|\widehat{C}_{Y|X}\|^2_{HS}}_{\text{regularization term}}.$$

Let $A := (\Psi_\theta \Psi_\theta^\top + n\lambda I_d)^{-1}$. Then,

$$\begin{aligned}\left\|\Phi - \widehat{C}_{Y|X}\Psi_\theta\right\|_{\mathscr{H}_\mathscr{X}}^2 &= \operatorname{tr}\Big[\left(\Phi - \Phi\Psi_\theta^\top A\Psi_\theta\right)\left(\Phi - \Phi\Psi_\theta^\top A\Psi_\theta\right)^\top\Big] \\ &= \operatorname{tr}\Big[\Phi\Phi^\top - 2\Phi\Psi_\theta^\top A\Psi_\theta\Phi^\top + \Phi\Psi_\theta^\top A\Psi_\theta\Psi_\theta^\top A\Psi_\theta\Phi^\top\Big].\end{aligned}$$

Note the identity $\Psi_\theta\Psi_\theta^\top A = I_d - n\lambda A$, so the last term becomes

$$\Phi\Psi_\theta^\top A(I_d - n\lambda A)\Psi_\theta\Phi^\top = \Phi\Psi_\theta^\top A\Psi_\theta\Phi^\top - n\lambda\,\Phi\Psi_\theta^\top A^2\Psi_\theta\Phi^\top.$$

Hence

$$\left\|\Phi - \widehat{C}_{Y|X}\Psi_\theta\right\|_{\mathscr{H}_\mathscr{X}}^2 = \operatorname{tr}\Big[\Phi\Phi^\top - \Phi\Psi_\theta^\top A\Psi_\theta\Phi^\top - n\lambda\,\Phi\Psi_\theta^\top A^2\Psi_\theta\Phi^\top\Big].$$

For the regularization term, we have

$$n\lambda\,\|\widehat{C}_{Y|X}\|_{HS}^2 = n\lambda\,\operatorname{tr}\big[\Phi\Psi_\theta^\top A^2\Psi_\theta\Phi^\top\big].$$

The $-n\lambda\,\operatorname{tr}(\cdot)$ term from the data-fit expansion is exactly cancelled by the $+n\lambda\,\operatorname{tr}(\cdot)$ in the regularization term. What remains is

$$n\mathscr{L}(\theta) = \operatorname{tr}\Big[\Phi\Phi^\top - \Phi\Psi_\theta^\top A\Psi_\theta\Phi^\top\Big] = \operatorname{tr}\Big[K_Y\left(I_n - \Psi_\theta^\top(\Psi_\theta\Psi_\theta^\top + n\lambda I_d)^{-1}\Psi_\theta\right)\Big],$$

by setting $K_Y = \Phi^\top\Phi$ and replacing $n\lambda \to \lambda$, we have the loss function introduced previously.

Discussions. This DF approach offers a partial solution to scalability challenges as well. Its computational complexity of $O(nd^2 + d^3)$ is typically smaller than the $O(n^3)$ complexity of classical approaches. During the training, mini-batch optimization can be be applied for further acceleration. However, the regularization parameter λ plays a vital role in both performance and numerical stability, and its selection often necessitates computationally expensive cross-validation procedures involving multiple NN optimizations. Additionally, this approach on its own does not address the remaining hyperparameter selection issue for $k_\mathscr{Y}$.

Chapter 3

Limitations of Conditional Kernel Mean Embeddings

This chapter examines in detail the limitations of current CMEs. First, we contrast their scalability and expressiveness with deep neural networks, highlighting where kernel-based methods fall short. Then, we explain why standard hyperparameter selection methods, such as cross-validation, are inapplicable to the selection of output kernel hyperparameters. Together, these insights motivate and sets stage for the proposal in Chapter 4.

1 Scalability

For kernel methods like CMEs, the computational cost of inverting the Gram matrix is $O(N^3)$, where N is the number of training points. It is well-established that this calculation becomes expensive (in terms of computational time and memory) and unstable as N grows beyond a few hundred thousand. This is the main reason why many research efforts have been conducted on cost-reduction techniques, based on methods such as Random Fourier Features (Rahimi & Recht, 2007) and Nyström approximations (Drineas & Mahoney, 2005) for kernel methods, and similarly inducing point approaches (Hensman et al., 2013) in the Gaussian Process community.

We empirically verify the $\mathcal{O}(n^3)$ computational cost using NumPy's `linalg.solve` wraps, which solves $AX = B$. This is typically used for solving regressions,

since it is more stable than naively computing `X = np.linalg.inv(A) @ B`. `linalg.solve` implements LAPACK's `GESV` routine to solve $AX = B$ by computing an LU factorization with partial pivoting and row interchanges, then performing forward and back substitution, at a cost of approximately $\frac{2}{3}n^3 + O(n^2)$ floating-point operations.

Table 3.1: Benchmark of `numpy.linalg.solve` compute times for varying matrix sizes.

matrix size n	computation time (ms)
2^{10}	15.3
2^{11}	91.5
2^{12}	4.26×10^2
2^{13}	2.40×10^3
2^{14}	1.49×10^4
2^{15}	1.55×10^5

The results are shown in Table 3.1, where we report the mean compute time over 5 runs which conducted on MacBook Pro with M2 system, only using CPU.. As n increases, the runtime scales close to, and for the largest n even exceeds, the expected $\mathcal{O}(n^3)$.

We also note that kernel-based approaches typically require hyperparameter search to achieve good performance. Each hyperparameter configuration entails one or more matrix inversions, compounding the scalability issues on large datasets.

In contrast for DNNs, with the use of mini-batch optimization explained above, the computational cost for each update scales with $\mathcal{O}(B)$ rather than $\mathcal{O}(n)$, enabling efficient updates especially when $B \ll n$. Moreover, modern deep learning frameworks exploit *GPU acceleration*, which efficiently handles the matrix-vector operations required for forward/backward passes on each mini-batch. As a result, training on large-scale datasets with n in the millions becomes feasible.

In Chapter 5, we compare the computational time of classic kernel-based CMEs and our NK-CMEs. On a synthetic task with n=5000, we verify that NK-CMEs run faster both during training and at evaluation.

2 Expressiveness

From theoretical perspective, it is well-studied that NNs can outperform kernel methods in terms of convergence rates, particularly when target functions exhibit properties like piece-wise smoothness (Imaizumi & Fukumizu, 2019), non-uniform smoothness (Suzuki, 2019), etc. In these settings, kernel methods are limited to sub-optimal rates, while NNs can achieve the minimax optimal rate. Intuitively, this advantage stems from the ability of NNs to learn adaptive and expressive features directly from the data, unlike kernel methods which rely on pre-defined RKHS features.

This limitation that kernel methods rely on pre-defined RKHS features, is also related to another line of work that shows infinitely-wide neural networks trained with small learning rates behave like kernel methods. At initialization, the network output is distributed as a Gaussian Process with covariance given by the Neural-Network Gaussian-Process (NNGP) kernel (Lee et al., 2018). When the same network is optimized by sufficiently small-step gradient descent, its evolution in function space follows the Neural Tangent Kernel (NTK) (Jacot et al., 2018). The Tensor-Programs framework generalizes this correspondence to modern architectures such as residual networks and Transformers, proving that their infinite-width limits are still Gaussian Processes or NTK (Yang, 2019). In all of these cases the feature map is effectively frozen: under NNGP it never changes, and under NTK it moves only by $\mathcal{O}(1/\text{width})$, so the network performs linear prediction in a fixed feature space. Consequently, the network performs linear prediction in a fixed feature space—the so-called lazy-learning regime rigorously analyzed by Chizat et al. (2019) and many subsequent studies.

In summary, for complex target functions, deep neural networks are expected to learn more efficiently than kernel methods. Although the NNGP and NTK viewpoints make kernels formally more expressive, they still amount to a linearization around random initialization; as a result, they remain less powerful than neural networks that learn representations directly from data.

In Chapter 5, we compare classic kernel based CMEs against deep feature based CMEs on several UCI datasets, where we find that the kernel-based CMEs consistently underperform.

3 Hyperparameter Tuning

CME mainly have three types of hyperparameters to tune.: parameters on $k_{\mathscr{X}}$, parameters on $k_{\mathscr{Y}}$ and regularization parameter λ.

Fortunately, for $k_{\mathscr{X}}$ and λ, it is possible to calculate the analytical leave-one-out (LOO) error and use it for tuning. The formula generalizes the well-known analytical LOO for kernel ridge regression. Remind that for CME we have $\widehat{C}_{Y|X} = \Phi(K_X + \lambda I)^{-1}\Psi^\top$, and define the *hat-matrix* $H := K_X(K_X + \lambda I)^{-1} \in \mathbb{R}^{n\times n}$ and the residual matrix $R := \Phi - \widehat{C}_{Y|X}\Psi = \Phi - H\Phi$.

Then, let $\widehat{C}^{(-i)}_{Y|X}$ be the estimator trained on all points except (x_i, y_i). The leave-one-out prediction at x_i is

$$\widehat{\phi}^{(-i)} = \widehat{C}^{(-i)}_{Y|X}\psi(x_i) = \phi(y_i) - \frac{R_{\bullet i}}{1 - H_{ii}},$$

where $R_{\bullet i}$ is the i-th column of R. Hence the squared LOO error is

$$\mathrm{LOO}(k_{\mathscr{X}}, \lambda) := \frac{1}{n}\sum_{i=1}^{n} \big\| \phi(y_i) - \widehat{\phi}^{(-i)} \big\|^2_{\mathscr{H}_{\mathscr{Y}}} = \frac{1}{n}\mathrm{tr}\Big[R\,\mathrm{diag}\big((1-H_{11})^{-2}, \ldots, (1-H_{nn})^{-2}\big)R^\top\Big].$$

Equivalently,

$$\mathrm{LOO}(k_{\mathscr{X}}, \lambda) = \frac{1}{n}\mathrm{tr}\left[\frac{R^\top R}{(I-H)^2}\right]$$

However, for $k_{\mathscr{Y}}$, there exist no "correct" way of tuning parameters. This is because the CME objective itself depends on the RKHS norm induced by $k_{\mathscr{Y}}$. Changing the kernel parameters thus alters the objective function fundamentally.

To illustrate this, we give a concrete example. Suppose we use a Gaussian RBF kernel for $k_{\mathscr{Y}}$ on the bimodal synthetic data from Chapter 5. As shown in Figure 3.1, when the bandwidth σ grows large, the RKHS loss (or average residual R) approaches zero. Consequently, if one performs cross-validation using the RKHS loss to select the kernel bandwidth, it will always pick the largest σ in the candidate set—regardless of whether that choice yields meaningful conditional embeddings.

Therefore, one must rely on heuristics. When using Gaussian kernel, median heuristics is widely used for selecting σ parameter:

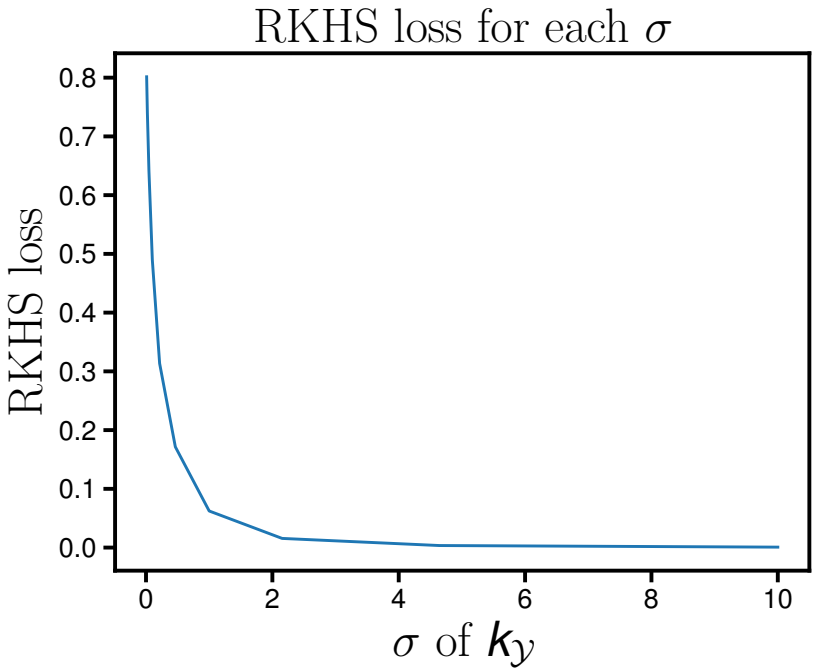

Figure 3.1: Illustrative example showing why the hyperparameter selection of the output kernel is nontrivial

1. Collect all (or a random subset of) pairwise squared distances $\{\|x_i - x_j\|^2 : 1 \leq i < j \leq N\}$.
2. Let $\hat{d}_{\text{med}}$ be the median of this set.
3. Set the bandwidth to $\sigma = \sqrt{\hat{d}_{\text{med}}}$.

