Expanding the Scope of Random Feature Models: Theory and Applications

by

Esha Saha

A thesis
presented to the University of Waterloo
in fulfillment of the
thesis requirement for the degree of
Doctor of Philosophy
in
Applied Mathematics

Waterloo, Ontario, Canada, 2023

Examining Committee Membership

The following served on the Examining Committee for this thesis. The decision of the Examining Committee is by majority vote.

External Examiner:	Ben Adcock Professor, Department of Mathematics Simon Fraser University
Supervisor(s):	Giang Tran Assistant Professor, Department of Applied Mathematics University of Waterloo
Internal Member(s):	Hans De Sterck Professor, Department of Applied Mathematics University of Waterloo
	Jun Liu Associate Professor, Department of Applied Mathematics University of Waterloo
Internal-External Member:	Zhou Wang Professor, Department of Electrical and Computer Engineering University of Waterloo

Statement of Contributions

Esha Saha was the sole author of Chapters 1, 2, and 6, which were written under the supervision of Dr. Giang Tran and were not written for publication. The research presented in Chapters 3 and 4 has been published in peer-reviewed journals, and the research presented in Chapter 5 has been submitted and is under review. Esha Saha's contributions to Chapters 3, 4 and 5 are detailed below.

Chapter 3: The research outcomes presented in this chapter are based on a collaboration between Esha Saha, Dr. Giang Tran, and Dr. Hayden Shaeffer. It has been published in [123]. Dr. Giang Tran and Dr. Hayden Shaeffer formulated the problem with some theoretical derivations. Esha Saha worked on some theoretical derivations and performed all the numerical simulations. The manuscript was jointly written and edited by Esha Saha, Dr. Giang Tran, and Dr. Hayden Shaeffer.

Chapter 4: The research presented in this chapter is a product of a collaboration between Esha Saha, Dr. Giang Tran, and Dr. Lam Si Tung Ho. It has been published in [122]. The project was formulated jointly by Esha Saha, Dr. Giang Tran and, Dr. Lam Si Tung Ho. Esha Saha performed all the numerical simulations. The manuscript was written by Esha Saha. Editing and verification of results were done jointly by Dr. Giang Tran and Dr. Lam Si Tung Ho.

Chapter 5: The research presented in this chapter is based on a collaboration between Esha Saha and Dr. Giang Tran. The manuscript is available at [124] and is under review. The problem was jointly formulated by Esha Saha and Dr. Giang Tran. Esha Saha worked on the theoretical formulation, and proofs and performed all numerical simulations. The manuscript was written by Esha Saha. Editing and proof verification was provided by Dr. Giang Tran.

Abstract

Data, defined as facts and statistics collected together for analysis is at the core of every inference or decision made by any living organism. Right from the time we are born, our brain collects data from everything that is happening around us and helps us to make decisions based on past experiences. With the advent of technology, humans have been trying to develop methods that can learn from data and generalize well based on past information. While this attempt has been greatly successful with the development of the machine learning community, one parallel field that also developed along with it is the need to have a theoretical understanding of these methods. It is important to understand the workings of the algorithms to be able to quantify the cause and nature of the error they can make so that informed decisions can be made using these results, especially for sensitive applications such as in the medical field.

At the heart of these methods lies the mathematical formulation and analysis of such learning algorithms. One such method that particularly caught the attention of researchers recently is the random feature model (RFM), introduced for reducing the complexity and faster computation of kernel methods in large-scale machine learning algorithms. These classes of methods can provide theoretical interpretations and have the potential to perform well numerically, thus being more reliable than black box methods such as deep neural networks. This thesis aims to explore RFMs by expanding their theory and applications in the machine-learning community. We begin our exploration by developing a fast algorithm for high dimensional function approximation using a random feature-based surrogate model. Assuming the target function is a lower-order additive function, we incorporate sparsity as a side information within our model to get numerical results that are better (or comparable) to other well-known methods and also provide risk and error bounds for our model. Extending the idea of learning functions, we build a model to learn and predict the dynamics of an epidemic from incomplete and scarce data. This model combines the idea of random feature approximation with the use of Takens' delay embedding theorem on the given input data. RFMs have majorly been explored in a form that resembles a shallow neural network with fixed hidden parameters. In our third project, motivated to work on the idea of multiple layers in an RFM, we propose an interpretable RFM whose architecture is inspired by diffusion models. We make the model interpretable by providing error bounds on the sampled data from its true distribution and show numerically that the proposed model is capable of generating images from data as well as denoising it.

Acknowledgements

I would like to start by thanking all those whom I look up to as my mentors. First and foremost, my deepest gratitude towards my advisor, Dr. Giang Tran who has been an incredible mentor to me. This thesis would not have been possible without her constant support, dedication, and mentoring. To my collaborators, Dr. Hayden Shaeffer and Dr. Lam Ho, thank you for your faith in me and for all the hard work to help me in my journey. Thank you to my Master's professor Dr. Debajyoti Choudhuri and my internship mentor Dr. A. S. Vasudeva Murthy (1957-2021) for cultivating an interest in research and always encouraging me to do my best. To my high school teacher, Ms. Tarang Chaudhry thank you for making me fall in love with Mathematics. Finally, my acknowledgments towards my mentors would be incomplete without thanking my parents, Dr. Subir Kumar Saha and Ms. Bulu Saha, for always reading and providing insightful feedback on my work. None of this would have been possible without their constant encouragement, inspiration, and support.

To Nandan, thank you for always being there. This journey would not have been half as fun without you. I will forever remain indebted to you for motivating me every day to go to the office whenever I felt lazy, for listening to my rants and frustrations, putting up with my childlike tantrums, and for always supporting me in my decisions even when they were unfavorable for you. I sincerely hope I can do the same for you one day. A big part of my success is owed to you.

A big thank you to Ala', from being the first person I met on campus to the constant friend at Waterloo, you will always be the most beautiful and smart friend I ever made. From attending classes together, going for coffee breaks every two hours, showing up for all the events over the campus and finally switching our offices frequently to work together, half of my time at the university belongs to you. To Kanika, for always spending hours on the phone with me even though it was midnight in your timezone, thank you for always motivating me to work harder. Thank you to Stephanie for always being the rational friend to guide us whenever we wanted to make life decisions emotionally. To Vipin and Saikiran, thank you for always making me feel confident whenever I would feel otherwise. To my extended family members in Toronto and Mississauga, thank you for letting me be a part of all your gatherings and never letting me feel like I was away from family.

I would also like to thank my committee members, Dr. Hans De Sterck, Dr. Jun Liu, Dr. Ben Adcock, and Dr. Zhou Wang, for their valuable time, feedback, and suggestions. Thank you to Maureen, Erin, Alicia, Samanthi, Nadia, and Hayleigh for always providing me with the resources and information needed at each and every step of my journey. Thank you to all my colleagues in and beyond Waterloo for all our interactions which have been invaluable to me.

Finally, thank you to Springer Nature for granting permission to use my research articles in the thesis.

Dedication

To all my teachers.

Table of Contents

List of Figures

List of Tables

Notations

Notations		Description
$\mathbf{a}, \cdots, \mathbf{z}$	:	Column vectors
$\mathbf{A}, \cdots, \mathbf{Z}$	:	Matrices
$\mathbf{A}^\star, \cdots, \mathbf{Z}^\star$	:	Complex conjugate of matrices
$\mathbf{A}^{-1}, \cdots, \mathbf{Z}^{-1}$		Inverse of matrices
$\mathbf{A}^{+}, \cdots, \mathbf{Z}^{+}$		Pseudo-inverse of matrices
$\mathbf{I}_n$	:	Identity matrix of size $n \times n$
$\mathbb{R}$	:	Set of real numbers
$\mathbb{C}$	:	Set of complex numbers
$[N]$	:	Set of integers upto N
$\lvert \cdot \rvert$	:	Absolute value
$\langle \cdot, \cdot \rangle$	:	Inner product
$\lVert \cdot \rVert$	:	General norm
$\lVert \mathbf{x} \rVert_0$	:	ℓ_0 norm of $\mathbf{x}$ defined by number of non-zero components in $\mathbf{x}$
$\lVert \mathbf{x} \rVert_1$	:	ℓ_1 norm of $\mathbf{x}$ defined by $\sum_{i=1}^{d} \lvert x_i \rvert$ where $\mathbf{x} = [x_1, \cdots, x_d]^T$
$\lVert \mathbf{x} \rVert_2$	:	ℓ_2 norm of $\mathbf{x}$ defined by $\sum_{i=1}^{d} \lvert x_i \rvert^2$ where $\mathbf{x} = [x_1, \cdots, x_d]^T$
$\lVert \mathbf{x} \rVert_\infty$	:	ℓ_∞ norm of $\mathbf{x}$ defined by $\max_{1 \le i \le d} \{\lvert x_i \rvert\}$ where $\mathbf{x} = [x_1, \cdots, x_d]^T$
$\lVert \mathbf{A} \rVert_{2\to 2}$	:	The operator norm (largest singular value) of a matrix $\mathbf{A}$ on ℓ_2
$\mathrm{card}(\cdot)$	:	Cardinality of the set i.e., the number of elements in a set
$\mathrm{supp}(\mathbf{x})$	:	Support of a vector $\mathbf{x} = [x_1, \cdots, x_d]^T$ given by $\{j \in [N] : x_j \neq 0\}$
$\int q(\mathbf{x}_{0:K}) d\mathbf{x}_{1:K}$	:	$\int q(\mathbf{x}_0, \cdots, \mathbf{x}_K) d\mathbf{x}_1 \cdots d\mathbf{x}_K$

Abbreviations

Abbreviations		**Description**
BIC	:	Bayesian Information Criterion
BPDN	:	Basis Pursuit Denoising Problem
CNN	:	Convolutional Neural Network
DNN	:	Deep Neural Network
DRFM	:	Diffusion Random Feature Model
HARFE	:	Hard-Ridge Random Feature Expansion
HTP	:	Hard Thresholding Pursuit
IHT	:	Iterative Hard Thresholding
LASSO	:	Least Absolute Value Shrinkage and Selection Operator
LLM	:	Large Language Model
NN	:	Neural Network
ODE	:	Ordinary Differential Equation
PCA	:	Principal Component Analysis
QCBP	:	Quadratically Controlled Basis Pursuit
ReLU	:	Rectified Linear Unit
ResNet	:	Residual Network
RFM	:	Random Feature Model
RFRR	:	Random Feature Ridge Regression
RKHS	:	Reproducing Kernel Hilbert Space
SDE	:	Stochastic Differential Equation
SPADE4	:	Sparsity and Delay Embedding Based Forecasting of Epidemics
SVM	:	Support Vector Machine

Chapter 1

Introduction

Since the origination of the term "machine learning" by Arthur Samuel, an IBM employee in 1959, it has been an area of research interest and development. Machine learning algorithms aim to build a model for prediction or decision-making by learning from sample data. Over the years, there has been an exponential increase in the development of data-driven models and methods in machine learning research, with applications in several diverse fields such as physics, chemistry, biology, economics, and engineering. Deep Neural Networks (DNNs) [86, 127] have become the building blocks for many data-driven models. The model architectures can range from simple feed-forward neural networks [9, 137] to complex architectures such as Convolutional Neural Networks (CNNs) [56, 64, 88, 114, 118], Residual Network (ResNet) [68, 92, 150], Large Language Models (LLMs) [23, 37, 39, 140, 145], etc. Other well-known data-driven methods include algorithms such as Sparse Identification of Nonlinear Dynamics (SINDy) [28, 154], Polynomial Chaos Expansion [1, 50, 126], etc. The success of these data-driven techniques may lead us to believe that the availability of data is enough to solve problems. However, if the training data is scarce or of poor quality (for example, noisy or incomplete data), the data-driven methods may fail or give misleading outputs. Moreover, most of the DNN-based models are black boxes and hence have very little interpretability. On the other hand, classical machine learning algorithms such as Support Vector Machines (SVMs) [33, 41], nearest neighbor [55, 139], decision trees [80, 82], Principle Component Analysis (PCA) [21, 146], etc. are interpretable since they have been developed from an understanding of the feature space, but underperform for complex tasks. Thus, there is a need to develop models and algorithms that are both interpretable and can serve as an alternative to DNNs for scientific machine-learning tasks.

We are particularly interested in looking at a class of models called random feature models (RFMs) [109–111], which are closely related to kernel methods. This research is based on understanding and expanding the scope of RFMs. We develop algorithms for applications in function approximation and epidemic prediction. The research also proposes a diffusion model-inspired interpretable RFM. In the remainder of this chapter, we recall random feature models and summarize existing literature on function approximation in Section 1.1, followed by some preliminary discussion on learning dynamical systems in

Section 1.2 and an introduction to diffusion models in Section 1.3. We also discuss the objectives and contributions of this thesis and provide an overview of the coming chapters in Section 1.4.

1.1 Random Feature Model

A complex-valued function approximation using RFM takes the form $y = \mathbf{c}^T \phi(\mathbf{W}^T \mathbf{x} + \mathbf{b}) \in \mathbb{C}$ where $\mathbf{x} \in \mathbb{R}^d$ is the input data, $\mathbf{W} \in \mathbb{R}^{d \times N}$ (N denotes the number of random features) is a random weight matrix, $\mathbf{b} \in \mathbb{R}^N$ is the bias vector, ϕ is a predefined non-linear function applied element-wise and $\mathbf{c} \in \mathbb{C}^N$ is the output layer vector. For an RFM, the output layer $\mathbf{c}$ is trained, while the hidden layer weights $\mathbf{W}$ and bias $\mathbf{b}$ are fixed. From the neural network perspective, an RFM is a two-layer network with a randomized but fixed hidden layer [109–111] as depicted in Figure 1.1.

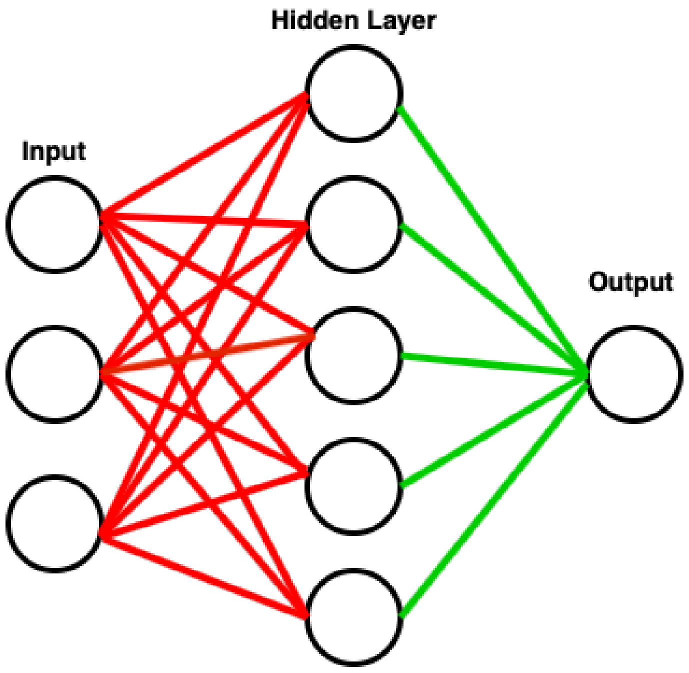


Figure 1.1: Depiction of a random feature model. The red lines denote the components of the (hidden) weight matrix which are randomly initialized and fixed. The green lines denote the components of the trainable (output) vector $\mathbf{c}$.

Random Feature Models and Regression

The motivation of random features lies in simplifying kernel methods that use a pre-defined basis (often nonlinear) in the form of a kernel $K(\mathbf{x}, \mathbf{y})$. We discuss the connection between RFMs and kernel methods in detail in Chapter 2. Minimization for kernel training problems can be formulated by using linear combinations of the kernel basis on the dataset [30, 69, 131, 157]. These methods need the kernel K to be applied to every pair of data samples, thus limiting their applications, especially in large-scale settings. For example, in [109] the authors show how to construct feature spaces that can uniformly approximate shift-invariant kernels $K(\mathbf{x}-\mathbf{y})$ to within ϵ with dimensions of $\mathcal{O}(d\epsilon^2 \log \frac{1}{\epsilon^2})$. The RFM [109–111] is a technique to avoid the computation cost of fully evaluating the kernel.

Learning the parameters of RFMs can be done using convex optimization techniques as training happens only over the last layer and can be solved similarly to regression problems or their variations. Thus, given a collection of m measurements, whose inputs (and outputs) are arranged column-wise in the matrix $\mathbf{X} \in \mathbb{R}^{d\times m}$ (and $\mathbf{Y} \in \mathbb{C}^{1\times m}$), the random feature regression problem becomes:

$$\min_{\mathbf{c}\in\mathbb{C}^N} \|\mathbf{Y} - \mathbf{c}^T\phi(\mathbf{W}^T\mathbf{X} + \mathbf{b})\|_2^2 + \mathcal{R}(\mathbf{c}), \tag{1.1}$$

with some penalty function $\mathcal{R} : \mathbb{C}^N \to \mathbb{R}$. Trained coefficient vector $\mathbf{c}$ is obtained as a solution to Equation 1.1. Properties with respect to approximation power, number of trainable features, etc. for these classes of methods can be analyzed and quantified. For example, choosing the ridge penalty with $\mathcal{R}(\mathbf{c}) = \lambda_m \|\mathbf{c}\|_2^2$ leads to the random feature ridge regression (RFRR) problem studied in [49, 89, 99, 111, 120] or when the ridge parameter $\lambda_m \to 0^+$, the RFRR becomes the min-norm interpolation problem (also referred to as ridge-less regression) [8, 10, 11, 67, 90, 99, 100, 142]. A detailed analysis of both the kernel ridge regression and the RFRR problems in terms of the dimensional parameters d, m, and N are provided in [36,99]. For example, in [120], it was shown that using $N = \mathcal{O}(m^{\frac{1}{2}} \log m)$ number of random features is sufficient to achieve a test error of $\mathcal{O}(m^{-\frac{1}{2}})$ for a function f in an Reproducing Kernel Hilbert Space (see Definition 2.2.5) when training the output layer of the RFRR problem. In general, to achieve a risk bound that scales like $\mathcal{O}\left(N^{-1} + m^{-\frac{1}{2}}\right)$ using the RFRR problem over the RKHS, properties of the spectrum of the kernel operator must be known [7,49]. In practice, the technical assumptions on the spectrum and its decay may be hard to verify for a given problem or dataset. In general, most analyses indicate that a large N is needed for low-risk.

Additionally, it has been observed that the global minimizer of the risk using the RFRR problem as a function of the ratio $\frac{N}{m}$ is achieved for values $\frac{N}{m} \gg 1$ [99, 100]. That is, the lower risk solutions occur in the very overparameterized limit. Since a large N can limit their usefulness in many scientific problems, an alternative approach is to use sparsity-promoting penalties (or algorithms) to obtain sparse or low-complexity models in the overparameterized setting. For example, in [66] an ℓ_1 basis pursuit denoising problem (see Definition 2.1.10) was used to train RFM from limited (and noisy) measurements. It was shown that when the "true" values of the final weight layer $\mathbf{c}$ are compressible, the sparsity level (number of nonzero features) s can be much smaller than N to achieve approximation bounds similar to bounds derived in [36,66]. Similar RFM-based algorithms that have been successful in approximating functions from noisy and limited data can be found in [125, 151]. Similar to the RFRR problem, it was shown that when the sparsity level $s = N$ and the number of measurements $m = \mathcal{O}(N \log(N))$, the risk is bounded by $\mathcal{O}\left(N^{-1} + m^{-\frac{1}{2}}\right)$ [36]. A related algorithm based on LASSO in [153] was used to iteratively add a sparse number of random features to the trained RFM. Non-convex penalties to promote sparsity can be used such as the ℓ_0 regularization. In [151], an iterative pruning approach (related to an ℓ_0 penalized problem as defined in Eq. (2.1)) is shown to perform well, in particular, for high-dimensional approximation problems with the inherent low-order structure. Their proposed algorithm connects sparsity-promoting methods for RFM to the pruning approaches for reducing the model complexity of overparameterized neural networks. Pruning algorithms focus on obtaining small subnetworks with similar accuracy to the full neural network [54,158]. Sparsity in RFM is an actively evolving area of research owing to its wide applicability in data-scarce problems and is discussed further in Chapters 2, 3 and 4.

1.2 System Identification and Epidemic Prediction from Partial Data

System identification involves the use of various methods for using data to build mathematical models of dynamical systems. Usually, dynamical systems are learned from measured data by determining a mathematical relation between the input and output data through a surrogate model. It is also a popular application of machine learning or artificial intelligence-based methods where the model is trained to learn the trajectory of a dynamical system. Given a $d-$dimensional time dependent data $\mathbf{x}(t) \in \mathbb{R}^d$, the goal is to find a function f such that $\frac{d\mathbf{x}(t)}{dt} \approx f(\mathbf{x}(t))$ i.e., approximating the function $f(\mathbf{x}(t))$ is essential for successful system identification. A common approach to learning the trajectory of a dynamical system is either by learning the function f or directly learning the solutions $\mathbf{x}(t)$ itself. Some well-known approaches for learning the solutions directly are given in [14,27,29,57]. There are several approaches in the literature on how to learn the function f successfully. One class of method assumes the function $f(\mathbf{x}(t))$ to have sparse representations with respect to a prescribed dictionary consisting of polynomials and/or trigonometric functions. The active terms are then identified using sparse regression [24,25,28,62,75,97,121,125,126,141]. For example, on recovery of chaotic systems from highly corrupted data, [141] gave the conditions under which a chaotic dynamical system can be recovered exactly from data. The main idea used to recover the system in [141] was by representing the governing equations in the space of known functions (polynomials) to construct a dictionary matrix and solve for the coefficients to recover the system. Similar works have also been done in [28] and [126]. Another approach based on neural networks is modeling partial or ordinary differential equations using either full data or partial observations with some side information to learn the system [3,6,43,59,63,85,95,96,104,108,112,112,129,136,148,149]. A significant feature of NNs that makes it unique is the presence of a nonlinear activation function.