In Chapter 5, we demonstrate on both synthetic and UCI datasets that the deep-feature approach with median heuristics consistently underperforms our NK-CME, whose bandwidth parameters are optimized.

One exception arises when a downstream task can guide hyperparameter tuning. For example, in instrumental variable regression (Xu et al., 2021), two regressions are performed: the first uses a CME and the second is a standard regression on the CME's outputs. In this setting, the performance of the downstream (second) regression provides a natural criterion for selecting the output-kernel parameters.

Chapter 4

Neural-Kernel Conditional Mean Embeddings

In this chapter, we introduce our Neural-Kernel Conditional Mean Embeddings (NK-CMEs) approach, which aims to address the scalability and expressiveness limitations of classical CMEs discussed in Chapter 2. We then present our hyperparameter optimization strategy for the kernel on output variables. Finally, we describe related methods for conditional distribution modeling, which we compare experimentally in the next chapter.

1 NK-CME: Model and Objective Function

While Deep Feature (DF) approach of Xu et al. (2021) partially addresses computational challenges of CME, it still restricts $\boldsymbol{\beta}(x)$ to a specific functional form involving matrix inversion. The core idea of our approach is to replace $\boldsymbol{\beta}(x)$ in (2.2) with $f(x;\theta) : \mathscr{X} \to \mathbb{R}^M$, a DNN parameterized by θ. We propose the CME estimator of the following form:

$$\hat{\mu}_{P(Y|X)}(x) = \sum_{a=1}^{M} \phi(\eta_a) f_a(x;\theta),$$

where $\eta_a \in \mathscr{Y}$ are M location parameters. These parameters can be optimized together with the NN parameter θ, or can be fixed to reduce the number of parameters. In this paper, we opt for fixing them throughout for easiness of optimization.

Analogous to (2.1) but without $\|C\|_{HS}^2$, θ is optimized through:

$$\min_{\theta} \frac{1}{n} \sum_{i=1}^{n} \left\| \phi(y_i) - \sum_{a=1}^{M} \phi(\eta_a) f_a(x_i;\theta) \right\|_{\mathscr{H}_{\mathscr{Y}}}^2 .$$

We denote this as $\min_\theta \frac{1}{n}\sum_{i=1}^n \hat{\ell}(\theta)$, where $\hat{\ell}(\theta)$ can be further expressed as:

$$\hat{\ell}(\theta) \propto -2\sum_{a} k_{\mathscr{Y}}(y_i,\eta_a) w_a + \sum_{a,b} k_{\mathscr{Y}}(\eta_a,\eta_b) w_a w_b, \tag{4.1}$$

with $w_a = f_a(x_i;\theta)$ and $w_b = f_b(x_i;\theta)$. The term $\sum_{i=1}^n k_{\mathscr{Y}}(y_i,y_i)$ is omitted, assuming fixed hyperparameters for $k_{\mathscr{Y}}$, for now. In contrast to DF, the whole process of calculating $\beta(x)$ with explicit features is effectively amortized with $f(x;\theta)$. This eliminates the need for both Gram matrix inversion and regularization parameter selection. We call this method Neural-Kernel Conditional Mean embeddings (NK-CMEs). We note that in the NK-CME objective, the Hibert-Schmidt regularization term in the original RKHS objective is omitted, and we instead regularize DNNs by standard techniques such as weight decays (Loshchilov & Hutter, 2019).

2 Towards Efficient Hyperparameter Optimization

NK-CME framework in theory, should address the scalability and expressiveness limitations of classical CMEs. However, selecting hyperparameters for $k_{\mathscr{Y}}$ remains a challenge due to the dependence of the objective values on the RKHS norm $\|\cdot\|_{\mathscr{H}}$; they are not comparable for different hyperparameters. Thus we emphasize again that $k_{\mathscr{Y}}$ hyperparameters cannot be selected by standard procedures such as cross-validation using RKHS-based loss function. While downstream tasks can sometimes guide tuning (e.g., in IV regression: (Xu et al., 2021)), one typically resorts to heuristic methods like the median heuristic (Gretton et al., 2005). Our aim in the following sections is to construct a criterion that can serve as a supplementary objective function for optimizing the $k_{\mathscr{Y}}$ parameter, offering a more effective approach to its selection.

2.1 Choice of Kernel $k_{\mathscr{Y}}$

Our initial step is to employ a positive definite kernel k that satisfies: $k(\cdot,\cdot) \geq 0$ and $\int k(y,\cdot)dy = 1$. Thus, the kernels we consider are both reproducing kernels

and smoothing kernels for densities, two often confused, but distinct notions of kernel functions. In the following, we adopt what we call the *Gaussian density kernel* for $k_{\mathscr{Y}}$:

$$k_\sigma(y,y') = \left(\frac{1}{\sqrt{2\pi\sigma^2}}\right)^{d_y} \exp\left(-\frac{\|y-y'\|^2}{2\sigma^2}\right).$$

We will denote the associated RKHS by $\mathscr{H}_\sigma$. This type of kernel has seen previous use in (Hsu & Ramos, 2019; Kim & Scott, 2012), and variants corresponding to other kernels, such as Laplace and Student kernels, are also available. Because Gaussian density kernel is also a smoothing kernel, CME estimator also gives an estimator $\hat{p}(y|x)$ of the conditional probability density, obtained through the inner product $\left\langle k_\sigma(y,\cdot), \hat{\mu}_{P(Y|X)}(x)\right\rangle_{\mathscr{H}_\sigma}$ as

$$\hat{\mathbb{E}}_{Y|x}[k_\sigma(y,Y)|X=x] = \sum_{a=1}^{M} k_\sigma(y,\eta_a) f_a(x;\theta). \tag{4.2}$$

This form of density estimate has been theoretically discussed in Kanagawa and Fukumizu (2014). It is also worth noting that with similar probability estimate, Hsu and Ramos (2019) constructed a marginal likelihood-like objective for hyperparameter selection of CMEs, within the context of likelihood-free inference. While they prove that this objective provides an asymptotically correct likelihood surrogate, it is not guaranteed to be positive or normalized for finite data. Consequently, we treat it solely as a proxy for conditional probability, limiting its use to hyperparameter selection criteria.

We emphasize that our focus is on CMEs, i.e. on representing conditional distributions as mean elements within the RKHS. This enables two key advantages: first, we can represent distributions even without constraining the NN output, and second, it opens up applications beyond standard density estimation tasks, as showcased by the unique properties of KMEs explored in chapter 6.

2.2 Approach 1: Iterative Optimization

Based on the conditional probability estimate (4.2), we propose optimizing the bandwidth parameter σ to minimize an objective function based on the L^2 norm. Accordingly, we adopt the following squared (SQ) error loss:

$$\mathscr{L}_{SQ} = \frac{1}{2}\iint \left(\hat{p}(y|x) - p(y|x)\right)^2 p(x)dxdy.$$

This objective function has been used in the density-ratio-based conditional density estimation method (Sugiyama et al., 2010). In our case, by plugging in $\hat{p}(y|x) = \sum_{a=1}^{M} k_\sigma(y, \eta_a) f_a(x; \theta)$, we obtain the following empirical loss $\frac{1}{n}\sum_{i=1}^{n} \hat{\ell}_{SQ}(\sigma)$, where:

$$\hat{\ell}_{SQ}(\sigma) = -2\sum_{a} k_\sigma(y_i, \eta_a) w_a + \sum_{a,b} k_{\sqrt{2}\sigma}(\eta_a, \eta_b) w_a w_b.$$

The derivation is given in section 3.1. In practice, we iteratively optimize θ and σ, through $\min_\theta \frac{1}{n}\sum_{i=1}^{n} \hat{\ell}(\theta)$ and $\min_\sigma \frac{1}{n}\sum_{i=1}^{n} \hat{\ell}_{SQ}(\sigma)$, every step. Since η_a within $k_\sigma(\cdot, \eta_a)$ are model parameters, and not randomly sampled during optimization, σ can be optimized stably using minibatch optimization, with minimal gradient variance. Complemented with regularization techniques such as weight decay (Loshchilov & Hutter, 2019), we observe that the optimization can generally be carried out without encountering severe overfitting.

2.3 Approach 2: Joint Optimization

Interestingly, the only distinction between (4.1) and $\hat{\ell}_{SQ}(\sigma)$ lies in their second terms, representing the squared functional norms on $\mathscr{H}_\sigma$ and $\mathscr{H}_{\sqrt{2}\sigma}$, respectively. This close resemblance hints at the potential for joint optimization of θ and σ using the objective function given in (4.1), expressed as $\min_{\theta,\sigma} \frac{1}{n}\sum_{i=1}^{n} \hat{\ell}(\theta, \sigma)$. The following theorem provides a key justification for this approach by establishing an inequality that connects these two norms:

Theorem 2.1 *Let* $f = \sum_{a=1}^{M} k_{\sqrt{2}\sigma}(\cdot, \eta_a) w_a \in \mathscr{H}_{\sqrt{2}\sigma}$ *and* $g = \sum_{a=1}^{M} k_\sigma(\cdot, \eta_a) w_a \in \mathscr{H}_\sigma$. *Then the following inequality holds:*

$$\|f\|_{\mathscr{H}_{\sqrt{2}\sigma}} \leq \|g\|_{\mathscr{H}_\sigma}.$$

The proof, provided in section 3.2, leverages the Fourier transform expression of RKHS for translation invariant kernels. By replacing the second term of $\hat{\ell}_{SQ}(\sigma)$ with this upper bound, we observe a coincidence with the original objective function (4.1). This result validates our proposal for *joint* optimization of θ and σ using (4.1), since it simultaneously optimizes an upper bound of $\hat{\ell}_{SQ}(\sigma)$ for σ, while optimizing the original loss function for θ. Joint optimization not only simplifies the procedure but also, as a valuable byproduct, mitigates overfitting in hyperparameter optimization due to the utilization of an upper bound. Indeed, we observe that joint optimization, with its combined advantages, yields stable performance across various datasets.

2.4 Discussions

We note that when using Gaussian density kernel and also optimizing σ, the omitted term in (4.1), $\sum_{i=1}^{n} k_{\mathscr{Y}}(y_i, y_i)$, is not constant anymore. Therefore, our joint optimization approach is not about just replacing Gaussian kernel with Gaussian density kernel and then optimizing σ with the original RKHS loss objective. The supplementary loss function based on squared error $\hat{\ell}_{SQ}(\sigma)$ is crucial for the hyperparamter optimization. An interesting aspect of the joint optimization approach is that the upper bound of this $\hat{\ell}_{SQ}(\sigma)$ (for optimizing σ) and the original RKHS objective (4.1) (for optimizing θ and thus $\sum_{i=1}^{n} k_{\mathscr{Y}}(y_i, y_i)$ omitted) coincides.

We also note that our supplementary squared error loss $\hat{\ell}_{SQ}(\sigma)$ can be calculated in closed-form due to a convenient property of the Gaussian density kernel. This property relates to the kernel's close connection to the Gaussian distribution (See Section 3.1 for derivation). This closed-form expression directly contributes to Theorem 2.1, justifying the efficient joint optimization strategy. With other kernel choices, while it is still possible to approximate $\hat{\ell}_{SQ}$, we may lose the computational efficiency that this Gaussian density kernel brings. On the other hand, it is also true that the best choice of kernel can depend on the task and desired properties. For instance, the Laplace-type kernel could be useful when capturing high-frequency information in the data is crucial.