The activation functions that are commonly used for most NNs are Rectified Linear Units (ReLU) or variations of ReLU, tangent hyperbolic (tanh), or sigmoid function. However, NNs can often be highly unstable and prone to overfitting as demonstrated in [61] which can lead them to under-perform in comparison to sparse regularization techniques. At the same time, their effectiveness is also dependent on the availability of training data topped with a disadvantage in terms of lack of theoretical understanding. Thus, their applications to data-scarce regimes such as in the fields of public health or medicine can be extremely limited. In terms of epidemic prediction, the COVID-19 pandemic was a wake-up call for the machine learning community to come up with algorithms that would give quick predictions with minimal data so that public health can be managed and appropriate measures can be taken to control disease spread.

Surveillance data, such as daily cases, are the prime source of information about emerging infectious disease epidemics. Predicting the trajectories of epidemics from surveillance data is currently one of the most active research areas, partially due to the COVID-19 pandemic. Compartmental models, which stratify the population into compartments according to health status, are the most popular tools for this task. They have been used to study many infectious disease epidemics including plague [72], Ebola [4, 71], measles [32], HIV [19], influenza [48], and COVID-19 [42, 159]. These models utilize dynamical systems to describe the dynamics of their compartments. Traditional methods in practice make use of an underlying model to fit the measured data and subsequently find the corresponding parameters in the system [13, 31, 143, 159]. The assumed underlying model is generally a newly proposed model describing the disease dynamics. However, epidemiological data is often only partially observed where available observations are either daily active cases or cumulative cases. The incomplete nature of data makes it hard to learn the parameters as missing data corresponding to other variables can lead to incorrect learning of parameters. Inspired by the properties laid out by various delay embedding theorems, we incorporate data corresponding to the unavailable variables using Takens' Delay embedding theorem [138]. There have been some notable works in using time delay for epidemic predictions such as using delay differential equations in the newly proposed underlying models [5, 44, 65]. However, these methods still require one to know the form of the underlying differential equations for learning, which is often challenging to model. Other works of delay embedding include developing eigensystem realization algorithm in system identification to learn the eigensystem [74], extracting qualitative dynamics from data [22], and studying nonlinear dynamics in the Koopman operator framework [26, 34, 76, 87]. Delay embedding can be combined with neural networks to identify parameters and make predictions [91, 144]. Thus a potential area of exploration would be to use delay embedding for predicting a particular variable of a dynamical system from incomplete input data. This idea has been developed and elaborated further in Section 2.4 and Chapter 4.

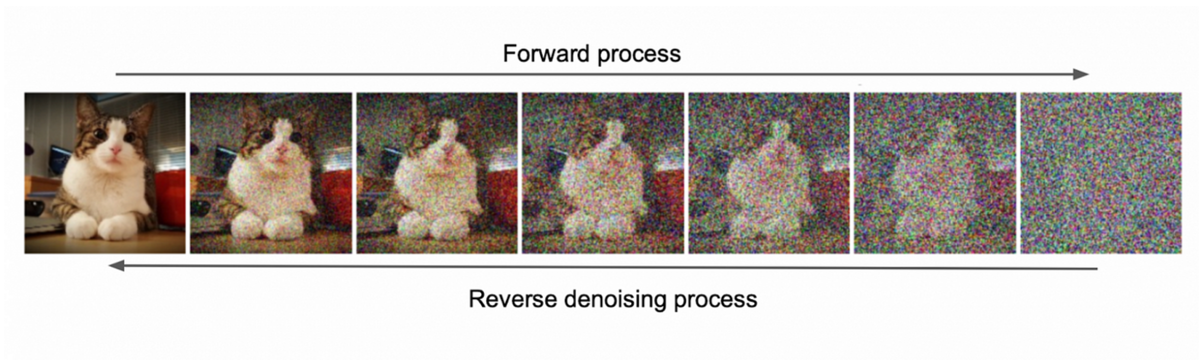


Figure 1.2: Illustration of a diffusion model training process. The forward process is a fixed Markov chain while the reverse process is generally learned using a parameterized model. Source: https://medium.com/@vasanth.ambrose/diffusion-a-shallow-dive-into-image-generation-models-43034a267cc1

1.3 Diffusion Models

Generative modeling has been successfully used to generate a wide variety of data. Some well-known models are Generative adversarial networks (GANs), flow-based models, autoregressive models, and Variational Autoencoders (VAEs) [46, 60, 79, 101, 152]. Diffusion models [70, 152] are a class of parameterized models trained using variational inference to generate samples matching the data from input distribution after a finite number of timesteps. The model learns to reverse a fixed Markov chain which adds noise to the input data until it is destroyed as depicted in Figure 1.2. The reverse Markov chain is often learned using a NN. Typically, U-Net [119] (or its variations) architecture is used for training diffusion models as it not only preserves the input-output dimensions, but the architecture of U-Net helps the model to extract and learn features through the downsampling convolutional layers. Once trained, new data is generated from noise by applying the trained model recursively on reverse timesteps. Much has been studied in terms of improving the performance of diffusion models by introducing different loss functions [70, 156], formulating the model as stochastic differential equations (SDEs) [133–135], exploring different sampling techniques post-training [94], etc. Since diffusion models generally make use of random noise added over multiple timesteps and complex parameterized models for training, the theoretical interpretability of these models is very limited in the literature. Existing theory focuses on the formulation of diffusion models using various mathematical concepts such as stochastic differential equations [133–135], ordinary differential equations [94], probabilistic models [70, 156], etc. We elaborate more in Section 2.5 which gives the background formulation of diffusion models. While there are some theoretical results involving error bounds of diffusion models [12, 35], they are mostly based on the assumption that the parameterized model being used for training converges. However, for complex networks such as a U-Net, quantifying the error bound is challenging. On the other hand, interpretable models such as RFMs are yet to be explored for training diffusion models. Motivated to build an interpretable method for diffusion models we propose a time-dependent random feature model for training of diffusion models which is further discussed in Chapter 5.

1.4 Objectives and Contributions

The main objective of this research is to develop RFM-based algorithms for data-scare problems and explore the idea of RFM in learning distributions using the training process of diffusion models. We approach this goal by first understanding and developing a sparsity-based RFM algorithm for function approximation. Our second objective is to work with incomplete and scarce data. In particular, we look at incomplete data coming from epidemiology that can be modeled using ODEs. Finally, we aim to develop an interpretable RFM architecture based on diffusion models. While RFM has been an area of much interest lately, applications of RFM to epidemiology and diffusion random features have not appeared in the literature. We list the contributions below in detail.

- We develop an algorithm based on sparsity and RFM which is used to train sparse additive random feature models. We include a hard thresholding step in the proposed algorithm which is a pruning step to obtain a subset of features that could lead to better generalization results with a smaller RFM architecture, (see also [54, 158]). Along with obtaining a generalization bound for our approach based on the proofs from [36, 66], we show using tests on both synthetic and real-world datasets that our proposed method performs comparably or better than other related algorithms. The sparsity priors help to obtain important variable dependencies from the data.

- Data availability is a challenge in building algorithms for epidemic predictions. We illustrate that it is feasible to forecast future dynamics of infectious diseases using delay embedding. Specifically, we approximate the rate of change in the observed variable (daily active or cumulative cases) as a sparse random feature expansion of its time-delayed mapping. Here, a sparsity constraint in the random feature model helps balance between the richness representation of the function space and the limited amount of available data. Then, we use the learned function to make a prediction over a future time window. We show that our method successfully outperforms benchmark methods based on simulated and real datasets of various epidemics including COVID-19 in Canada, Ebola in Guinea, Zika in Giradot, and Flu in China. Further, our comparisons with other function approximation techniques reveal that our RFM-based dictionary can consistently perform comparably as the best-performing dictionary regardless of the dataset.

- Inspired by the process of building diffusion models, we proposed a model architecture for generative RFM. It is one of the first works introduced to combine the idea of random features with generative models. The new architecture acts as a bridge between the theoretical and generative aspects of the diffusion model by providing approximation bounds of samples generated by diffusion model-based training algorithms. We show that for each fixed timestep, our Diffusion Random Feature Model (DRFM) can be reduced to a (shallow) random feature model preserving all the relevant theoretical properties.

1.5 Thesis Overview

The outline of the remaining chapters is given below.

Chapter 2. We recall some important definitions and concepts from compressive sensing, dynamical systems, and probability theory which are key to understanding the thesis contents.

Chapter 3. In this chapter we propose an algorithm for approximating high dimensional sparse additive functions called the Hard-Ridge Random Feature Expansion method (HARFE). We use a hard-thresholding pursuit-based algorithm applied to the sparse ridge regression (SRR) problem to approximate the coefficients with respect to the random feature matrix. As side information, a random sparse connectivity pattern is added in the random feature matrix for the additive function assumption. We derive theoretical results showing that HARFE is guaranteed to converge with a given error bound that depends on the noise and the parameters of the sparse ridge regression model. Along with risk bounds derivations, we also derive a risk bound on the learned model and show based on numerical results on synthetic data as well as on real datasets that HARFE obtains lower (or comparable) error than other state-of-the-art algorithms. The contents of this chapter have been taken, with modifications from the article:

Saha, E., Schaeffer, H. and Tran, G., 2023. HARFE: Hard-ridge Random Feature Expansion. *Sampling Theory, Signal Processing, and Data Analysis,* 21(2), pp.1-24. https://link.springer.com/content/pdf/10.1007/s43670-023-00063-9.pdf.

Chapter 4. Compartmental models, one of the most popular tools for modeling and predicting infectious disease epidemics may not capture the true dynamics of the epidemic due to the complexity of the disease transmission and human interactions. We propose an algorithm named Sparsity and Delay Embedding-based Forecasting (SPADE4) for predicting epidemics. SPADE4 predicts the future trajectory of an observed variable without the knowledge of the other variables or the underlying system. The key ideas are to use an RFM with sparse regression to handle the data scarcity issue and employ Takens' delay embedding theorem to capture the nature of the underlying system from the observed variable. We show that our approach outperforms compartmental models when applied to both simulated and real data. The contents of this chapter are taken, with modification from the article:

Saha, E., Ho, L. S. T. & Tran, G. SPADE4: Sparsity and Delay Embedding for Forecasting Epidemics, *Bull Math Biol.* **85**, 71 (2023). https://link.springer.com/article/10.1007/s11538-023-01174-z

Chapter 5. Most diffusion models are computationally expensive and difficult to interpret with a lack of theoretical justification. Random feature models (RFMs) on the other hand have gained popularity due to their interpretability but their application to complex machine learning tasks remains limited. In this chapter, we present a diffusion model-inspired random feature model that is interpretable and gives comparable numerical results to a fully connected neural network having the same number of trainable parameters. We validate our findings by generating samples on the Fashion-MNIST dataset and instrumental audio data. The contents of this chapter are taken, with modification from the article (under review):

Saha, E. & Tran, G. Diffusion Random Feature Model. arxiv preprint. 2023. https://arxiv.org/abs/2310.04417

Chapter 6. We conclude the thesis with concluding remarks and a discussion on possible future work and open problems corresponding to the research presented.

Chapter 2

Background

In this chapter, we give the definitions, results, and derivation essential to the research presented in the thesis. This chapter is divided into three main sections. In Section 2.1 we provide definitions and results from compressive sensing along with some well-known sparse recovery algorithms followed by a discussion on the formulation of random feature models and their theoretical results. Some elementary discussion on ODEs and epidemic models are given in Section 2.4. In the last section of this chapter, Section 2.5, we review the basic background of probability and the formulation of diffusion models.

2.1 Basic Algorithms in Compressive Sensing

A common problem in linear algebra is to solve the problem $\mathbf{y} = \mathbf{A}\mathbf{x}$ where the observed data $\mathbf{y} \in \mathbb{C}^m$ is connected to an unknown vector $\mathbf{x} \in \mathbb{C}^N$ via a measurement matrix $\mathbf{A} \in \mathbb{C}^{m\times N}$. This can be found in many practical problems such as in signal and image processing where one has to extract quantities of interest from measured data [2, 20, 40, 106]. Often the matrix $\mathbf{A} \in \mathbb{C}^{m\times N}$ models the linear information measurement process. For successful recovery of the vector $\mathbf{x} \in \mathbb{C}^N$, the amount of measured data m has to be at least as large as the number of components of $\mathbf{x}$ [53]. Practically, this is often difficult to incorporate in most current technology, such as medical imaging, radar, mobile communication, etc. This leads to an underdetermined system which leads to the case of infinitely many solutions (assuming at least one solution exists). Thus, without any additional information, it is hard to recover $\mathbf{x}$ when $m < N$. The key idea of compressive sensing lies in recovering the vector $\mathbf{x}$ when $m < N$ with the assumption that the vector $\mathbf{x}$ is sparse i.e., most of its components are zero. Below we recall some relevant definitions, results, and algorithms for sparse recovery of vectors. For a detailed understanding, we refer the reader to [53].

For a matrix $\mathbf{A} \in \mathbb{C}^{m\times N}$, a sparse representation of $\mathbf{y}$ can be obtained by solving

$$\underset{\mathbf{z}\in\mathbb{C}^N}{\text{minimize}} \ \|\mathbf{z}\|_0 \text{ subject to } \mathbf{A}\mathbf{z} = \mathbf{y}. \tag{2.1}$$

The above problem is known as the ℓ_0-minimization problem whose main idea is to find the sparsest vector $\mathbf{z}$ satisfying $\mathbf{y} = \mathbf{Az}$. Note that in general, the recovered vector $\mathbf{z}$ may not be unique. We first recall a definition and a result stated as Theorem 2.13 in [53] which gives the conditions under which the recovered vector $\mathbf{z}$ will be unique.

Definition 2.1.1 (s-Sparse Vectors). *A vector $\mathbf{x} \in \mathbb{C}^N$ is said to be $s-$sparse it it has atmost s nonzero components.*

Theorem 2.1.2. *Given $\mathbf{A} \in \mathbb{C}^{m\times N}$, the following properties are equivalent.*

1. *Every s-sparse vector $\mathbf{x} \in \mathbb{C}^N$ is the unique s-sparse solution of $\mathbf{Az} = \mathbf{Ax}$, that is, if $\mathbf{Ax} = \mathbf{Az}$ and both $\mathbf{x}$ and $\mathbf{z}$ are s-sparse then $\mathbf{x} = \mathbf{z}$.*
2. *The null space $\ker(\mathbf{A})$ does not contain an $2s$-sparse vector other than the zero vector, that is, $\ker(\mathbf{A}) \cap \{\mathbf{z} \in \mathbb{C}^N : \|\mathbf{z}\|_0 \leq 2s\} = \{\mathbf{0}\}$.*
3. *For every subset $S \subseteq [N]$ with $\operatorname{card}(S) \leq 2s$, the submatrix $\mathbf{A}_S$ (consisting of the columns indexed by S) is injective as a map from $\mathbb{C}^S$ to $\mathbb{C}^m$.*
4. *Every set of $2s$ columns of $\mathbf{A}$ is linearly independent.*

The ℓ_0-minimization problem is a non-convex problem. Additionally, the problem by itself is also NP-hard in general. Thus, a common and alternative approach to the above problem is to use the ℓ_1 norm, which is a convex relaxation of the ℓ_0-problem. The ℓ_1-minimization problem consists of solving

$$\underset{\mathbf{z}\in\mathbb{C}^N}{\text{minimize}} \ \|\mathbf{z}\|_1 \text{ subject to } \mathbf{Az} = \mathbf{y}.$$

Next, we discuss some elementary definitions followed by a few well-known formulations of the ℓ_1- optimization problem.

Definition 2.1.3 (Restricted Isometry Constant). *The s^{th} restricted isometry constant $\delta_s = \delta_s(\mathbf{A})$ of a matrix $\mathbf{A} \in \mathbb{C}^{m\times N}$ is the smallest $\delta \geq 0$ such that*

$$(1-\delta)\|\mathbf{x}\|_2^2 \leq \|\mathbf{Ax}\|_2^2 \leq (1+\delta)\|\mathbf{x}\|_2^2,$$

for all $s-$sparse vectors $\mathbf{x} \in \mathbb{C}^N$. Equivalently, it is also defined as,

$$\delta_s = \max_{S\subseteq[N],\operatorname{card}(S)\leq s} \|\mathbf{A}_S^\star \mathbf{A}_S - \mathbf{I}_{\operatorname{card}(S)}\|_{2\to 2}.$$

Definition 2.1.4 (Null Space Property). *A matrix $\mathbf{A} \in \mathbb{C}^{m\times N}$ is said to satisfy null space property relative to a set $S \subseteq [N]$ if*

$$\|\mathbf{v}_S\|_1 < \|\mathbf{v}_{\overline{S}}\|_1 \textit{ for all } \mathbf{v} \in \ker(\mathbf{A}) \setminus \{\mathbf{0}\},$$

where $\overline{S} = [N] \setminus S$.

Definition 2.1.5 (Robust Null Space Property). *The matrix* $\mathbf{A} \in \mathbb{C}^{m \times N}$ *is said to satisfy the robust null space property (with respect to* $\|.\|$*) with constants* $0 < \rho < 1$ *and* $\tau > 0$ *relative to a set* $S \subseteq [N]$ *if*

$$\|\mathbf{v}_S\|_1 \leq \rho \|\mathbf{v}_{\overline{S}}\|_1 + \tau \|\mathbf{A}\mathbf{v}\| \text{ for all } \mathbf{v} \in \mathbb{C}^N,$$

where $\mathbf{v}_S$ *denoted the restriction of the indices of* $\mathbf{v}$ *to the set* S *and* $\overline{S} = [N] \setminus S$.

Definition 2.1.6 (ℓ_q- Robust Null Space Property). *Given* $q \geq 1$*, the matrix* $\mathbf{A} \in \mathbb{C}^{m \times N}$ *is said to satisfy the* ℓ_q*-robust null space property of order* s *(with respect to* $\|.\|$*) with constants* $0 < \rho < 1$ *and* $\tau > 0$ *relative to a set* $S \subseteq [N]$ *with* $\operatorname{card}(S) \leq s$ *if*

$$\|\mathbf{v}_S\|_q \leq \frac{\rho}{s^{1-1/q}} \|\mathbf{v}_{\overline{S}}\|_1 + \tau \|\mathbf{A}\mathbf{v}\| \text{ for all } \mathbf{v} \in \mathbb{C}^N,$$

where $\mathbf{v}_S$ *denoted the restriction of the indices of* $\mathbf{v}$ *to the set* S *and* $\overline{S} = [N] \setminus S$.

Definition 2.1.7 (Best s-term Approximation). *The* ℓ_1 *distance to the best* s*-term approximation of* $\mathbf{x}$ *is defined by*

$$\kappa_{1,s}(\mathbf{x}) = \inf \left\{ \|\mathbf{z} - \mathbf{x}\|_1 : \mathbf{z} \in \mathbb{C}^N, \, \mathbf{z} \text{ is } s-\text{sparse} \right\}.$$

The value $\kappa_{1,s}(\mathbf{x})$ provides a measure for the compressibility of the vector $\mathbf{x}$ with respect to the ℓ_1. It can obtained by considering a s-sparse vector $\mathbf{z} \in \mathbb{C}^N$, whose nonzero entries are the s largest absolute entries of $\mathbf{x}$ [53]. Note that in particular $\kappa_{1,s}(\mathbf{x}) = 0$ if $\mathbf{x}$ is $s-$sparse and $\kappa_{1,s}(\mathbf{x}) \leq \|\mathbf{x}\|_1$ always [66].

Definition 2.1.8 (Basis Pursuit). *Given a measurement matrix* $\mathbf{A} \in \mathbb{C}^{m \times N}$ *and a measurement vector* $\mathbf{y} \in \mathbb{C}^m$*, Basis Pursuit consists in finding the minimizer of the following problem*

$$\min_{\mathbf{z} \in \mathbb{C}^N} \|\mathbf{z}\|_1 \text{ subject to } \mathbf{A}\mathbf{z} = \mathbf{y}. \tag{2.2}$$

In general the measurement vector $\mathbf{y} \in \mathbb{C}^m$ is noisy, $\mathbf{y} = \mathbf{A}\mathbf{x} + \mathbf{e}$ for some measurement error $\mathbf{e} \in \mathbb{C}^m$ such that $\|\mathbf{e}\|_2 \leq \eta$ for some $\eta \geq 0$. A general ℓ_1-minimization problem, which takes measurement noise into account, is given by the quadratically-constrained basis pursuit.