3 Proofs

3.1 Derivation of the Empirical SQ Loss

The SQ loss can be further expanded as follows:

$$\begin{aligned}\mathscr{L}_{SQ} &= \frac{1}{2}\iint \left(\hat{p}(y|x) - p(y|x)\right)^2 p(x)dxdy. \\ &= \frac{1}{2}\iint \left(\hat{p}(y|x)\right)^2 p(x)dxdy - \iint \hat{p}(y|x)p(x,y)dxdy + C,\end{aligned}$$

where C is the constant term. By plugging in $\hat{p}(y|x) = \sum_{a=1}^{M} k_\sigma(y, \eta_a) f_a(x;\theta)$, the empirical estimate can be written as:

$$\hat{\mathscr{L}}_{SQ} = \frac{1}{2}\sum_i \sum_{a,b} \left(\int k_\sigma(\eta_a, y) k_\sigma(y, \eta_b) dy\right) f_a(x_i;\theta) f_b(x_i;\theta) - \sum_i \sum_a k_\sigma(y_i, \eta_a) f_a(x_i;\theta).$$

The second term corresponds to the first term in $\hat{\ell}_{SQ}(\sigma)$. For the first term, we have an integral $\int k_\sigma(\eta_a, y)k_\sigma(y, \eta_b)dy$. With k_σ having the form of Gaussian density, this can be calculated analytically:

$$\begin{aligned}
\int k(\eta_a, y)k(y, \eta_b)dy &= \left(\frac{1}{2\pi\sigma^2}\right)^{d_y} \int \exp\left(-\frac{1}{2\sigma^2}(|\eta_a - y|^2 + |y - \eta_b|^2)\right) dy \\
&= \left(\frac{1}{2\pi\sigma^2}\right)^{d_y} \int \exp\left(-\frac{1}{4\sigma^2}\left(4\left|y - \frac{\eta_a + \eta_b}{2}\right|^2 + |\eta_a - \eta_b|^2\right)\right) dy \\
&= \left(\frac{1}{2\pi\sigma^2}\right)^{d_y} \int \exp\left(-\frac{1}{\sigma^2}\left|y - \frac{\eta_a + \eta_b}{2}\right|^2\right) dy \exp\left(-\frac{1}{4\sigma^2}|\eta_a - \eta_b|^2\right) \\
&= \left(\frac{1}{2\pi(\sqrt{2}\sigma)^2}\right)^{d_y} \exp\left(-\frac{|\eta_a - \eta_b|^2}{2(\sqrt{2}\sigma)^2}\right) \\
&= k_{\sqrt{2}\sigma}(\eta_a, \eta_b).
\end{aligned}$$

combining these elements, we obtain:

$$\hat{\mathscr{L}}_{SQ} = \sum_{i=1}^{n} \left(-2\sum_a k_\sigma(y_i, \eta_a)f_a(x_i;\theta) + \sum_{a,b} k_{\sqrt{2}\sigma}(\eta_a, \eta_b)f_a(x_i;\theta)f_b(x_i;\theta)\right).$$

3.2 Proof of Theorem 2.1

We first introduce the Fourier expression of RKHS given by a shift invariant integrable kernel $k(x-y)$ on $\mathbb{R}^d$. Let $\hat{k}$ denote the Fourier transform of $k(z)$:

$$\hat{k}(\omega) = \frac{1}{(2\pi)^{d/2}} \int k(z)e^{-\sqrt{-1}\omega^T z}dz.$$

It is known (e.g. Girosi et al., 1995; Saitoh, 1997) that a function g on $\mathbb{R}^d$ is in the corresponding RKHS $\mathscr{H}_k$ if and only if

$$\int \frac{|\hat{g}(\omega)|^2}{\hat{k}(\omega)} d\omega < \infty$$

and its squared RKHS norm is given by

$$\|g\|^2_{\mathscr{H}_k} = \int \frac{|\hat{g}(\omega)|^2}{\hat{k}(\omega)} d\omega.$$

Note that by Bochner's theorem, the Fourier transform $\hat{k}(\omega)$ takes non-negative values.

It is well known that, the Fourier transform of Gaussian density kernel $k_\sigma(z) = \frac{1}{(2\pi\sigma^2)^{d/2}}\exp(-\frac{\|z\|^2}{2\sigma^2})$ is given by

$$\hat{k}_\sigma(\omega) = \frac{1}{(2\pi)^{d/2}} e^{-\frac{\sigma^2\|\omega\|^2}{2}}. \tag{4.3}$$

Let

$$g = \sum_{a=1}^{M} k_\sigma(\cdot, \eta_a) w_a \in \mathscr{H}_\sigma \quad \text{and} \quad f = \sum_{a=1}^{M} k_{\sqrt{2}\sigma}(\cdot, \eta_a) w_a \in \mathscr{H}_{\sqrt{2}\sigma}$$

be the two functions in Theorem 2.1. It is easy to see from the general Fourier formulas that

$$\hat{g}(\omega) = \sum_a w_a e^{-\sqrt{-1}\eta_a^T\omega}\hat{k}_\sigma(\omega)$$

and thus we obtain from (4.3)

$$\|g\|^2_{\mathscr{H}_\sigma} = \frac{1}{(2\pi)^{d/2}} \int \Big|\sum_a w_a e^{-\sqrt{-1}\eta_a^T\omega}\Big|^2 \exp\Big(-\frac{\sigma^2\|\omega\|^2}{2}\Big) d\omega.$$

Similarly, by replacing σ with $\sqrt{2}\sigma$, we have

$$\|f\|^2_{\mathscr{H}_{\sqrt{2}\sigma}} = \frac{1}{(2\pi)^{d/2}} \int \Big|\sum_a w_a e^{-\sqrt{-1}\eta_a^T\omega}\Big|^2 \exp\big(-\sigma^2\|\omega\|^2\big) d\omega.$$

It follows from $\exp\big(-\sigma^2\|\omega\|^2\big) \le \exp\Big(-\frac{\sigma^2\|\omega\|^2}{2}\Big)$ for any ω that

$$\|f\|^2_{\mathscr{H}_{\sqrt{2}\sigma}} \le \|g\|^2_{\mathscr{H}_\sigma},$$

which completes the proof.

4 Related Approaches for Conditional Distribution Modeling

Mixture Density Networks (MDNs). Bishop (1994) introduced Mixture Density Networks (MDNs) as a way to model conditional probability distributions.

Let $x \in \mathscr{X}$ be the input and $y \in \mathscr{Y} \subset \mathbb{R}^{d_y}$ the output. MDNs assume that $p(y \mid x)$ can be represented by a mixture of M Gaussian components:

$$p(y \mid x) \;=\; \sum_{m=1}^{M} \alpha_m(x)\, \mathscr{N}\Big(y;\, \mu_m(x),\, \Sigma_m(x)\Big),$$

where $\alpha_m(x)$ are the mixing coefficients (nonnegative and summing to 1), and each $\mathscr{N}(\,\cdot\,;\mu,\Sigma)$ is a Gaussian with mean $\mu_m(x)$ and covariance $\Sigma_m(x)$. A neural network predicts all these parameters:

$$\{\alpha_m(x)\}_{m=1}^{M}, \quad \{\mu_m(x)\}_{m=1}^{M}, \quad \{\Sigma_m(x)\}_{m=1}^{M}.$$

These outputs are learned via maximum likelihood, typically by minimizing the negative log-likelihood of the mixture model. Due to their simplicity, MDNs have found broad usage in tasks such as likelihood-free inference and probabilistic forecasting (Makansi et al., 2019; Papamakarios & Murray, 2016).

Normalizing Flows (NFs). Normalizing Flows (Papamakarios et al., 2021; Rezende & Mohamed, 2015) provide a framework to build flexible distributions by transforming a simple base distribution (e.g. isotropic Gaussian) via a sequence of *invertible* mappings. Concretely, if $Z \sim p_0(z)$ is the base distribution in $\mathbb{R}^d$, and $f_\theta(\cdot)$ is a differentiable bijection with inverse $f_\theta^{-1}(\cdot)$, then setting

$$Y \;=\; f_\theta(Z)$$

induces a distribution over Y, whose density can be computed via the change of variables formula:

$$p_\theta(y) \;=\; p_0\big(f_\theta^{-1}(y)\big)\, \Big|\det \nabla f_\theta^{-1}(y)\Big|.$$

Conditional normalizing flows extend this idea by allowing f_θ to depend on x, so that $p(y \mid x)$ is modeled via an invertible transform of a base distribution conditioned on x. A common example is the rational-quadratic spline flow (Durkan et al., 2019), which builds monotonic piecewise-rational-quadratic transformations for each dimension. These conditional flows have become widely used for simulation based inference in scientific domains (Lueckmann et al., 2021).

Conditional Denoising Diffusion (CARD). Recently, Han et al. (2022) proposed a method named *CARD* (Classification and Regression Diffusion) to perform conditional density estimation using diffusion-based generative modeling. The approach comprises two main parts:

1. *Conditional Mean Estimator:* A neural network first predicts the conditional mean $\mu(x) \approx \mathbb{E}[Y \mid x]$.

2. *Denoising Diffusion Model:* A second neural network then learns to generate samples around that mean by successively reversing a Gaussian diffusion process. In other words, starting from noise, the model learns to iteratively refine samples using a denoising procedure conditioned on x.

Because diffusion models can capture highly flexible, multi-modal distributions, CARD demonstrates strong performance on diverse practical tasks (Han et al., 2022).

We will compare these methods with our proposed approaches in the following chapter.

Chapter 5

Applications to Conditional Density Estimation

This chapter presents experimental results on conditional density estimation tasks. To investigate the effectiveness of our approach, we conduct experiments on both toy and real-world datasets. We focus on conditional density estimation tasks where the dimension of the output variable is one ($d_y = 1$). Note that the input variable can be multi-dimensional, and we conduct experiments on such settings using UCI datasets (Dua & Graff, 2017), where d_x can be up to 90.

We denote our approach proposed in Chapter 4, Section 2.2 and Section 2.3 as Proposal-Iterative and Proposal-Joint, respectively. We compare with Deep feature approach (DF), Mixture Density Networks (MDN), Normalizing Flow (NF), and Diffusion model (CARD). For DF, we tested two models: DF used with the median heuristic for bandwidth selection (DF-med) and DF with bandwidth fixed to 0.1 (DF-0.1). For NF, we use the conditional version of autoregressive Neural Spline Flow (Durkan et al., 2019).

Evaluation metrics employed in both toy and UCI settings require samples from the learned model. To sample points from CME-based approaches, we employ kernel herding (Y. Chen et al., 2010), a deterministic sampling method that yields super-samples.

1 Experiments on Toy Data

1.1 Toy Data Set Up

We conduct experiments on three distinct toy datasets: Bimodal, Skewed, and Ring. Each dataset features a one-dimensional input variable and comprises 5000 data points for training.

These datasets are designed to challenge models with diverse distributional characteristics: Bimodal dataset comprised of two Gaussian distributions, but the degree of bimodality and noise levels vary depending on the value of x. Skewed dataset are sampled from a skew-normal distribution, but distribution parameters, such as skewness, change as a function of x. Ring dataset features a ring-shaped distribution with an embedded box-shaped distribution.

The visualization of the toy datasets used in the experiment is shown in Figure 5.1. The generating process is provided in the following:

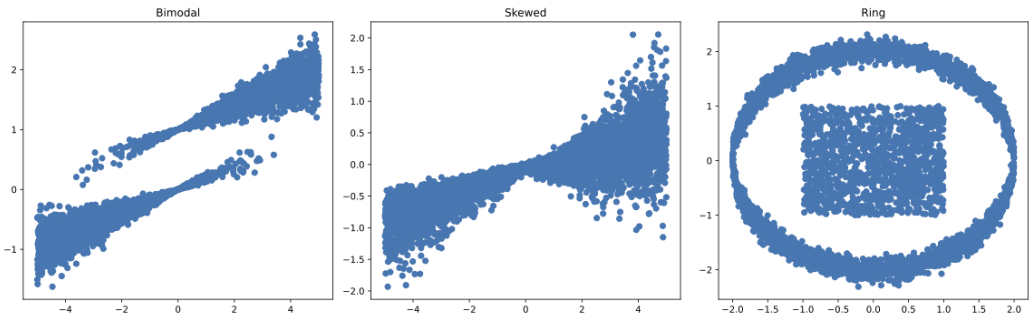

Figure 5.1: Visualization of three different toy datasets used for evaluating conditional density estimations.

Bimodal:

$y = 0.2x + P + \varepsilon$, where $x \sim U(-5,5)$, $P \sim Binomial\left(\frac{1}{1+\exp(-1.5x)}\right)$, and $\varepsilon \sim N(0,(0.05x)^2)$.

Skewed:

$y \sim SkewNormal(\xi(x),\omega(x),\alpha(x))$, where $x \sim U(-5,5)$, $\xi(x) = 0.1x$, $\omega(x) =$

$0.1|x| + 0.05$, and $\alpha(x) = -8 + 8 \cdot \left(\frac{1}{1+\exp(-x)}\right)$. Here, $\xi(x)$ corresponds to location, $\omega(x)$ to scale, and $\alpha(x)$ to skewness.

Ring:

$$y = \begin{cases} \begin{cases} U(-1,1) & P_1 = 1/2 \\ Ring(x) & P_1 = 1/2 \end{cases} & |x| \leq 1 \\ Ring(x) & |x| > 1 \end{cases},$$

and

$$Ring(x) = \begin{cases} 2 \cdot \sin(\arccos(x/2)) + \varepsilon & P_2 = 1/2 \\ 2 \cdot \sin(-\arccos(x/2)) + \varepsilon & P_2 = 1/2 \end{cases},$$

where $x \sim U(-2,2)$ and $\varepsilon \sim N(0, 0.1^2)$.

1.2 Evaluation Metric and Results

Evaluation Metric:

We evaluate models using a metric based on the Wasserstein-1 distance (WAS1), defined by

$$W_1(\mu, \nu) = \inf_{\pi \in \Gamma(\mu,\nu)} \int_{\mathbb{R}\times\mathbb{R}} |x - y| d\pi(x, y),$$

where $\Gamma(\mu, \nu)$ is the set of probability distributions whose marginals are μ and ν. In the one-dimensional case, this metric can be readily calculated using packages like SciPy (Virtanen et al., 2020).