Definition 2.1.9 (Quadratically-Constrained Basis Pursuit (QCBP)). *Given a measurement matrix* $\mathbf{A} \in \mathbb{C}^{m \times N}$ *and a measurement vector* $\mathbf{y} \in \mathbb{C}^m$*, QCBP consists in finding the minimizer of the following problem:*

$$\min_{\mathbf{z} \in \mathbb{C}^N} \|\mathbf{z}\|_1 \text{ subject to } \|\mathbf{A}\mathbf{z} - \mathbf{y}\|_2 \leq \eta. \tag{2.3}$$

The solution to the above problem is also linked to the *basis pursuit denoising* (BPDN) problem and the *least absolute shrinkage and selection operator* (LASSO) defined below.

Definition 2.1.10 (Basis Pursuit Denoising Problem (BPDN))**.** *Given a measurement matrix* $\mathbf{A} \in \mathbb{C}^{m\times N}$ *and a measurement vector* $\mathbf{y} \in \mathbb{C}^m$*, BPDN consists in finding the minimizer of the following problem*

$$\min_{\mathbf{z}\in\mathbb{C}^N} \lambda\|\mathbf{z}\|_1 + \|\mathbf{A}\mathbf{z}-\mathbf{y}\|_2^2, \tag{2.4}$$

for some parameter $\lambda \geq 0$*.*

Definition 2.1.11 (Least Absolute Shrinkage and Selection Operator (LASSO))**.** *Given a measurement matrix* $\mathbf{A} \in \mathbb{C}^{m\times N}$ *and a measurement vector* $\mathbf{y} \in \mathbb{C}^m$*, LASSO consists in finding the minimizer of the following problem*

$$\min_{\mathbf{z}\in\mathbb{C}^N} \|\mathbf{A}\mathbf{z}-\mathbf{y}\|_2 \text{ subject to } \|\mathbf{z}\|_1 \leq \tau, \tag{2.5}$$

for some parameter $\tau \geq 0$*.*

Below we state Proposition 3.2 from [53] which shows that the solutions obtained from BPDN, QCBP, and LASSO are equivalent.

Theorem 2.1.12. *The conditions below give the equivalence of solutions obtained using BPDN, QCBP, and LASSO.*

1. *If* $\mathbf{x}$ *is a minimizer of BPDN with* $\lambda > 0$*, then there exists* $\eta = \eta_{\mathbf{x}} \geq 0$ *such that* $\mathbf{x}$ *is a minimizer of the QCBP problem.*
2. *If* $\mathbf{x}$ *is a minimizer of QCBP with* $\eta \geq 0$*, then there is* $\tau = \tau_{\mathbf{x}} \geq 0$ *such that* $\mathbf{x}$ *is a minimizer of the LASSO.*
3. *If* $\mathbf{x}$ *is a minimizer of the LASSO with* $\tau > 0$*, then there is* $\lambda = \lambda_{\mathbf{x}} \geq 0$ *such that* $\mathbf{x}$ *is a minimizer of BPDN.*

Below we state Theorem 4.22 from [53] which gives the robustness of the QCBP problem.

Theorem 2.1.13. *Suppose that a matrix* $\mathbf{A} \in \mathbb{C}^{m\times N}$ *satisfies the* ℓ_2*-robust null space property of order* s *with constants* $0 < \rho < 1$, $\tau > 0$ *and* $p > 1$*. Then, for any* $\mathbf{x} \in \mathbb{C}^N$*, a solution* $\mathbf{x}^\sharp$ *of QCBP with* $\mathbf{y} = \mathbf{A}\mathbf{x} + \mathbf{e}$ *and* $\|\mathbf{e}\|_2 \leq \eta$ *approximates the vector* $\mathbf{x}$ *with* ℓ_p*-error*

$$\|\mathbf{x}-\mathbf{x}^\sharp\|_p \leq \frac{C}{s^{1-1/p}}\kappa_{1,s}(\mathbf{x}) + Ds^{1/p-1/2}\eta, \tag{2.6}$$

for some constants $C, D > 0$ *depending only on* ρ *and* τ*.*

We also discuss some well-known greedy algorithms. These are algorithms for non-convex optimization problems. The approach is based on solving problems by making a locally optimal choice at each stage with the expectation of finding the global optimum. They may not be able to find the best solution to many problems as they do not explore the solution space too much. Greedy methods, as the name suggests have the advantage of faster convergence to a local solution and are computationally cheaper. We use the notation $H_s(\mathbf{z})$ for the hard thresholding operator of order s which keeps the s largest absolute entries of a vector. It gives the best s-term approximation to $\mathbf{z} \in \mathbb{C}^N$. The notation $L_s(\mathbf{z})$ gives the index set of s largest entries of $\mathbf{z} \in \mathbb{C}^N$ in modulus. Then, $H_s(\mathbf{z}) = \mathbf{z}_{L_s(\mathbf{z})}$.

Algorithm 2.1.14 (Orthogonal Matching Pursuit (OMP) Algorithm)**.** *Given measurement matrix $\mathbf{A} \in \mathbb{C}^{m\times N}$, measurement vector $\mathbf{y} \in \mathbb{C}^m$, initialize $S^0 = \varphi$, $\mathbf{x}^0 = \mathbf{0}$. Repeat the following steps until a stopping criterion is met at $n = \bar{n}$,*

$$S^{n+1} = S^n \cup \left\{ j_{n+1} := \underset{j\in[N]}{\operatorname{argmax}} \{ |(\mathbf{A}^\star(\mathbf{y} - \mathbf{A}\mathbf{x}^n))_j| \} \right\}, \tag{2.7}$$

$$\mathbf{x}^{n+1} = \underset{\mathbf{z}\in\mathbb{C}^{\mathbf{N}}}{\operatorname{argmin}} \{ \|\mathbf{y} - \mathbf{A}\mathbf{z}\|_2, \operatorname{supp}(\mathbf{z}) \subseteq S^{n+1} \}. \tag{2.8}$$

The final output is the $\bar{n}$-sparse vector $\mathbf{x}^\sharp = \mathbf{x}^{\bar{n}}$.

At each iteration, the OMP algorithm adds one index to a target support S^n followed by updating a target vector $\mathbf{x}^n$ that best fits the measurements such that it is supported on the support S^n. Below we state a result that shows that the OMP algorithm can recover sparse vectors $\mathbf{x}$. The result implies stability and robustness of the OMP algorithm and is given as Proposition 6.24 in [53].

Theorem 2.1.15. *Suppose $\mathbf{A} \in \mathbb{C}^{m\times N}$, let $\mathbf{y} = \mathbf{A}\mathbf{x} + \mathbf{e}$ for some $s-$ sparse $\mathbf{x} \in \mathbb{C}^N$ with $S = \operatorname{supp}(\mathbf{x})$ and some $\mathbf{e} \in \mathbb{C}^m$. Let $(\mathbf{x}^n)$ denote the sequence defined by the OMP algorithm started at an index set S^0. With $s^0 = \operatorname{card}(S^0)$ and $s' = \operatorname{card}(S \setminus S^0)$, if $\delta_{s+s^0+12s'} < \frac{1}{6}$, then there is a constant $C > 0$ depending only on $\delta_{s+s^0+12s'}$ such that*

$$\|\mathbf{y} - \mathbf{A}\mathbf{x}^{\bar{n}}\|_2 \leq C\|\mathbf{e}\|_2, \quad \bar{n} := 12s'. \tag{2.9}$$

Another well-known class of greedy algorithms includes thresholding methods. We discuss two well-known thresholding algorithms namely iterative hard thresholding and hard thresholding pursuit. The iterative hard thresholding algorithm is an iterative algorithm used to solve the underdetermined system $\mathbf{y} = \mathbf{A}\mathbf{x}$, knowing that the solution is s-sparse. The steps are described below.

Algorithm 2.1.16 (Iterative Hard Thresholding (IHT) Algorithm)**.** *Given a measurement matrix $\mathbf{A} \in \mathbb{C}^{m\times N}$ and a measurement vector $\mathbf{y} \in \mathbb{C}^m$, the algorithm starts with an initial s-sparse vector $\mathbf{x}^0 \in \mathbb{C}^N$, typically $\mathbf{x}^0 = \mathbf{0}$, and produces a sequence $(\mathbf{x}^n)$ defined inductively by*

$$\mathbf{x}^{n+1} = H_s(\mathbf{x}^n + \mathbf{A}^*(\mathbf{y} - \mathbf{A}\mathbf{x}^n)),$$

where the hard thresholding operator H_s keeps the s largest modulus components of a vector, such that $H_s(\mathbf{z})$ is a (not necessarily unique) best s-term approximation $\mathbf{z} \in \mathbb{C}^N$. The final output is obtained when a stopping criterion is met at $n = \bar{n}$ and is given by the s-sparse vector $\mathbf{x}^\sharp = \mathbf{x}^{\bar{n}}$.

Algorithm 2.1.17 (Hard Thresholding Pursuit (HTP) Algorithm)**.** *Suppose $\mathbf{A} \in \mathbb{C}^{m \times N}$ is a measurement matrix with a measurement vector $\mathbf{y} \in \mathbb{C}^m$, the algorithm starts with an initial s-sparse vector $\mathbf{x}^0 \in \mathbb{C}^N$, typically $\mathbf{x}^0 = \mathbf{0}$, and produces a sequence $(\mathbf{x}^n)$ by repeating the following steps until a stopping criterion is met at $n = \bar{n}$*

$$\begin{aligned} S^{n+1} &= L_s(\mathbf{x}^n + \mathbf{A}^\star(\mathbf{y} - \mathbf{A}\mathbf{x}^n)), \\ \mathbf{x}^{n+1} &= \underset{\mathbf{z} \in \mathbb{C}^N}{\operatorname{argmin}} \{\|\mathbf{y} - \mathbf{A}\mathbf{z}\|_2, \operatorname{supp}(\mathbf{z}) \subseteq S^{n+1}\}. \end{aligned}$$

The final output is the s-sparse vector $\mathbf{x}^\sharp = \mathbf{x}^{\bar{n}}$.

The result below gives one of the well-known results for sparse recovery of vectors from given measurements using IHT and HTP stated as Theorem 6.21 from [53].

Theorem 2.1.18. *Suppose that the $(6s)^{th}$ order restricted isometry constant of $\mathbf{A} \in \mathbb{C}^{m \times N}$ satisfies $\delta_{6s} < \frac{1}{\sqrt{3}}$, then for any $\mathbf{x} \in \mathbb{C}^N$ and $\mathbf{e} \in \mathbb{C}^m$, the sequence $\mathbf{x}^{(n)}$ defined by the IHT or HTP with $y = \mathbf{A}\mathbf{x} + \mathbf{e}$, $\mathbf{x}^0 = \mathbf{0}$, using $2s$ instead of s in the algorithm, satisfies*

$$\|\mathbf{x}^n - \mathbf{x}\|_2 \leq 2\beta^n \|\mathbf{x}\|_2 + \frac{D_1}{\sqrt{s}} \kappa_{1,s}(\mathbf{x}) + D_2 \|\mathbf{e}\|_2, \tag{2.10}$$

for all $n \geq 0$ where the constants $\beta \in (0,1)$, $D_1, D_2 > 0$ depend only on δ_{6s}. In particular, if the sequence $\mathbf{x}^{(n)}$ clusters around some $\mathbf{x}^\sharp \in \mathbb{C}^N$, then

$$\|\mathbf{x} - \mathbf{x}^\sharp\|_2 \leq \frac{D_1}{\sqrt{s}} \kappa_{1,s}(\mathbf{x}) + D_2 \|\mathbf{e}\|_2. \tag{2.11}$$

We would like to mention that although there are other well-known algorithms in literature for sparse recovery of vectors from measured data, we discuss the methods that are closely connected to the research undertaken in the thesis. The main focus of one of the research projects is to develop the HTP algorithm for ridge regression problems with a random feature matrix $\mathbf{A}$ for function approximation. This idea is further developed and detailed in Chapter 3.

2.2 Kernel Methods

In this section, we discuss function approximation using kernels, which serves as one of the building blocks for random feature methods. In this section, we give a brief insight into kernel methods and their advantages. For a detailed reading, we refer the readers to [73, 128]. Given a training set $S = \{(\mathbf{x}^{(k)}, y^{(k)})\}_{k=1}^{m}$, where $\mathbf{x}^{(k)} \in \mathbb{R}^d$ and $y^{(k)} \in \mathbb{R}$, a linear regression model consists of finding the best interpolation real-valued function f of the form

$$f(\mathbf{x}) = \langle \boldsymbol{\omega}, \mathbf{x} \rangle = \mathbf{x}^T \boldsymbol{\omega} = \sum_{i=1}^{d} w_i x_i, \tag{2.12}$$

for each $\mathbf{x} = [x_1, \cdots, x_d]^T \in S$. This is one of the simplest forms of data fitting where the linear function f of the features $\mathbf{x}$ matches the corresponding label y such that

$$|y - f(\mathbf{x})| = |y - \langle \boldsymbol{\omega}, \mathbf{x} \rangle| \approx 0.$$

This task is also known as linear regression. Geometrically, f is a hyperplane fitted through the given d-dimensional points. The illustration in Figure 2.1 shows an example for $d = 1$. Ideally, we are interested in finding a function for which the training errors for all the

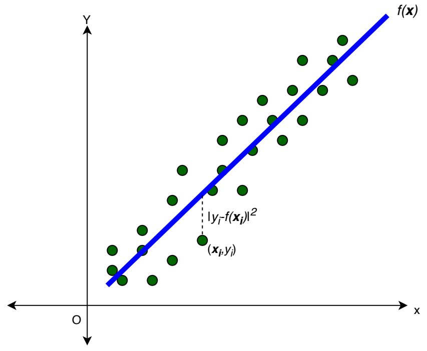


Figure 2.1: Fitting a function $f(\mathbf{x}) = \boldsymbol{\omega}^T \mathbf{x}$ to a set of data points using linear regression

samples in S are small, i.e., $|y - f(\mathbf{x})|$ is small for all $(\mathbf{x}^{(k)}, y^k) \in S$. The most commonly chosen measure is the sum of the squares of the individual errors between the training data and a particular function. This is defined by another real-valued function $\mathcal{L}$ given by

$$\mathcal{L}(f, S) = \mathcal{L}(\boldsymbol{\omega}, S) = \sum_{k=1}^{m} (y^{(k)} - f(\mathbf{x}^{(k)}))^2 = \|\mathbf{y} - \mathbf{X}\boldsymbol{\omega}\|_2^2, \tag{2.13}$$

where $\mathbf{y} = [y^{(1)}, \cdots, y^{(m)}]^T \in \mathbb{R}^m$, $\mathbf{X} = [\mathbf{x}^{(1)}, \cdots, \mathbf{x}^{(m)}]^T \in \mathbb{R}^{m \times d}$, and $\boldsymbol{\omega} \in \mathbb{R}^d$. As seen in Figure 2.1, this corresponds to minimizing the distance between the given points and the hyperplane being fitted. To minimize the value of $\mathcal{L}$ over $\boldsymbol{\omega}$, we set $\dfrac{\partial \mathcal{L}}{\partial \boldsymbol{\omega}}$ equal to zero i.e.,

$$\frac{\partial \mathcal{L}(\boldsymbol{\omega}, S)}{\partial \boldsymbol{\omega}} = -2\mathbf{X}^T \mathbf{y} + 2\mathbf{X}^T \mathbf{X} \boldsymbol{\omega} = 0. \tag{2.14}$$

Thus, we get the normal equations given by

$$\mathbf{X}^{\mathbf{T}} \mathbf{X} \boldsymbol{\omega} = \mathbf{X}^T \mathbf{y}. \tag{2.15}$$

If $(\mathbf{X}^T\mathbf{X})^{-1}$ exists, then we can directly express $\boldsymbol{\omega}$ as

$$\boldsymbol{\omega} = (\mathbf{X}^T\mathbf{X})^{-1}(\mathbf{X}^T\mathbf{y}), \tag{2.16}$$

where the obtained $\boldsymbol{\omega}$ is the unique vector that minimizes the loss.

If $(\mathbf{X}^T\mathbf{X})$ is not invertible, then Eq. (2.15) has infinitely many solutions. In order to choose an $\boldsymbol{\omega}$ out of the infinite ones, there are various options. One such approach is to use the pseudo-inverse of $(\mathbf{X}^{\mathbf{T}}\mathbf{X})$ to obtain $\boldsymbol{\omega}$. Then the solution is given by

$$\boldsymbol{\omega} = (\mathbf{X}^T\mathbf{X})^{+}(\mathbf{X}^T\mathbf{y}), \tag{2.17}$$

where we use the superscript '$+'$ to denote the pseudo-inverse of a matrix. Note that using the pseudo-inverse gives a unique solution which such that $\boldsymbol{\omega}$ has the smallest ℓ_2 norm. Another formulation for obtaining an $\boldsymbol{\omega}$ based on finding the minimum-norm solution is known as regularization, which is formulated and discussed in Equations (2.20) and (2.21).

Note that in order to find a representation of the function f as a linear combination of the inner products of input data, the expression in Eq. (2.16) can also be rewritten as

$$\boldsymbol{\omega} = \mathbf{X}^T\mathbf{X}\left((\mathbf{X}^T\mathbf{X})^{-1}\right)^2\mathbf{X}^T\mathbf{y}, \tag{2.18}$$

which allows us to write $\boldsymbol{\omega} = \sum\limits_{k=1}^{m} \alpha_k \mathbf{x}^{(k)}$ or $\boldsymbol{\omega} = \mathbf{X}^T\boldsymbol{\alpha}$, where $\boldsymbol{\alpha} = \mathbf{X}\left((\mathbf{X}^T\mathbf{X})^{-1}\right)^2\mathbf{X}^T\mathbf{y} \in \mathbb{R}^m$. Thus the target function f can be evaluated as

$$f(\mathbf{x}) = \left\langle \sum_{k=1}^{m} \alpha_k \mathbf{x}^{(k)}, \mathbf{x} \right\rangle = \sum_{k=1}^{m} \alpha_k \left\langle \mathbf{x}^{(k)}, \mathbf{x} \right\rangle. \tag{2.19}$$

The predicted output on a new data point can be computed using the learned function $f(\mathbf{x}) = \langle \boldsymbol{\omega}, \mathbf{x} \rangle$.

As mentioned above, another technique to choose an $\boldsymbol{\omega}$ when there are infinitely many solutions is by using regularization. The main idea is to restrict our choice of $\boldsymbol{\omega}$ by imposing certain conditions. For example, for the case of least squares regression, the choice parameters with small ℓ_2 norm of the weights is a well-known criterion called *ridge regression*. Then, the modified loss function becomes

$$\mathcal{L}(f, S) = \mathcal{L}(\boldsymbol{\omega}, S) = \sum_{k=1}^{m}(y^{(k)} - f(\mathbf{x}^{(k)}))^2 + \lambda \sum_{j=1}^{d} w_j^2 = \|\mathbf{y} - \mathbf{X}\boldsymbol{\omega}\|_2^2 + \lambda\|\boldsymbol{\omega}\|_2^2, \tag{2.20}$$

where λ denotes the regularization parameter and controls the strength of the 'weight decay'. Using the loss function in Eq. (2.20) and similar steps as in Equations (2.14) and (2.15), the solution to obtain $\boldsymbol{\omega}$ can be computed as

$$\boldsymbol{\omega} = (\mathbf{X}^T\mathbf{X} + \lambda\mathbf{I})^{-1}(\mathbf{X}^T\mathbf{y}). \tag{2.21}$$

Note that the term $(\mathbf{X}^T\mathbf{X} + \lambda\mathbf{I})$ is always invertible as long as $\lambda > 0$. This result can be verified from the fact that $\mathbf{X}^T\mathbf{X}$ is positive semi-definite and $\lambda\mathbf{I}$ is positive definite for $\lambda > 0$ making $\mathbf{X}^T\mathbf{X} + \lambda\mathbf{I}$ a positive definite matrix. We recall positive semi-definite and positive definite matrices below for reference.