The evaluation process involves the following steps: we first draw 50 samples from the learned models for each 200 equally spaced evaluation points x_{test}. We then, compute the WAS1s between these samples and 50 points sampled from the true generating process (for each x_{test}). We average the WAS1 values for all evaluation points and report the mean and standard deviation across 10 independent runs.

Results:

The overall results are presented in Table 5.1, demonstrating that our proposed approaches outperform competitors including NF and CARD. Crucially for DF,

Table 5.1: WAS1 for toy datasets. Values are multiplied by 100.

Dataset	WAS1 ↓ (lower the better)						
	Proposal-Iterative	Proposal-Joint	DF-med	DF-0.1	MDN	NF	CARD
Bimodal	5.88 ± 0.28	$\mathbf{5.67 \pm 0.28}$	10.87 ± 0.23	5.93 ± 0.23	6.23 ± 0.27	6.83 ± 0.50	6.59 ± 0.46
Skewed	4.86 ± 0.23	$\mathbf{4.68 \pm 0.22}$	6.07 ± 0.19	5.26 ± 0.19	5.76 ± 0.18	6.56 ± 0.29	5.93 ± 0.26
Ring	$\mathbf{21.13 \pm 0.99}$	$\mathbf{21.10 \pm 1.14}$	26.17 ± 1.16	22.16 ± 1.25	26.66 ± 0.80	27.99 ± 1.74	29.91 ± 0.97

the median heuristic fails to achieve optimal performance. It only becomes competitive with our approaches when the parameter is set to 0.1, which was identified through empirical trials. This highlights the general challenge of hyperparameter selection for the output variable kernel in CME-based methods, where both our iterative and joint approaches demonstrably lead to more effective parameter selection.

1.3 Experiments on Computational Costs

Evaluation time

To compare empirical computational times, we conducted additional experiments on our toy dataset (Bimodal). We begin by comparing the average time required to evaluate $f(x;\theta)$ for our proposal, and $\beta(x)$ for DF and classic kernel-based CME (Kernel). The experiment was conducted on MacBook Pro with M2 system, only using CPU.

Table 5.2: Computational time for evaluation

Proposal	DF	Kernel
$53.7\mu s \pm 201ns$	$82.6\mu s \pm 120ns$	$646ms \pm 43ms$

Even for a moderate data size of $N = 5000$, our proposal is about 10^4 faster than Kernel. This efficiency gap will become increasingly significant as N grows large, and particularly in scenarios requiring repeated evaluations, such as gradient-based optimization or RL tasks.

Training time

The comparison of training time is generally difficult due to its dependence on factors like the number of iterations, batch sizes, and number of hyperparameters to cross-validate. We report the training time in our toy data setting for our proposal (Proposal-Joint) and DF. For classic CME, we leverage the analytical solution of LOOCV to efficiently choose the best parameters among 10 bandwidths and 10 lambdas (and consider this as "training").

Table 5.3: Computational time for training

Proposal	DF	Kernel
$44.2s \pm 16.6ms$	$100.6s \pm 1.6s$	$305s \pm 2.44s$

It can be seen that our proposal is about 7 times faster than Kernel in this setting.

1.4 Ablations on learned σ

We first show that just applying the Gaussian density kernel and optimizing the original RKHS loss is not enough to optimize σ. Figure 5.2 shows the final (Full) RKHS loss when training NK-CME on the bimodal dataset with various fixed σ. As observed in the classical CME case, larger σ drives the loss close to zero, confirming that optimizing σ via the full RKHS loss remains infeasible.

By contrast, we show the case when we applied the objective used for the Proposal-Joint. By omitting the term $\sum_{i=1}^{n} k_{\mathscr{Y}}(y_i, y_i)$ and making it the upper bound of the square-error loss, it is now possible to learn σ using this modified loss. Figure 5.3 show that both objective and σ converges to sensible values regardless of the initialization of σ.

2 Experiments on UCI Datasets

2.1 Setup

To further investigate our approaches, we conduct experiments on 8 real-world regression benchmark datasets from the UCI repository, following the experimental setting of Han et al. (2022). We provide details of the datasets below:

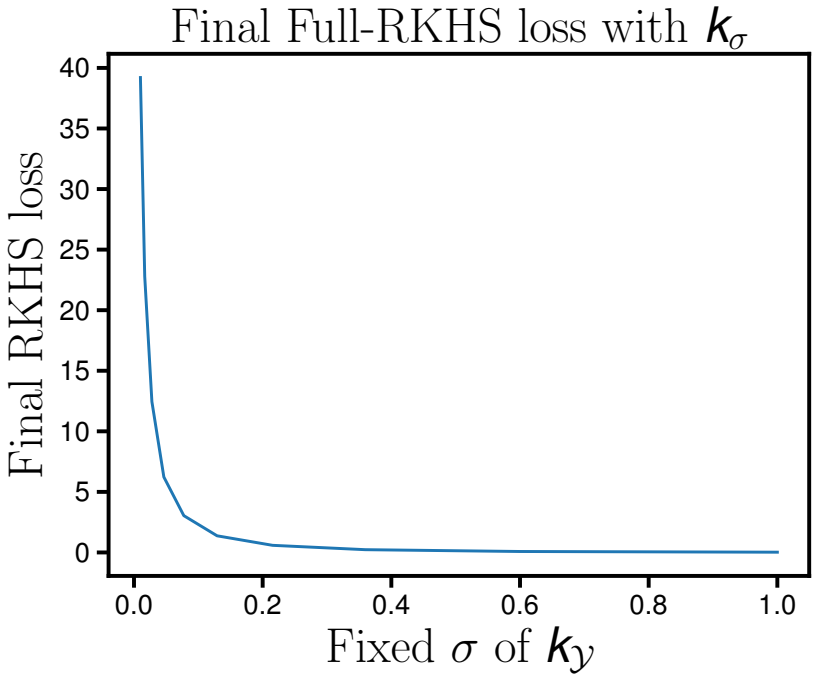

Figure 5.2: Illustration of why simply applying the gaussian density kernel is not enough.

We report the number of data points and input features for each dataset in Table 5.4. Compared to the UCI settings considered in Han et al. (2022), we excluded the Yacht dataset due to its limited size of only 308 data points, and the Wine dataset because its output variable consists of only 5 discrete values and exhibits near-linear relationships.

Table 5.4: (number of data points , number of input features) for each dataset

Dataset	Boston	Concrete	Energy	Kin8nm	Naval	Power	Protein	Year
	$(506, 13)$	$(1030, 8)$	$(768, 8)$	$(8192, 8)$	$(11934, 16)$	$(9568, 4)$	$(45730, 9)$	$(515345, 90)$

We follow experimental protocols of Han et al. (2022): (a) we employ the same train-test splits with a $90\%/10\%$ ratio, and use 20 folds for all datasets except Protein (5 folds) and Year (1 fold), (b) we standardize both input and output variables for training and remove standardization for evaluation, and (c) for each test data

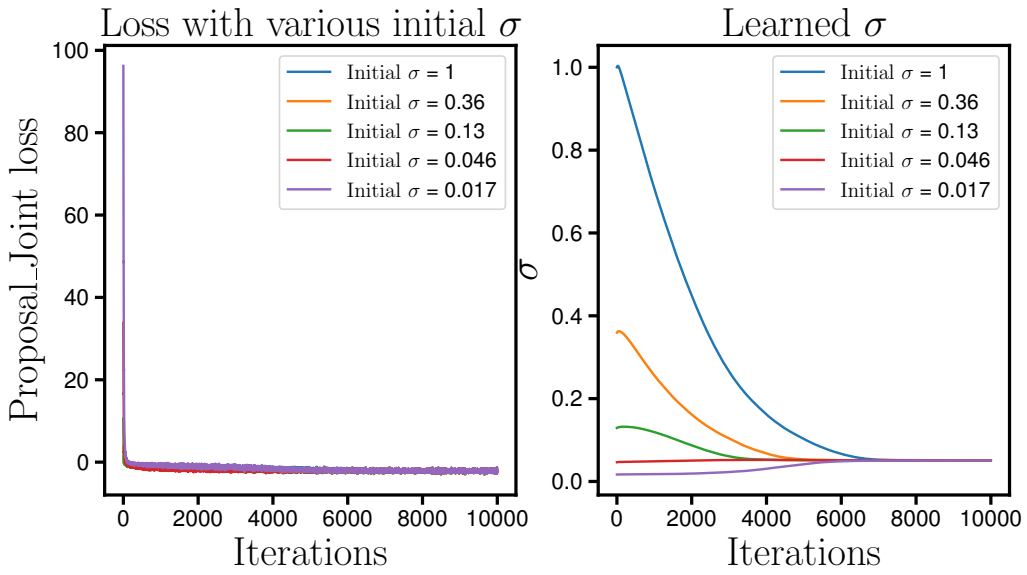

Figure 5.3: Objective and learned σ for the Proposal-Joint case. It shows that the method learns σ robustly regardless of the initialization.

point x_{test}, we sample 1000 points from learned models, conditioned on x_{test}, and calculate the metric described below.

2.2 Evaluation Metric and Results

Evaluation Metric: Since UCI datasets do not have the "true generating process", we adopt the Quantile Interval Coverage Error (QICE) metric proposed also by (Han et al., 2022). To compute QICE, we first generate sufficient number of samples for each conditional values x_i. We then divide the generated samples into L equally spaced bins, resulting in L quantile intervals with boundaries denoted as $\hat{y}_i^{\text{low}_j}$ and $\hat{y}_i^{\text{high}_j}$. Then compute the following quantity:

$$\text{QICE} = \frac{1}{L}\sum_{j=1}^{L}\left|r_j - \frac{1}{L}\right|, \text{ where}$$

$$r_j = \frac{1}{n}\sum_{i=1}^{n} 1_{y_i \geq \hat{y}_i^{\text{low}_j}} \cdot 1_{y_i \leq \hat{y}_i^{\text{high}_j}}.$$

Table 5.5: QICE (in %) for UCI datasets.

Dataset	QICE ↓ (lower the better)						
	Proposal-Iterative	Proposal-Joint	DF-med	DF-0.1	MDN	NF	CARD
Boston	$\mathbf{3.15 \pm 0.92}$	$\mathbf{3.09 \pm 0.53}$	5.74 ± 1.17	3.88 ± 0.82	4.85 ± 1.19	4.09 ± 0.95	3.45 ± 0.83
Concrete	3.06 ± 0.82	3.21 ± 0.72	4.05 ± 0.95	4.11 ± 0.66	3.21 ± 0.79	2.90 ± 0.79	$\mathbf{2.30 \pm 0.66}$
Energy	$\mathbf{2.89 \pm 0.69}$	3.29 ± 0.73	7.14 ± 0.88	3.23 ± 0.90	3.54 ± 0.85	3.62 ± 1.23	4.91 ± 0.94
Kin8nm	0.98 ± 0.29	$\mathbf{0.91 \pm 0.19}$	4.70 ± 0.32	2.60 ± 0.41	2.42 ± 0.32	1.75 ± 0.87	$\mathbf{0.92 \pm 0.25}$
Naval	10.88 ± 1.09	6.81 ± 1.07	7.09 ± 1.04	8.71 ± 0.37	8.34 ± 3.41	4.41 ± 1.54	$\mathbf{0.80 \pm 0.21}$
Power	0.88 ± 0.24	$\mathbf{0.84 \pm 0.18}$	4.81 ± 0.33	2.91 ± 0.18	1.34 ± 0.35	1.35 ± 0.59	0.92 ± 0.21
Protein	$\mathbf{0.48 \pm 0.05}$	0.55 ± 0.17	1.83 ± 0.19	0.80 ± 0.07	1.09 ± 0.35	0.80 ± 0.38	0.71 ± 0.11
Year	$0.71 \pm$ NA	$\mathbf{0.53} \pm$ NA	$1.80 \pm$ NA	$0.56 \pm$ NA	$0.74 \pm$ NA	$1.02 \pm$ NA	$\mathbf{0.53} \pm$ NA

When the learned conditional distribution perfectly matches the true distribution, we expect approximately $1/L$ of the true data to fall within each of the L quantile intervals, resulting in a QICE value of 0. In this experiment, we follow Han et al. (2022) and set $L = 10$. We report mean and standard deviation of the QICE metric across all splits.

Results:

The overall results are presented in Table 5.5. It can be seen that our proposed approaches frequently outperform existing methods such as DF, MDN, and NF, and often prove competitive with CARD in terms of the QICE metric. This is particularly impressive considering CARD requires two separate NN optimization procedures: 1. Conditional mean estimator optimization, and 2. Denoising diffusion model optimization, guided by the optimized conditional mean estimator. In contrast, our approaches achieve a comparable level of high-quality density estimation as diffusion-based CARD, while only using a single NN. The Proposal-Joint approach stands out with its exceptional efficiency, requiring only a standard NN training procedure.