Definition 2.2.1 (Positive Semi-Definite (PSD) and Positive Definite (PD) Matrices)**.** *A matrix $\mathbf{M}$ is called positive semi-definite if $\mathbf{x}^T\mathbf{M}\mathbf{x} \geq 0\ \forall \mathbf{x} \in \mathbb{R}^d$. If $\mathbf{x}^T\mathbf{M}\mathbf{x} > 0$ for any nonzero vector $\mathbf{x} \in \mathbb{R}^d$, then the matrix $\mathbf{M}$ is said to be positive definite.*

Regularization is a popular technique used to restrict the choice of functions when the data is noisy and thus an exact fitting of data is undesirable. Whether we choose to use regularization or not, the learned function in the above approaches is linear. However, the relation between the input data and the output is often nonlinear, and thus for a better representation we need to map the inputs to a new feature space such that it can capture nonlinear relations. We transform the features of the input data into a new feature space with dimension N in such a way that the relation between the transformed feature space and the output can be represented in a linear form. Consider an embedding map $\psi : \mathbb{R}^d \to \mathbb{R}^N$ which helps to convert nonlinear relations to linear ones. For example, consider a classification problem where one has to learn a function that separates two classes of data in $\mathbb{R}^2$. While it may not always be possible to find a hyperplane (line in $\mathbb{R}^2$) that can easily separate the two classes, mapping it to a higher dimension may make it linearly separable (see Figure 2.2). Thus, consider a transformed set $\hat{S} = \{(\psi(\mathbf{x}^{(k)}), y^{(k)})\}_{k=1}^m$ and a prediction function $\hat{f}$ of the form

$$\hat{f}(\mathbf{x}) = \langle \boldsymbol{\omega}, \psi(\mathbf{x}) \rangle. \tag{2.22}$$

Thus, assuming we have the data matrix $\mathbf{X} = [\mathbf{x}^{(1)}, \cdots, \mathbf{x}^{(m)}]^T$ and output vector $\mathbf{y} = [y^{(1)}, \cdots, y^{(m)}]^T$, the learned weights using linear regression can be computed using steps similar to Equations (2.14) and (2.15) and is given by

$$\begin{aligned} \boldsymbol{\omega} &= (\psi(\mathbf{X})^T\psi(\mathbf{X}))^{-1}\psi(\mathbf{X})^T\mathbf{y} && (2.23)\\ &= \left([\psi(\mathbf{x}^{(1)}, \cdots, \psi(\mathbf{x}^{(m)})]) \begin{bmatrix} \psi(\mathbf{x}^{(1)}) \\ \vdots \\ \psi(\mathbf{x}^{(m)}) \end{bmatrix} \right)^{-1} [\psi(\mathbf{x}^{(1)}), \cdots, \psi(\mathbf{x}^{(m)})] \begin{bmatrix} y^{(1)} \\ \vdots \\ y^{(m)} \end{bmatrix}, && (2.24) \end{aligned}$$

provided $(\psi(\mathbf{X})^T\psi(\mathbf{X}))^{-1}$ exists. If it does not exist then one may choose to use the pseudo-inverse or regularization techniques to obtain $\boldsymbol{\omega}$ as discussed before. As in Eq. (2.16), the the prediction function can be evaluated using,

$$\hat{f}(\mathbf{x}) = \sum_{k=1}^{m} \alpha_k \left\langle \psi(\mathbf{x}^{(k)}), \psi(\mathbf{x}) \right\rangle, \tag{2.25}$$

where α_k is the k^{th} component of $\boldsymbol{\alpha} = \psi(\mathbf{X})((\psi(\mathbf{X})^T\psi(\mathbf{X}))^+)^2\psi(\mathbf{X})^T\mathbf{y} \in \mathbb{R}^m$. Thus for evaluating the prediction function for every new point, one needs to compute the inner

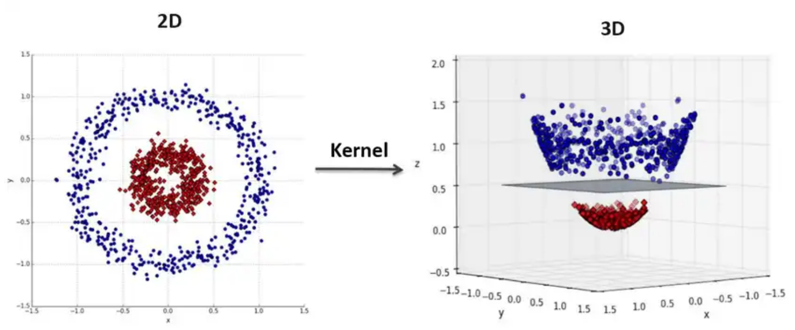


Figure 2.2: Illustration of the kernel trick. Source: https://borisburkov.net/2021-08-03-1/

product with each of the training points. An easier alternative way to compute this inner product efficiently would be to possibly use a direct function of the input features instead of using the explicit mapping ψ. A function that performs this direct computation is known as a kernel function.

Definition 2.2.2 (Kernel function [128]). *A kernel is a function $K : X \times X \to \mathbb{R}$ such that for all $\mathbf{x}, \bar{\mathbf{x}} \in X$ it satisfies*

$$K(\mathbf{x}, \bar{\mathbf{x}}) = \langle \psi(\mathbf{x}), \psi(\bar{\mathbf{x}}) \rangle,$$

where ψ is a mapping from $X \subseteq \mathbb{R}^d$ to an (inner product) feature space $\psi(\mathbf{X})$.

Kernel functions help to cut down on the computational cost of calculating the inner product between feature maps of every two data points. If the kernel function is known, a direct evaluation of the kernel function at each input point would give the inner product values. We give a simple example (taken with modification from [128]) below in order to show why using a kernel function can be advantageous.

Example: In this particular example, we illustrate the advantage of using kernel functions commonly known as the 'kernel trick'. Suppose that for given input data $\mathbf{x} = [x_1, x_2]^T \in \mathbb{R}^2$, the nonlinear feature map $\psi : \mathbb{R}^2 \to \mathbb{R}^3$ is defined by

$$\psi\left(\begin{bmatrix} x_1 \\ x_2 \end{bmatrix}\right) = \begin{bmatrix} x_1^2 \\ x_2^2 \\ \sqrt{2}x_1x_2 \end{bmatrix}.$$

Then for the above choice of ψ, we can see that,

$$\begin{aligned}\langle \psi(\mathbf{x}), \psi(\mathbf{y})\rangle &= \begin{bmatrix} y_1^2 & y_2^2 & \sqrt{2}y_1y_2\end{bmatrix}\begin{bmatrix} x_1^2 \\ x_2^2 \\ \sqrt{2}x_1x_2\end{bmatrix} \\ &= x_1^2y_1^2 + x_2^2y_2^2 + 2x_1x_2y_1y_2 \\ &= (x_1y_1 + x_2y_2)^2 \\ &= \left\langle \begin{bmatrix} x_1 \\ x_2 \end{bmatrix}, \begin{bmatrix} y_1 \\ y_2\end{bmatrix}\right\rangle^2.\end{aligned}$$

Thus we can choose its associated kernel function defined by $K(\mathbf{x}, \mathbf{y}) = \langle \mathbf{x}, \mathbf{y}\rangle^2$ for $\mathbf{x}, \mathbf{y} \in \mathbb{R}^2$ since $\langle \psi(\mathbf{x}), \psi(\mathbf{y})\rangle = \langle \mathbf{x}, \mathbf{y}\rangle^2$. Suppose for two input data $\mathbf{x} = [2, 3]^T$ and $\mathbf{y} = [4, 5]^T$, we use the definition of the function ψ for the evaluation of the inner product. Then computations are as follows:

$$\psi(\mathbf{x}) = \psi\left(\begin{bmatrix} 2 \\ 3\end{bmatrix}\right) = \begin{bmatrix} 4 \\ 9 \\ 6\sqrt{2}\end{bmatrix} \text{ and } \psi\left(\begin{bmatrix} 4 \\ 5\end{bmatrix}\right) = \begin{bmatrix} 16 \\ 25 \\ 20\sqrt{2}\end{bmatrix}.$$

$$\text{Thus, } \langle \psi(\mathbf{x}), \psi(\mathbf{y})\rangle = 4(16) + 9(25) + 2(6)(20) = 529.$$

On the other hand, using the function $K(\mathbf{x}, \mathbf{y}) = \langle \mathbf{x}, \mathbf{y}\rangle^2$ directly, we get

$$\langle \psi(\mathbf{x}), \psi(\mathbf{y})\rangle = \left\langle \begin{bmatrix} 2 \\ 3\end{bmatrix}, \begin{bmatrix} 4 \\ 5\end{bmatrix}\right\rangle^2 = (8 + 15)^2 = 529.$$

The above example demonstrates the advantage of using a kernel function for direct computation of the inner product. Thus in Eq. (2.23), the kernel function can be used directly for computation of $\psi(\mathbf{X})^T\psi(\mathbf{X})$. The method of using a kernel function is especially beneficial if the input dimension d is very large. Thus, going back to the example of the classification problem in $\mathbb{R}^2$, we can observe from Figure 2.2 that although the given set of data points may not be separable, lifting the points into a three-dimensional space can make them separable. For example, if we use the quadratic function $\langle \mathbf{x}, \mathbf{y}\rangle^2$ on each data point, we can lift the set of points in the $\mathbb{R}^2$ plane into a three-dimensional parabola-like structure. The lifting separates the red and blue points allowing us to pass a decision hyperplane for classification. Some other well-known examples of kernel functions are:

1. Linear kernel: $\forall \mathbf{x}, \bar{\mathbf{x}} \in X \subseteq \mathbb{R}^d$, $K(\mathbf{x}, \bar{\mathbf{x}}) = \mathbf{x}^T\bar{\mathbf{x}}$.
2. Polynomial kernel: $\forall \mathbf{x}, \bar{\mathbf{x}} \in X \subseteq \mathbb{R}^d$ and $c \in \mathbb{R}$, $c \geq 0$, $n \geq 1$, $K(\mathbf{x}, \bar{\mathbf{x}}) = (\mathbf{x}^T\bar{\mathbf{x}} + c)^n$.
3. Gaussian kernel: $\forall \mathbf{x}, \bar{\mathbf{x}} \in X \subseteq \mathbb{R}^d$ and $\sigma > 0$, $K(\mathbf{x}, \bar{\mathbf{x}}) = \exp\left(\dfrac{-\|\mathbf{x} - \bar{\mathbf{x}}\|^2}{2\sigma^2}\right)$.
4. $\forall \mathbf{x}, \bar{\mathbf{x}} \in X \subseteq \mathbb{R}^d$ and $a, b \in \mathbb{R}$, $K(\mathbf{x}, \bar{\mathbf{x}}) = \tanh(a(\mathbf{x}^T\bar{\mathbf{x}}) + b)$. (Note that this kernel is valid only under particular choices of $a, b, \mathbf{x}, \bar{\mathbf{x}}$.)

Thus, if the kernel function representing a feature map is known, one can easily replace the computation of the inner product $\langle \psi(\mathbf{x}), \psi(\bar{\mathbf{x}}) \rangle$ with the evaluation of the function $K(\mathbf{x}, \bar{\mathbf{x}})$ for $\mathbf{x}, \bar{\mathbf{x}} \in X \subseteq \mathbb{R}^d$ to cut down on computational costs. Note that for a function to qualify as a valid kernel, it must correspond to an inner product in some feature space. An easy way to check if a function K is a valid kernel function or not is to check if the kernel matrix (obtained from the Gram matrix) $\mathbf{K}$ generated from the given function K is a positive semi-definite matrix or not (see Definition 2.2.1 and Theorem 2.2.4). We give the definitions of Gram and kernel matrix for the sake of reference below.

Definition 2.2.3. *(Gram and Kernel Matrix) Suppose* $X = \{\mathbf{x}^{(1)}, \cdots, \mathbf{x}^{(m)}\} \subseteq \mathbb{R}^d$ *be a set of* m *vectors. Then the Gram matrix* $\mathbf{G}$ *is a* $m \times m$ *matrix whose entries are given by*

$$\mathbf{G}_{ij} = \mathbf{x}^{(i)T}\mathbf{x}^{(j)}.$$

The 'Kernel matrix' is the Gram matrix generated by the set of vectors $\{\psi(\mathbf{x}^{(1)}), \cdots, \psi(\mathbf{x}^{(m)})\}$.