It is also worth noting that in the UCI experiments, DF-0.1 clearly underperforms compared to our approaches, crucially demonstrating the importance of tailoring bandwidth σ to each dataset for optimal performance. This suggests that a fixed σ value, as used in DF-0.1, is typically insufficient for achieving good results across diverse datasets.

2.3 Performance of classical CMEs

To compare the empirical performance of kernel methods with NN-based approaches, we report the QICE values of classic kernel-based CME on UCI datasets. We used the analytical LOOCV solution for CME to select hyperparameters (the output-kernel bandwidth $k_{\mathscr{Y}}$ was fixed at 0.1).

Table 5.6: QICE values for classic kernel-based CME on UCI datasets.

Dataset	QICE value
Boston	3.97 ± 0.83
Concrete	4.51 ± 0.70
Energy	4.51 ± 0.95
Kin8nm	4.42 ± 0.25
Naval	10.77 ± 0.23
Power	1.25 ± 0.21
Protein	N/A (did not complete within 24 h)
Year	N/A (OOM during matrix inversion)

Classic CME generally performs worse than our NK-CME and deep-feature methods, confirming the advantage of NN-adaptive representations.

3 Additional Details on Experiments

3.1 Toy Data: Architecture and Hyperparameter Choices

We used NNs with two fully-connected hidden layers, each containing 50 ReLU activation units. For the optimizer, we used AdamW (Loshchilov & Hutter, 2019). Other architectural and hyperparameter choices for each model are provided below:

Proposals: We set the number of location points $M = 100$, and η_a were chosen as uniformly spaced grid points within the closed interval bounded by the minimum and maximum values observed in the training data. The learning rate was set to 1e-4, the batch size was set to 50, and the number of training epochs was set to 1000, and σ was initialize to 1.0. For Proposals, we have a final layer that outputs the weights on the location points.

DFs: The regularization parameter λ was set to 0.1, the learning rate was set to 1e-4, the batch size was set to 50, and the number of training epochs was set to 1000. For the kernel, we used Gaussian kernel. For DFs, the second hidden layer also corresponds to the final layer that outputs the features.

MDN: The number of mixing component was set to 10, the learning rate was set to 1e-4, the batch size was set to 50, and the number of training epochs was set to 1000. For MDN, we have three final layers that output means, variances and mixing weights.

NF: We used the conditional version of the autoregressive Neural Spline Flow (Durkan et al., 2019) implementation by Stimper et al. (2023). The number of flows were set to 5, the learning rate was set to 1e-3, the batch size was set to 256, and the number of training epochs was set to 100. For NF, the number of NNs mirrors the number of flows, resulting in 5 NNs in this case.

CARD: We used the implementation provided by Han et al. (2022), and a GitHub repository https://github.com/lightning-uq-box/lightning-uq-box. For the conditional mean estimator, we applied the same NN architecture as other methods, and the learning rate was set to 1e-3, the batch size was set to 256, and the number of training epochs was set to 100. For denoising diffusion model, we used the same architecture used in the toy data experiments in Han et al. (2022): NN based on three fully-connected hidden layers with 128 units. The learning rate was set to 1e-3, the batch size was set to 256, the number of time steps was set to 1000, and the number of training epochs was set to 5000. Note that for CARD, we first optimize the conditional mean estimator, and then use this to guide the next optimization procedure for the denoising diffusion model.

3.2 UCI: Architecture and Hyperparameter Choices

For Boston, Concrete, Energy and Kin8nm, we used NNs with three fully-connected hidden layers, each containing 50 ReLU activation units. For Naval, Power, Protein and Year, we used NNs with three fully-connected hidden layers, each containing 100 ReLU activation units. For the optimizer, we used AdamW (Loshchilov & Hutter, 2019). We report the learning rate, the batch size and the number of training epochs in Table 5.7, Table 5.8, and Table 5.9, respectively. Other architectural and hyperparameter choices for each model are provided below:

Proposals: We used Spectral Normalization (SN) (Miyato et al., 2018) in the second hidden layer and the final output layer. We set the number of location

points $M = 100$, and η_a were chosen as uniformly spaced grid points within the closed interval bounded by the minimum and maximum values observed in the training data. The bandwidth σ was initialized to 1.0. For Year dataset, when iteratively optimizing kernel parameter (Proposal-Iterative), we opted to update σ every 4 steps.

DFs: We used SN in the third hidden layer (which also corresponds to the final output layer). The regularization parameter λ was set to 0.1, and for the kernel we used Gaussian kernel.

MDN: We used SN in the third hidden layer. The number of mixing components was set to 10, except for Protein and Year where it was set to 5.

NF: We used the conditional version of the autoregressive Neural Spline Flow (Durkan et al., 2019) implementation by Stimper et al. (2023). For NF, we found that the model can easily overfit, and so we used slightly different NN architectures from the others: For Boston, Concrete, Energy and Kin8nm, we used NNs with **two** fully-connected hidden layers, each containing 50 ReLU activation units. For Naval, Power, Protein and Year, we used NNs with three fully-connected hidden layers, each containing **50** ReLU activation units. The number of flows were set to 5, and the number of training epoch was set as 10 to mitigate overfitting.

CARD: We directly adopt the results presented in Han et al. (2022).

Table 5.7: Learning rates for UCI experiments

	Proposals	DFs	MDN	NF
Boston	0.0005	0.0005	0.0005	0.001
Concrete	0.0005	0.0005	0.0005	0.001
Energy	0.0005	0.0005	0.0005	0.001
Kin8nm	0.0005	0.0005	0.0005	0.001
Naval	0.001	0.001	0.001	0.001
Power	0.001	0.001	0.001	0.001
Protein	0.001	0.001	0.001	0.001
Year	0.001	0.001	0.001	0.001

3.3 UCI: Additional Results: RMSE

Table 5.10 presents the RMSE values for each dataset. RMSE was computed using the same samples employed for QICE metric evaluation. Results show that CARD outperforms other methods in most cases. This aligns with expectations, as

Table 5.8: Batch sizes for UCI experiments

	Proposals	DFs	MDN	NF
Boston	32	32	32	50
Concrete	32	32	32	50
Energy	32	32	32	50
Kin8nm	100	100	100	100
Naval	256	100	100	256
Power	100	100	100	256
Protein	256	100	256	256
Year	512	256	512	256

Table 5.9: Training epochs for UCI experiments

	Proposals	DFs	MDN	NF
Boston	500	500	250	10
Concrete	500	500	250	10
Energy	500	500	250	10
Kin8nm	500	500	250	10
Naval	500	500	250	10
Power	500	500	250	10
Protein	500	500	250	10
Year	50	50	50	10

the initial step of CARD involves training a conditional mean estimator. Notably, other methods do not include explicit conditional mean estimation during training or within their objective functions. Consequently, we interpret comparable RMSE values to CARD as evidence that the generated samples avoids any pathological behaviors that may potentially exploit the QICE metric.

Table 5.10: RMSE for UCI datasets. For Kin8nm and Naval dataset, values are multiplied by 100.

Dataset	RMSE ↓ (lower the better)						
	Proposal-Iterative	Proposal-Joint	DF-med	DF-0.1	MDN	NF	CARD
Boston	3.56 ± 1.02	3.45 ± 1.08	3.40 ± 0.92	4.01 ± 1.08	3.36 ± 1.14	4.14 ± 1.18	$\mathbf{2.61 \pm 0.63}$
Concrete	6.40 ± 0.83	5.98 ± 0.63	5.84 ± 0.54	7.74 ± 0.76	5.62 ± 0.58	7.22 ± 0.62	$\mathbf{4.77 \pm 0.46}$
Energy	1.19 ± 0.21	0.99 ± 0.16	0.69 ± 0.13	2.67 ± 0.25	2.24 ± 0.43	2.88 ± 0.30	$\mathbf{0.52 \pm 0.07}$
Kin8nm	8.19 ± 0.23	8.11 ± 0.29	7.11 ± 0.21	9.52 ± 0.27	6.95 ± 0.22	7.98 ± 0.32	$\mathbf{6.32 \pm 0.18}$
Naval	0.09 ± 0.02	0.17 ± 0.03	$\mathbf{0.01 \pm 0.00}$	0.21 ± 0.02	0.08 ± 0.05	0.36 ± 0.08	0.02 ± 0.00
Power	3.96 ± 0.20	3.89 ± 0.19	4.08 ± 0.16	4.56 ± 0.17	$\mathbf{3.79 \pm 0.17}$	4.19 ± 0.18	3.93 ± 0.17
Protein	3.97 ± 0.05	3.93 ± 0.03	4.77 ± 0.05	4.83 ± 0.04	3.79 ± 0.04	4.38 ± 0.04	$\mathbf{3.73 \pm 0.01}$
Year	$8.82 \pm$ NA	$8.82 \pm$ NA	$8.88 \pm$ NA	$8.92 \pm$ NA	$8.79 \pm$ NA	$8.80 \pm$ NA	$\mathbf{8.70} \pm$ NA

Chapter 6

Applications to Distributional Reinforcement Learning

This chapter presents an application of NK-CME framework to distributional reinforcement learning (RL). We begin with a brief review of RL and distributional RL, then introduce our NK-CME–based method and highlight how it compares with existing approaches. Finally, we present experimental results on classic control environments.

1 Backgrounds on Reinforcement Learning

1.1 Setup and Q-Learning

Markov Decision Processes (MDPs). Reinforcement Learning (RL) commonly adopts the framework of a *Markov Decision Process (MDP)* (Puterman, 2014) to formalize agent-environment interactions. An MDP is defined as a tuple

$$\big(\mathscr{S},\, \mathscr{A},\, R,\, P,\, \gamma\big),$$

where:

- $\mathscr{S}$ is the *state space*, describing all possible configurations in which the agent might find itself.

- $\mathscr{A}$ is the *action space*, enumerating all moves or decisions the agent can make.
- $P(\cdot \mid s,a)$ is the *transition probability* distribution that governs how the environment evolves from state s upon taking action a.
- $R(s,a) \in \mathbb{R}$ is the *reward function*, specifying the immediate scalar reward the agent receives in state s when it takes action a.
- $\gamma \in (0,1]$ is the *discount factor*, regulating how future rewards are weighted relative to immediate ones.

At each discrete time step $t \in \{0,1,2,\ldots\}$, the agent observes a state $s_t \in \mathscr{S}$ and chooses an action $a_t \in \mathscr{A}$. The environment then (probabilistically) transitions to a new state $s_{t+1} \sim P(\cdot \mid s_t, a_t)$, and provides a reward $r_t = R(s_t, a_t)$. A *policy* π is a rule for selecting actions, often written $\pi(a_t \mid s_t)$. The objective of the agent is usually to choose a policy π that *maximizes* the long-term sum of discounted rewards.

Return and State-Action Values. We now define the random variable representing discounted cumulative return under policy π, starting from state-action pair (s,a), as

$$Z^{\pi}(s,a) \;=\; \sum_{t=0}^{\infty} \gamma^{t}\, R\big(s_t, a_t\big),$$

where $s_0 = s$, $a_0 = a$, and for $t \geq 0$,

$$s_{t+1} \;\sim\; P(\cdot \mid s_t, a_t), \quad a_t \;\sim\; \pi(\cdot \mid s_t).$$

The corresponding *Q-value function* is defined as the expectation of Z^{π}:

$$Q^{\pi}(s,a) \;=\; \mathbb{E}\big[Z^{\pi}(s,a)\big].$$

Intuitively, $Q^{\pi}(s,a)$ tells us how much total discounted reward we expect to accumulate if we act according to π, starting from (s,a).

Bellman Optimality and Q-Learning. A fundamental principle in RL is the *Bellman optimality operator*. Denoting by $\mathscr{T}$ the operator that maps any candidate Q-function $Q : \mathscr{S} \times \mathscr{A} \to \mathbb{R}$ to a new function $\mathscr{T}Q$, we have:

$$(\mathscr{T}Q)(s,a) \;\equiv\; \mathbb{E}\big[R(s,a)\big] \;+\; \gamma \mathbb{E}_{P}\Big[\max_{a'}\; Q(s',a')\Big],$$

where $s' \sim P(\cdot \mid s,a)$. The optimal Q-value function

$$Q^\star \;=\; \max_\pi Q^\pi$$

is the fixed point of $\mathscr{T}$, i.e. it satisfies $Q^\star = \mathscr{T}Q^\star$. A classical algorithm for finding $Q^\star$ is *Q-learning* (Watkins & Dayan, 1992), in which one iteratively updates an estimate $\hat{Q}$ toward the target given by the Bellman operator.

In practice, *Deep Q-Network* (DQN) (Mnih et al., 2015) approximates $\hat{Q}$ by a DNN $Q_\theta(s,a)$, and trains θ by minimizing the temporal-difference (TD) error:

$$\hat{\ell}(\theta) \;=\; \Big(r + \gamma \max_{a'} Q_{\theta^-}(s',a') - Q_\theta(s,a)\Big)^2,$$

where θ^- denotes a set of target-network parameters that are periodically copied from θ to stabilize training. Experience tuples (s,a,r,s') are stored in a replay buffer, and mini-batches are sampled from this buffer so that the DNN sees a diverse distribution of past experiences.

1.2 Distributional Reinforcement Learning

Distributional RL. Instead of focusing solely on the mean of $Z^\pi(s,a)$, *distributional RL* (DRL) (Bellemare et al., 2017, 2023) aims to model its entire distribution. That is, rather than seeking $Q^\pi(s,a) \;=\; \mathbb{E}\big[Z^\pi(s,a)\big]$, one seeks an estimate of the random variable $Z^\pi(s,a)$.

In standard Q-learning, one would write the Bellman operator in terms of expectations, but distributional RL generalizes this by writing a *distributional Bellman operator* $\mathscr{T}$ that satisfies

$$(\mathscr{T}Z)(s,a) \;\overset{D}{=}\; R(s,a) \;+\; \gamma Z\Big(s', \arg\max_{a'} \mathbb{E}_P[Z(s',a')]\Big),$$

where $\overset{D}{=}$ indicates equality in *distribution* and $s' \sim P(\cdot \mid s,a)$. This operator implicitly updates the entire distribution of returns, not just their mean.

The advantage of this perspective is that the variance, higher moments, or sometimes multimodality of returns can play a role in learning an optimal policy. For instance, distributional RL methods could capture more information about the reward, risk-sensitive behavior or achieve better exploration.

Categorical DQN (CDQN). Bellemare et al. (2017) introduced *Categorical DQN* (sometimes called C51), which represents the return distribution $Z(s,a)$ by a categorical distribution with fixed support or "atoms". Concretely, let $\{z_1, z_2, \ldots, z_M\} \subset \mathbb{R}$ be a set of fixed atoms, spaced along a pre-defined grid (e.g. from a minimum value $V_{\min}$ to a maximum value $V_{\max}$ with equal spacing). The agent's DNN outputs a set of vectors $\boldsymbol{\theta}(s,a) = \big(\theta_1(s,a), \ldots, \theta_M(s,a)\big)$, where each $\theta_i(s,a)$ is non-negative and

$$\sum_{i=1}^{M} \theta_i(s,a) \;=\; 1.$$

The return distribution is thus modeled as

$$Z_\theta(s,a) \;=\; \sum_{i=1}^{M} \theta_i(s,a)\, \delta_{z_i},$$

where δ_{z_i} denotes the Dirac delta at z_i. Training proceeds by applying a distributional variant of the TD error, effectively minimizing the Kullback–Leibler (KL) divergence or cross-entropy between the predicted distribution and a Bellman target. This approach have been reported to significantly outperforms standard DQN on many Arcade Learning Environment benchmarks because it captures the distribution shape (not just the mean) of returns.

Quantile-Based DQN. Another branch of distributional RL employs *quantile regression* to model $Z(s,a)$ (Dabney et al., 2018). Instead of discretizing the *value* domain via fixed atoms as in CDQN, these methods discretize the cumulative distribution function (CDF) by maintaining a finite set of quantile levels. Let $\tau_1, \tau_2, \ldots, \tau_N \in (0,1)$ be such levels, and also $\tau_i = \frac{i}{N}$. A DNN then outputs quantile locations $\big\{\theta_i(s,a)\big\}_{i=1}^{N}$, where each $\theta_i(s,a) \in \mathbb{R}$ corresponds to the estimated quantile of $Z(s,a)$ at level τ_i. Consequently, one can approximate the return distribution by the empirical measure $\hat{Z}(s,a) \;=\; \frac{1}{N}\sum_{i=1}^{N} \delta_{\theta_i(s,a)}$, treating each θ_i as the τ_i-quantile. A Huber-like quantile loss is used to push the current quantile estimates toward Bellman-updated target quantiles.

Moment Matching DQN (MMDQN). Recently, Nguyen-Tang et al. (2021) proposed *Moment Matching DQN (MMDQN)*. Here, the return distribution is modeled within RKHS via a finite set of particles $\{\tilde{z}_1, \ldots, \tilde{z}_M\}$. Unlike CDQN's fixed atoms, these particles are learnable, similarly to the quantile-based approach. Thus, each particle to have equal weight, yielding an empirical measure:

$$\hat{Z}(s,a) \;=\; \frac{1}{M} \sum_{m=1}^{M} \delta_{\tilde{z}_m(s,a)}.$$

The learning objective is derived based on MMD (Gretton et al., 2012):

$$\mathrm{MMD}\big(\hat{Z}(s,a),\ (\mathscr{T}\hat{Z})(s,a)\big),$$

where $\mathscr{T}\hat{Z}$ is the distributional Bellman target. Empirically, MMDQN has shown improved performance on some Atari benchmarks, demonstrating how nonparametric kernel embedding of the distribution can be leveraged in RL.

While Nguyen-Tang et al. (2021) did not explicitly formulate their approach based on CMEs, their formulation can be interpreted as a CME variant with uniform weights. We compare and contrast these approaches with our proposed method in section 2.3. We compare and contrast these approaches with our proposed method in section 2.3.

2 NK-CME based Distributional RL

2.1 Model and objectives

We propose modeling the (conditional) distribution of $Z(s,a)$ using our NN-CME hybrid approach. This approach leverages a kernel $k_{\mathscr{Z}}(\cdot,z) \in \mathscr{H}_{\mathscr{Z}}$ and represents distributions as:

$$\mu_{P(Z|S,A)}(x) = \sum_{a=1}^{M} k_{\mathscr{Z}}(\cdot,\eta_a) f_a(x;\theta),$$

where x is a tuple of state and action. Here, η_a are chosen analogously to atoms δ_a in CDQN, effectively covering anticipated support of the distribution. Importantly, our CME parameterization excels by bypassing the need for matrix inversion, enabling efficient action selection. In contrast, applying existing CME approaches (such as DF) in this setting would necessitate evaluating $\boldsymbol{\beta}(x)$ in (2.2) by accessing the entire replay buffer and performing matrix inversion on this data. This can be prohibitively expensive, especially when performed repeatedly.

To construct a loss function in the context of deep Q-learning, we define a metric between the distribution of $R(s,a) + \gamma Z(s',a')$ and that of $Z(s,a)$. Maximum Mean Discrepancy (MMD) (Gretton et al., 2012) provides a principled way to measure the discrepancy between these distributions in the RKHS:

$$\hat{d}_{k_{\mathscr{Z}}} = \left\| \sum_{a=1}^{M} k_{\mathscr{Z}}(\cdot,\tau_a) v_a - \sum_{a=1}^{M} k_{\mathscr{Z}}(\cdot,\eta_a) w_a \right\|_{\mathscr{H}_{\mathscr{Z}}},$$

where $\tau_a = r + \gamma\eta_a$ and $v_a = f_a(x;\theta^-)$. Squaring this MMD leads to our DRL-specific loss function $\hat{d}^2_{k_\mathscr{Z}}(\theta)$:

$$\hat{d}^2_{k_\mathscr{Z}}(\theta) = \sum_{a,b} k_\mathscr{Z}(\tau_a,\tau_b)v_a v_b - 2\sum_{a,b} k_\mathscr{Z}(\tau_a,\eta_b)v_a w_b + \sum_{a,b} k_\mathscr{Z}(\eta_a,\eta_b)w_a w_b.$$

Thus, we have retained the CME parametrization from chapter 4, but crucially adopted an MMD-based loss function to suit RL tasks. This flexible adaptation is enabled by both representing distributions as mean elements in the RKHS and exploiting the unique property of KMEs.

2.2 Fusing Kernels

Selecting the hyperparameters for $k_\mathscr{Z}$ is an important issue, also in RL settings. Corollary 4.1 of Killingberg and Langseth (2023) implies that careful selection of the bandwidth parameter is crucial for ensuring theoretical convergence in moment matching-based DRL, when used with the Gaussian kernel. Inspired by MMD-FUSE (Biggs et al., 2023), an MMD-based two-sample testing approach, we propose using a distribution over kernels, $k \in \mathscr{K}$. Our approach employs the following log-sum-exp type loss function:

$$\text{FUSE} = \log\left(\mathbb{E}_{k\sim\omega}\left[\exp(\hat{d}^2_{k_\mathscr{Z}}(\theta))\right]\right)$$

where ω is an element of $\mathscr{M}(\mathscr{K})$, the set of distributions over the kernels. For instance, we can define ω as a uniform distribution over collections of Gaussian kernels with various bandwidths $\sigma > 0$, where σ are chosen from a uniformly discretized interval. The Donsker-Varadhan equality (Donsker & Varadhan, 1975), as stated in Biggs et al. (2023), offers the following interpretation: $\text{FUSE} = \sup_\rho \mathbb{E}_{k\sim\rho}\left[\hat{d}_k(\theta)\right] - \text{KL}[\rho\|\omega]$, where KL denotes the Kullback-Leibler divergence, $\rho \in \mathscr{M}(\mathscr{K})$ can be loosely interpreted as the "posterior" over kernels, and ω as a "prior".

The intuitive mechanism of this fusing strategy is as follows: Corollary 4.1 of Killingberg and Langseth (2023) suggests that choosing large enough bandwidth is necessary to guarantee theoretical contraction. However, making it too large reduces the kernel's power to distinguish close samples. Intuitively, the fusion strategy calculates a soft maximum of MMD values with different bandwidths, ensuring MMD's test power by giving more weight to smaller bandwidths while

also mixing in larger bandwidths. While a theoretical analysis of this approach within the DRL context remains important future work, empirical results suggest its performance advantages compared to using a single fixed kernel.

2.3 Discussions

Both our approach and MMDQN utilize CMEs to model the distributions of $Z(s,a)$. However, they differ in what their NNs parameterize. MMDQN directly learns atoms through its network and assigns them uniform weights, resulting in the representation: $\mu_{P(Z|S,A)}(x) = \frac{1}{M}\sum_{a=1}^{M} k_{\mathscr{Z}}(g_a(x;\theta),\cdot)$. While this eliminates the need for manual atom specification, its reliance on uniform weights might hinder its ability to accurately model complex distributions exhibiting multimodality or skewness.

When the predefined atoms η_a in our approach coincide with the fixed atoms δ_a in CDQN, the parameterizations closely resemble each other. However, a key difference arises in the loss functions: CDQN employs cross-entropy loss, treating the distribution as categorical, whereas we leverage an MMD-based loss function. In essence, while our CME-based representation and MMD-based loss share similarities with MMDQN, our approach aligns more closely with CDQN in its atom structure and NN parameterization, but employs a distinct loss function.

3 Experimental Results

3.1 Setup

To investigate the unique aspects of our method and gain insights into DRL design choices, we tested it on three classic control environments from Gymnasium (Towers et al., 2023): CartPole, Acrobot, and MountainCar. We briefly describe each environment below:

CartPole-v1: An agent controls a cart on a frictionless track, aiming to balance a pole. The state space is four-dimensional, encompassing cart position, velocity, pole angle, and pole tip velocity. The agent can choose to push the cart left or right, receiving a $+1$ reward per balanced time step. Episodes terminate when the pole angle exceeds ± 12, the cart reaches the track edge, or the episode exceeds

500 steps.

Acrobot-v1: This environment features a two-linked pendulum with a controllable joint. The agent aims to swing the outer link's end to a specific height. The six-dimensional state describes joint angles and velocities, and the agent can apply no torque, torque left, or torque right. It receives a -1 reward per step before reaching the goal, and episodes end upon reaching the goal or exceeding 500 steps.

MountainCar-v0: This environment challenges an agent to drive an underpowered car up the mountain to the right. The agent must gain momentum by moving back and forth to achieve this goal. The state comprises the car's position and velocity, and the agent can push left, do nothing, or push right. Each step incurs a -1 reward until reaching the goal position, and episodes end upon reaching the goal or exceeding 200 steps.

While our approach does not inherently constrain the output of $f(x;\theta)$ to be positive or sum to one, we applied a softmax function in the last layer, analogous to CDQN, making the NN architecture identical to that of CDQN.

We also note that while DRL algorithms are often evaluated on complex Atari2600 games, such experiments require extensive computational resources. For instance, the work by (Obando-Ceron & Castro, 2021) required approximately 5 days to run each agent on each Atari environment using a GPU. As (Obando-Ceron & Castro, 2021) demonstrated through comprehensive benchmark evaluations, valuable scientific insights can still be drawn from smaller-scale environments. Therefore, our focus in this experiment is on understanding how different parameterizations and loss functions impact various aspects of learning and control effectiveness.

3.2 Results

Figure 6.1 presents the overall results, demonstrating the consistent effectiveness of our approach across three environments. In CartPole, all methods achieve near-optimal performance, attaining approximately 500 cumulative reward. However, our method demonstrates faster and more stable convergence. In Acrobot, while all methods exhibit stability, ours achieves the highest cumulative reward. On MountainCar, our approach clearly outperforms the others.

Importantly, despite identical distribution parameterization to CDQN, our approach consistently achieves superior performance. This suggests that the MMD-

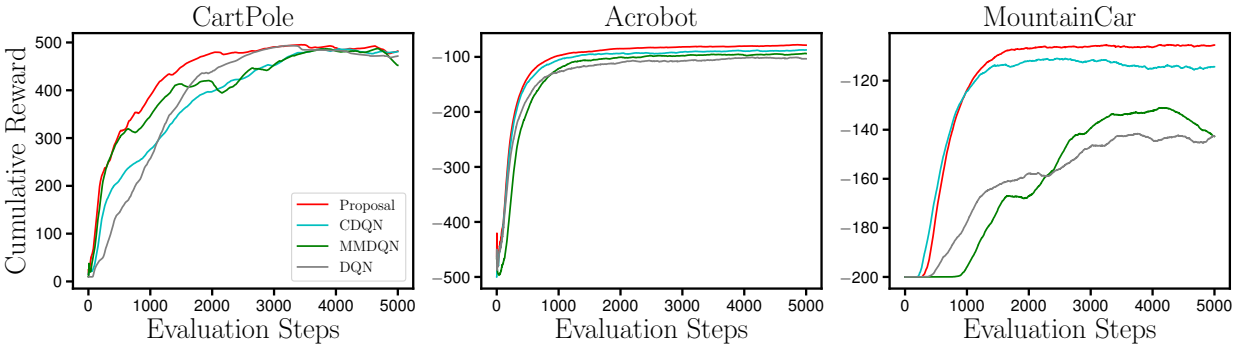

Figure 6.1: Performance comparison on three environments. We report the mean of cumulative rewards across 10 independent runs.

based loss function combined with our fusion strategy contributes to the improved performance. Investigating theoretical justification for this improvement may present a potentially fruitful research direction.

Despite improved performance reported by Nguyen-Tang et al. (2021) on Atari tasks, MMDQN displayed instability in classic control environments. We hypothesize that this disparity arises from the inherent differences in reward structures. Sparse rewards in Atari environments potentially favor MMDQN's direct particle placement learning, while predetermined atom structure might be more effective in classic control environments with denser rewards. This suggests that the choice of distribution parameterization may depend on the reward structure of environments, highlighting the influence of inductive bias.

4 Details on RL Experiments

4.1 Architecture and Hyperparameter Choices

We first describe architecture and hyperparameter choices shared across all methods: The NN architecture comprised two fully-connected hidden layers, each containing 50 ReLU activation units. For the optimizer, we used Adam (Kingma & Ba, 2015), and learning rates were adjusted for each environment: 1e-4 for CartPole and 1e-3 for Acrobot and MountainCar. A discount factor γ of 0.99 and batch size of 32 were used consistently. The replay buffer held 10,000 experiences, with

parameter updates occurring every 2 steps and target network updates every 100 steps (except where noted). For exploration, an ε-greedy policy with linear decay was implemented, starting ε with 1.0 and decaying to 0.01 over 10,000 steps. Finally, agents interact with the environment for a total of 500,000 steps, and were evaluated on an independent test episode every 100 steps with $\varepsilon = 0.001$. Also note that to isolate the influence of our method's design choices, we refrained from employing common RL techniques such as double Q-learning (van Hasselt et al., 2016) and prioritized experience replay (Schaul et al., 2016). Other architectural and hyperparameter choices for each model are provided below:

Proposal: We set η_a as uniform grid of 51 points, ranging from -100 to 100. Consequently, the final output layer dimension of NN becomes $51 \times |A|$ where $|A|$ represents the action space size. We also apply softmax fuction to normalize these 51 points associate with each action, which we find to result in improved performance. For kernel, we used Gaussian kernel. Following Biggs et al. (2023), the distribution over kernels ω was defined as a uniform distribution over collections of Gaussian kernels with 10 different bandwidths $\sigma > 0$. These bandwidths were selected from a uniform grid spanning the interval between half the 5th percentile and half the 95th percentile of the distance values $\{\|\eta_a - \eta'_a\|\}$.

CDQN: We set δ_a as uniform grid of 51 atoms, ranging from -100 to 100. Consequently, the final output layer dimension of NN becomes $51 \times |A|$. Softmax fuction is applied to normalize these 51 atoms associate with each action. For CartPole, we set the target update period to 1000, as this led to improved performance.

MMDQN: We set the number of particles learned by NN to 51. Consequently, the final output layer dimension of NN becomes $51 \times |A|$. For CartPole and MountainCar, we set the target update period to 1000, and also scale the reward by 0.1 (aiding particle learning by NN), as this led to improved performance. For kernel, we followed Nguyen-Tang et al. (2021) and used (uniform) mixture of 10 Gaussian kernels, with bandwidths set as uniformly spaced values between 1 and 100. When the reward was scaled by 0.1, this bandwidth range was adjusted to 1 to 10.

DQN: Final output layer dimension of the NN is $|A|$.

4.2 Additional Results on Proposed Model

To investigate the impact of our fusing strategy (Proposal-Fuse), we conducted a comparative analysis with a variant of our proposal that employs a single band-

width parameter (Proposal-Single). We used the same architecture and hyperparameter as CDQN, and we set the bandwidth to 10. The result shown in Figure 6.2 demonstrates a significant performance improvement when using the fusing strategy. Without the fusing, Proposal-Single is slightly worse than CDQN on CartPole and MountainCar. This highlights both the efficacy of the fusing approach and the critical role of kernel hyperparameter selection in DRL contexts.

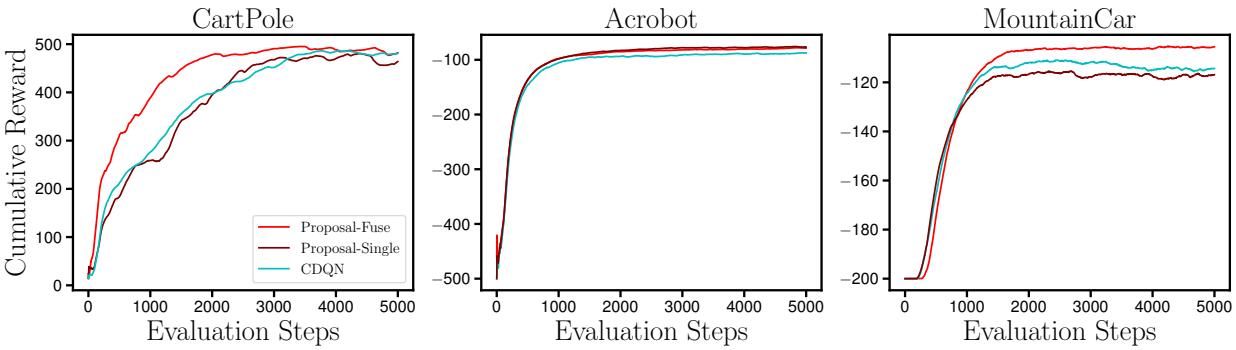

Figure 6.2: Comparison of our proposed methods using single and fused kernels. We report the mean of cumulative rewards across 10 independent runs.

Chapter 7

Conclusions

1 Summary

This thesis set out to extend CMEs for greater effectiveness in challenging, potentially large-scale and high-dimensional machine learning tasks. To do so, we first identified the key limitations of conventional CMEs—scalability, expressiveness, and output kernel hyperparameter tuning—and then advanced the framework with Neural-Kernel Conditional Mean Embeddings (NK-CMEs), which integrate deep learning into the CME paradigm.

NK-CMEs replaced the costly Gram matrix inversion with an expressive deep neural network while retaining the output variable kernel for representing distributions. This new framework thus aimed to address both scalability and expressiveness without sacrificing the attractive properties classical CMEs built on positive definite kernels.

To tackle the nontrivial challenge of output kernel hyperparameter tuning, we introduced a Gaussian density kernel and devised two learning schemes—an iterative two step procedure and a fully joint optimization—to optimize the kernel bandwidth alongside the neural network parameters. We demonstrated that NK-CMEs, when combined with these optimization strategies can represent complex conditional distributions and achieved competitive, often superior, performance on synthetic and real world density estimation benchmarks, when compared with other deep learning based methods.

Finally, we showed that NK-CMEs naturally integrate into distributional re-

inforcement learning, yielding a novel algorithm that outperformed existing approaches in terms of cumulative rewards. This highlighted the versatility and practical impact of NK-CMEs to challenging tasks that were previously out of reach for classic CMEs.

2 Future Works

We list limitations of this study and promising future work directions:

- While our approach effectively handled one-dimensional outputs, verifying the effectiveness with more challenging multidimensional settings presents an important area for future investigation. We believe the biggest challenge in multidimensional settings is the choice of location points. A straightforward way may be to use the k-means clustering method on training data and use the obtained clusters as the location points. More sophisticated approaches may be to optimize these points, similar to inducing point methods in Gaussian Process literature (Hensman et al., 2013). However, it is well-known in the GP community that choosing the number of inducing points, their initialization, and optimizing them with gradient-based methods present significant challenges. We believe recent developments in this topic (Burt et al., 2020) may be helpful in our future work.
- Our approach seamlessly lends itself to diverse other settings, including causal inference tasks where using CMEs offers distinct advantages (Chau et al., 2021; Muandet et al., 2021; Park et al., 2021; Singh et al., 2019). These works utilize CMEs as the core foundation for tasks such as estimating counterfactual distributions and distributional treatment effects. Our method can be readily integrated into these settings by replacing classical CMEs. Importantly, our novel technique for optimizing hyperparameters on k_y would benefit any CME application.

References

Bellemare, M. G., Dabney, W., & Munos, R. (2017). A distributional perspective on reinforcement learning. *International Conference on Machine Learning*, 449–458.

Bellemare, M. G., Dabney, W., & Rowland, M. (2023). *Distributional reinforcement learning* [http://www.distributional-rl.org]. MIT Press.

Biggs, F., Schrab, A., & Gretton, A. (2023). Mmd-fuse: Learning and combining kernels for two-sample testing without data splitting. *Advances in Neural Information Processing Systems*, *36*, 75151–75188.

Bishop, C. M. (1994). *Mixture density networks* [Report NCRG/94/004].

Bishop, C. M. (2006). *Pattern recognition and machine learning*. Springer.

Bochner, S. (1933). Monotone funktionen, stieltjessche integrale und harmonische analyse. *Mathematische Annalen*, *108*, 378–410.

Burt, D. R., Rasmussen, C. E., & van der Wilk, M. (2020). Convergence of sparse variational inference in gaussian processes regression. *Journal of Machine Learning Research*, *21*(131), 1–63.

Chau, S. L., Ton, J.-F., Gonzalez, J., Teh, Y. W., & Sejdinovic, D. (2021). BayesIMP: Uncertainty Quantification for Causal Data Fusion. *Advances in Neural Information Processing Systems (NeurIPS)*, *34*, 3466–3477.

Chen, Y., Welling, M., & Smola, A. (2010). Super-samples from kernel herding. *Proceedings of the 26th Conference on Uncertainty in Artificial Intelligence*, 109–116.

Chen, Z., Zhang, K., Chan, L., & Schölkopf, B. (2014). Causal discovery via reproducing kernel hilbert space embeddings. *Neural Computation*, 1484–1517.

Chizat, L., Oyallon, E., & Bach, F. (2019). On lazy training in differentiable programming. *Advances in Neural Information Processing Systems (NeurIPS)*, 2933–2943.

Dabney, W., Rowland, M., Bellemare, M., & Munos, R. (2018). Distributional reinforcement learning with quantile regression. *Proceedings of the AAAI Conference on Artificial Intelligence*, *32*.

Dalmasso, N., Pospisil, T., Lee, A. B., Izbicki, R., Freeman, P. E., & Malz, A. I. (2019). Conditional density estimation tools in python and r with applications to photometric redshifts and likelihood-free cosmological inference. https://arxiv.org/abs/1908.11523

Donsker, M. D., & Varadhan, S. R. S. (1975). Asymptotic evaluation of certain markov process expectations for large time. *Communications on Pure and Applied Mathematics*, *28*(1), 1–47.

Drineas, P., & Mahoney, M. W. (2005). On the nystrom method for approximating a gram matrix for improved kernel-based learning. *Journal of Machine Learning Research*, *6*(72), 2153–2175.

Dua, D., & Graff, C. (2017). Uci machine learning repository. http://archive.ics.uci.edu/ml

Durkan, C., Bekasov, A., Murray, I., & Papamakarios, G. (2019). Neural spline flows. *Advances in Neural Information Processing Systems*, 7509–7520.

Engle, R. F. (1982). Autoregressive conditional heteroskedasticity with estimates of the variance of united kingdom inflation. *Econometrica*, 987–1007.

Fan, J., Yao, Q., & Tong, H. (1996). Estimation of conditional densities and sensitivity measures in nonlinear dynamical systems. *Biometrika*, *83*, 189–206.

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

Fukumizu, K., Gretton, A., Sun, X., & Schölkopf, B. (2008). Kernel measures of conditional dependence. *Advances in Neural Information Processing Systems*, 489–496.

Fukumizu, K., Song, L., & Gretton, A. (2013). Kernel Bayes' rule: Bayesian inference with positive definite kernels. *Journal of Machine Learning Research*, *14*(82), 3753–3783.

Gärtner, T. (2003). A survey of kernels for structured data. *SIGKDD Explorations Newsletter*.

Girosi, F., Jones, M., & Poggio, T. (1995). Regularization theory and neural networks architectures. *Neural Computation*, *7*(2), 219–269. https://doi.org/10.1162/neco.1995.7.2.219

Gloeckler, M., Deistler, M., Weilbach, C. D., Wood, F., & Macke, J. H. (2024). All-in-one simulation-based inference. In R. Salakhutdinov, Z. Kolter, K. Heller, A. Weller, N. Oliver, J. Scarlett, & F. Berkenkamp (Eds.), *Proceedings of the 41st international conference on machine learning* (pp. 15735–15766).

Goodfellow, I., Bengio, Y., & Courville, A. (2016). *Deep learning*. MIT Press.

Gretton, A., Borgwardt, K. M., Rasch, M. J., Schölkopf, B., & Smola, A. (2012). A kernel two-sample test. *Journal of Machine Learning Research*, *13*(25), 723–773.

Gretton, A., Herbrich, R., Smola, A., Schölkopf, B., & Hyvärinen, A. (2005). Kernel methods for measuring independence. *Journal of Machine Learning Research*, *6*, 2075–2129.

Grünewälder, S., Lever, G., Baldassarre, L., Patterson, S., Gretton, A., & Pontil, M. (2012). Conditional mean embeddings as regressors. *Proceedings of the 29th International Conference on Machine Learning*, 1823–1830.

Han, X., Zheng, H., & Zhou, M. (2022). CARD: Classification and regression diffusion models. *Advances in Neural Information Processing Systems*.

Hensman, J., Fusi, N., & Lawrence, N. D. (2013). Gaussian processes for big data. *Proceedings of the Conference on Uncertainty in Artificial Intelligence*, 282–290.

Ho, J., Jain, A., & Abbeel, P. (2020). Denoising diffusion probabilistic models. *Advances in Neural Information Processing Systems (NeurIPS)*, *33*, 6840–6851.

Hsu, K., & Ramos, F. (2019). Bayesian learning of conditional kernel mean embeddings for automatic likelihood-free inference. *Proceedings of the Twenty-Second International Conference on Artificial Intelligence and Statistics*, *89*, 2631–2640.

Imaizumi, M., & Fukumizu, K. (2019). Deep neural networks learn non-smooth functions effectively. *Proceedings of the Twenty-Second International Conference on Artificial Intelligence and Statistics*, *89*, 869–878.

Jacobs, R. A., Jordan, M. I., Nowlan, S. J., & Hinton, G. E. (1991). Adaptive mixtures of local experts. *Neural Computation*, *3*, 79–87.

Jacot, A., Gabriel, F., & Hongler, C. (2018). Neural tangent kernel: Convergence and generalization in neural networks. *Advances in Neural Information Processing Systems*.

Kanagawa, M., & Fukumizu, K. (2014). Recovering Distributions from Gaussian RKHS Embeddings. In S. Kaski & J. Corander (Eds.), *Proceedings of the seventeenth international conference on artificial intelligence and statistics* (pp. 457–465, Vol. 33). PMLR.

Killingberg, L., & Langseth, H. (2023). The multiquadric kernel for moment-matching distributional reinforcement learning. *Transactions on Machine Learning Research*.

Kim, J., & Scott, C. D. (2012). Robust kernel density estimation. *Journal of Machine Learning Research*, *13*(82), 2529–2565.

Kingma, D. P., & Ba, J. (2015). Adam: A method for stochastic optimization. *International Conference on Learning Representations (ICLR)*.

Lee, J., Bahri, Y., Novak, R., Schoenholz, S. S., Pennington, J., & Sohl-Dickstein, J. (2018). Deep neural networks as gaussian processes. *International Conference on Learning Representations*.

Li, Z., Meunier, D., Mollenhauer, M., & Gretton, A. (2022). Optimal rates for regularized conditional mean embedding learning. In A. H. Oh, A. Agarwal, D. Belgrave, & K. Cho (Eds.), *Advances in neural information processing systems*.

Loshchilov, I., & Hutter, F. (2019). Decoupled weight decay regularization. *International Conference on Learning Representations*.

Lueckmann, J.-M., Boelts, J., Greenberg, D., Goncalves, P., & Macke, J. (2021). Benchmarking simulation-based inference. *Proceedings of The 24th International Conference on Artificial Intelligence and Statistics*, *130*, 343–351.

Makansi, O., Ilg, E., Çiçek, Ö., & Brox, T. (2019). Overcoming limitations of mixture density networks: A sampling and fitting framework for multimodal future prediction. *IEEE International Conference on Computer Vision and Pattern Recognition*.

Meinshausen, N. (2006). Quantile regression forests. *Journal of Machine Learning Research*, *7*(35), 983–999. http://jmlr.org/papers/v7/meinshausen06a.html

Mercer, J. (1909). Functions of positive and negative type, and their connection with the theory of integral equations. *Philosophical Transactions of the Royal Society of London. Series A*, *209*, 415–446.

Miyato, T., Kataoka, T., Koyama, M., & Yoshida, Y. (2018). Spectral normalization for generative adversarial networks. *International Conference on Learning Representations*.

Mnih, V., Kavukcuoglu, K., Silver, D., Rusu, A. A., Veness, J., Bellemare, M. G., Graves, A., Riedmiller, M., Fidjeland, A. K., Ostrovski, G., et al. (2015). Human-level control through deep reinforcement learning. *Nature*, *518*(7540), 529–533.

Muandet, K., Kanagawa, M., Saengkyongam, S., & Marukatat, S. (2021). Counterfactual mean embeddings. *Journal of Machine Learning Research*, *22*(162), 1–71.

Muandet, K., Fukumizu, K., Sriperumbudur, B., Schölkopf, B., & Gretton, A. (2017). Kernel mean embedding of distributions: A review and beyond. *Foundations and Trends® in Machine Learning*, *10*(1-2), 1–141.

Nguyen-Tang, T., Gupta, S., & Venkatesh, S. (2021). Distributional reinforcement learning via moment matching. *Proceedings of the AAAI Conference on Artificial Intelligence*, *35*(10), 9144–9152.

Obando-Ceron, J. S., & Castro, P. S. (2021). Revisiting rainbow: Promoting more insightful and inclusive deep reinforcement learning research. *Proceedings of the 38th International Conference on Machine Learning*.

Papamakarios, G., & Murray, I. (2016). Fast ε-free inference of simulation models with bayesian conditional density estimation. *Advances in Neural Information Processing Systems*, *29*, 1028–1036.

Papamakarios, G., Nalisnick, E., Rezende, D. J., Mohamed, S., & Lakshminarayanan, B. (2021). Normalizing flows for modeling and inference. *Journal of Machine Learning Research*, *22*(57), 1–64.

Park, J., Shalit, U., Schölkopf, B., & Muandet, K. Conditional distributional treatment effect with kernel conditional mean embeddings and u-statistic regression. In: In *Proceedings of 38th international conference on machine learning*. *139*. Proceedings of Machine Learning Research. PMLR, 2021, 8401–8412.

Park, J., & Muandet, K. (2020). A measure-theoretic approach to kernel conditional mean embeddings. *Advances in Neural Information Processing Systems*, *33*, 21247–21259.

Puterman, M. L. (2014). *Markov decision processes: Discrete stochastic dynamic programming*. John Wiley & Sons.

Rahimi, A., & Recht, B. (2007). Random features for large-scale kernel machines. *Advances in Neural Information Processing Systems*, *20*.

Rasmussen, C. E., & Williams, C. K. I. (2006). *Gaussian processes for machine learning*. MIT Press.

Rezende, D. J., & Mohamed, S. (2015). Variational inference with normalizing flows. *Proceedings of The 32nd International Conference on Machine Learning*, *37*, 1530–1538.

Rosenblatt, M. (1969). Conditional probability density and regression estimates. In *Multivariate analysis ii* (pp. 25–31). Academic Press.

Saitoh, S. (1997). *Integral transforms, reproducing kernels and their applications*. Chapman; Hall/CRC.

Schaul, T., Quan, J., Antonoglou, I., & Silver, D. (2016). Prioritized experience replay. *International Conference on Learning Representations*.

Schölkopf, B., & Smola, A. J. (2001). *Learning with kernels: Support vector machines, regularization, optimization, and beyond*. MIT Press.

Shimizu, E., Fukumizu, K., & Sejdinovic, D. (2024). Neural-kernel conditional mean embeddings. *Forty-first International Conference on Machine Learning*.

Singh, R., Sahani, M., & Gretton, A. (2019). Kernel instrumental variable regression. *Advances in Neural Information Processing Systems*, *32*, 4595–4607.

Song, L., Fukumizu, K., & Gretton, A. (2013). Kernel embeddings of conditional distributions: A unified kernel framework for nonparametric inference in graphical models. *IEEE Signal Processing Magazine*, *30*(4), 98–111.

Song, L., Huang, J., Smola, A., & Fukumizu, K. (2009). Hilbert space embeddings of conditional distributions with applications to dynamical systems. *International Conference on Machine Learning*, 961–968.

Sriperumbudur, B. K., Gretton, A., Fukumizu, K., Schölkopf, B., & Lanckriet, G. R. (2010). Hilbert space embeddings and metrics on probability measures. *Journal of Machine Learning Research*, *11*, 1517–1561.

Steinwart, I., & Scovel, C. (2012). Mercer's theorem on general domains: On the interaction between measures, kernels, and rkhss. *Constructive Approximation*, *35*(3), 363–417.

Stimper, V., Liu, D., Campbell, A., Berenz, V., Ryll, L., Schölkopf, B., & Hernández-Lobato, J. M. (2023). Normflows: A pytorch package for normalizing flows. *Journal of Open Source Software*, *8*(86), 5361.

Sugiyama, M., Takeuchi, I., Suzuki, T., Kanamori, T., Hachiya, H., & Okanohara, D. (2010). Conditional density estimation via least-squares density ratio estimation. *Proceedings of the Thirteenth International Conference on Artificial Intelligence and Statistics*, *9*, 781–788.

Suzuki, T. (2019). Adaptivity of deep relu network for learning in besov and mixed smooth besov spaces: Optimal rate and curse of dimensionality. *7th International Conference on Learning Representations, ICLR 2019*.

Talwai, P., Shameli, A., & Simchi-Levi, D. (2022). Sobolev norm learning rates for conditional mean embeddings. *Proceedings of The 25th International Conference on Artificial Intelligence and Statistics*, 10422–10447.

Towers, M., Terry, J. K., Kwiatkowski, A., Balis, J. U., Cola, G. d., Deleu, T., Goulão, M., Kallinteris, A., KG, A., Krimmel, M., Perez-Vicente, R., Pierré, A., Schulhoff, S., Tai, J. J., Shen, A. T. J., & Younis, O. G. (2023). Gymnasium. https://zenodo.org/record/8127025

van Hasselt, H., Guez, A., & Silver, D. (2016). Deep reinforcement learning with double q-learning. *Proceedings of the Thirtieth AAAI Conference on Artificial Intelligence*, 2094–2100.

Virtanen, P., Gommers, R., Oliphant, T. E., Haberland, M., Reddy, T., Cournapeau, D., Burovski, E., Peterson, P., Weckesser, W., Bright, J., van der Walt, S. J., Brett, M., Wilson, J., Millman, K. J., Mayorov, N., Nelson, A. R. J., Jones, E., Kern, R., Larson, E., ... SciPy 1.0 Contributors. (2020). SciPy 1.0: Fundamental Algorithms for Scientific Computing in Python. *Nature Methods*, *17*, 261–272.

Watkins, C. J., & Dayan, P. (1992). Q-learning. *Machine learning*, *8*(3-4), 279–292.

Xu, L., Chen, Y., Doucet, A., & Gretton, A. (2022). Importance weighted kernel Bayes' rule. *Proceedings of the 39th International Conference on Machine Learning*, *162*, 24524–24538.

Xu, L., Chen, Y., Srinivasan, S., de Freitas, N., Doucet, A., & Gretton, A. (2021). Learning deep features in instrumental variable regression. *International Conference on Learning Representations*.

Xu, L., & Gretton, A. (2023). A neural mean embedding approach for back-door and front-door adjustment. *The Eleventh International Conference on Learning Representations*.

Xu, L., & Gretton, A. (2025). Kernel single proxy control for deterministic confounding. *The 28th International Conference on Artificial Intelligence and Statistics*.

Yang, G. (2019). Tensor programs i: Wide feedforward or recurrent neural networks of any architecture are gaussian processes. *CoRR*, *abs/1910.12478*. http://arxiv.org/abs/1910.12478