Mercer's condition stated below gives the conditions for a function K to be a valid kernel without the actual computation of the kernel matrix.

Theorem 2.2.4 (Mercer's Theorem)**.** *Suppose* $K : X \times X \to \mathbb{R}$ *is a function. Then* K *is positive semi-definite if and only if*

$$\sum_{i=1}^{m}\sum_{j=1}^{m} c_i c_j K(\mathbf{x}^{(i)}, \mathbf{x}^{(j)}) \geq 0,$$

for all $\mathbf{x}^{(i)}, \mathbf{x}^{(j)} \in X \subseteq \mathbb{R}^d$ *and real numbers* $c_1, \cdots, c_m$.

Definition 2.2.5 (Reproducing Kernel Hilbert Space (RKHS) associated with a Kernel)**.** *Let* $K : X \times X \to \mathbb{R}$ *be a symmetric and positive semi-definite kernel. Then, there exists a Hilbert space* H *and a mapping* ψ *from* X *to* H *such that*

$$\forall \mathbf{x}, \bar{\mathbf{x}} \in X, K(\mathbf{x}, \bar{\mathbf{x}}) = \langle \psi(\mathbf{x}), \psi(\bar{\mathbf{x}}) \rangle.$$

Furthermore, H *has the reproducing property given by*

$$\forall h \in H, \forall \mathbf{x} \in X, h(\mathbf{x}) = \langle h, K(\mathbf{x}, \cdot) \rangle.$$

Then H *is called a reproducing kernel Hilbert space associated with* K*. The definition implies that the evaluation of* h *at* $\mathbf{x}$ *can be written as,*

$$h(\mathbf{x}) = \sum_{k=1}^{m} \alpha_k K(\mathbf{x}^{(k)}, \mathbf{x}),$$

for some $\alpha_1, \cdots, \alpha_k \in \mathbb{R}$.

The RKHS H associated with kernel K is called a feature space associated with feature mapping ψ. In general, the feature space associated with kernel map K is not unique. The above theorem implies that symmetric positive semi-definite kernels can be used to implicitly define a feature space or feature vectors which play an important role in determining the success of learning algorithms. Thus one can look for useful symmetric positive semi-definite kernels that represent a versatile feature space rather than seeking the feature by itself. Although kernel methods can cut down the cost of evaluating the inner product, it is not easy to find a function that would separate the given data. Additionally, kernel-based methods require the computation of kernel matrix on the data which scales poorly with the size of the training dataset. Thus a dataset with half a million training examples might take days to train on modern workstations. To simplify the kernel methods, random features were introduced in [109]. We discuss the random feature method and some recent results corresponding to them in the following section.

2.3 Learning using Random Feature Methods

The kernel trick or kernel function is based on the fact that any positive definite function $K(\mathbf{x}, \mathbf{y})$ with $\mathbf{x}, \mathbf{y} \in X \subseteq \mathbb{R}^d$ defines an inner product between data points transformed through ψ and can be quickly computed as $\langle \psi(\mathbf{x}), \psi(\mathbf{y}) \rangle = K(\mathbf{x}, \mathbf{y})$. This helps to cut down on the cost of computing the inner product between every two data points. However, even this 'cut down' cost can be very high. Also, this method requires one to know the kernel function beforehand and requires evaluations of $K(\mathbf{x}, \mathbf{y})$. As a result, large training sets incur large computational and storage costs [109]. The authors in [109] propose the use of an explicit mapping of the data to a low-dimensional Euclidean inner product space using a randomized feature map $\xi : \mathbb{R}^d \to \mathbb{R}^N$ so that the inner product between a pair of transformed points approximates their kernel evaluation:

$$K(\mathbf{x}, \mathbf{y}) = \langle \psi(\mathbf{x}), \psi(\mathbf{y}) \rangle \approx \xi(\mathbf{x})^T \xi(\mathbf{y}).$$

For kernel methods, evaluation on a new point $\mathbf{x}$ is computed by $f(\mathbf{x}) = \sum\limits_{k=1}^{m} c_k K(\mathbf{x}^{(k)}, \mathbf{x})$, which requires $O(Nd)$ operations, while the RFM can be evaluated by $f(\mathbf{x}) = \mathbf{c}^T \xi(\mathbf{x})$, which requires only $O(N + d)$ operations [109]. An alternative perspective is to view the RFM as a nonlinear randomized function approximation. From the neural network point of view, an RFM is a two-layer network with a randomized but fixed single hidden layer. We define RFM below [109–111].

Definition 2.3.1 (Random Feature Approximation). *Let $f : \mathbb{R}^d \to \mathbb{C}$ be a given unknown function. Then RFM approximation of f takes the form*

$$f(\mathbf{x}) \approx \mathbf{c}^T \xi(\mathbf{x}) = \mathbf{c}^T \phi(\mathbf{W}^T \mathbf{x} + \mathbf{b}), \tag{2.26}$$

where $\mathbf{x} \in \mathbb{R}^d$ is the input data, ξ is the feature map defined through a random matrix $\mathbf{W} \in \mathbb{R}^{d \times N}$, bias vector $\mathbf{b} \in \mathbb{R}^N$ and a pre-defined non-linear function ϕ, and $\mathbf{c} \in \mathbb{C}^N$ is the final (learnt) weight layer.

In the above definition, we assume that the entries of the matrix $\mathbf{W} = [\boldsymbol{\omega}_{j,k}]$ are independent and identically distributed (i.i.d.) random variables generated by the (user-defined) probability density function $\rho(\boldsymbol{\omega})$ i.e. $\boldsymbol{\omega}_{j,k} \sim \rho(\boldsymbol{\omega})$ for all $1 \le j \le d$ and $1 \le k \le N$. Similarly, the components of the bias vector $\mathbf{b}$ are also sampled i.i.d from a predefined distribution. The output layer $\mathbf{c}$ is trained, while the hidden layer $\mathbf{W}$ is fixed. The main idea of random feature methods is to transform the input with the map ϕ and apply linear methods to approximate the solution of the nonlinear kernel machine. The random feature maps help in the quick evaluation of the machine. Thus, given a collection of m measurements, the objective is defined below.

Definition 2.3.2 (Optimization Problem for Random Features)**.** *Suppose the inputs (and outputs) are arranged column-wise in the matrix $\mathbf{X} \in \mathbb{R}^{d\times m}$ (and $\mathbf{Y} \in \mathbb{C}^{1\times m}$), then random feature regression problem becomes training $\mathbf{c}$ by optimizing:*

$$\min_{\mathbf{c}\in\mathbb{C}^N} \|\mathbf{Y} - \mathbf{c}^T\phi(\mathbf{W}^T\mathbf{X} + \mathbf{b})\|_2^2 + \mathcal{R}(\mathbf{c}), \tag{2.27}$$

with some penalty function $\mathcal{R} : \mathbb{C}^N \to \mathbb{R}$ and weight matrix $\mathbf{W} \in \mathbb{R}^{N\times d}$, bias vector $\mathbf{b} \in \mathbb{R}^N$ and a predefined nonlinear function ϕ.

While the function ϕ can be chosen as any non-linear function, certain choices of ϕ lead to the approximation of well-known kernel functions. One such well-known random feature is the Random Fourier Feature (RFF) which approximates shift-invariant kernels [109–111]. A brief discussion on RFF and shift-invariant kernels below demonstrates the idea of using random features to construct and approximate kernels. We also recall some relevant definitions and theorems [109].

Definition 2.3.3 (Shift Invariant Kernels)**.** *A kernel function K on $\mathbb{R}^d$ is called shift-invariant if $K(\mathbf{x}, \mathbf{y}) = g(\mathbf{x} - \mathbf{y})$, for some complex-valued positive definite function g on $\mathbb{R}^d$.*

Theorem 2.3.4 (Bochner's Theorem)**.** *A continuous kernel $K(\mathbf{x}, \mathbf{y}) = K(\mathbf{x} - \mathbf{y})$ on $\mathbb{R}^d$ is positive definite if and only if $K(\mathbf{x} - \mathbf{y})$ is the Fourier transform of a non-negative measure.*

Random Fourier Features. We give a well-known example from [109] of how random feature maps can be constructed to approximate kernel functions. As a consequence of Bochner's theorem, the authors in [109] define $\xi_{\boldsymbol{\omega}}(\mathbf{x}) = \exp(i\boldsymbol{\omega}^T\mathbf{x})$ such that

$$K(\mathbf{x} - \mathbf{y}) = \int_{\mathbb{R}^d} \rho(\boldsymbol{\omega}) \exp(i\boldsymbol{\omega}^T(\mathbf{x} - \mathbf{y}))d\boldsymbol{\omega} = \mathbb{E}_{\boldsymbol{\omega}}[\xi_{\boldsymbol{\omega}}(\mathbf{x})\xi_{\boldsymbol{\omega}}(\mathbf{y})^*], \tag{2.28}$$

and $\xi_{\boldsymbol{\omega}}(\mathbf{x})\xi_{\boldsymbol{\omega}}(\mathbf{y})^*$ gives an unbiased estimate of $K(\mathbf{x}, \mathbf{y})$ when $\boldsymbol{\omega}$ is drawn from ρ. Thus it is possible to obtain a real-valued mapping satisfying $\mathbb{E}_{\boldsymbol{\omega}}[\xi_{\boldsymbol{\omega}}(\mathbf{x})\xi_{\boldsymbol{\omega}}(\mathbf{y})] = K(\mathbf{x}, \mathbf{y})$ by choosing $\xi_{\boldsymbol{\omega}}(\mathbf{x}) = \sqrt{2}\cos(\boldsymbol{\omega}^T\mathbf{x} + b)$, where $\boldsymbol{\omega}$ is drawn from $\rho(\boldsymbol{\omega})$ and b is drawn uniformly from $[0, 2\pi]$.

Although RFM has received considerable attention due to its easy implementation, interpretability, and quick evaluation, it requires more measurements than trainable parameters for accuracy. This limits their usage for data-scarce applications. In order to make random features suitable for such applications, sparse random feature expansions (SRFE) introduced in [66] were developed using ideas from compressive sensing to generate random feature expansions. The theoretical guarantees of SRFE hold even in the data-scarce setting. Below we discuss some key definitions and results of SRFE from [66] and [36].

Definition 2.3.5 (Order-q Additive Functions). *Fix $d, q, K \in \mathbb{N}$ with $1 \leq q \leq d$. A function $f : \mathbb{R}^d \to \mathbb{C}$ is called an order-q additive function with at most K terms if there exist K complex-valued functions $g_1, \ldots, g_K : \mathbb{R}^q \to \mathbb{C}$ such that*

$$f(\mathbf{x}) = \frac{1}{K}\sum_{j=1}^{K} g_j(\mathbf{x}|_{\mathcal{S}_j}), \tag{2.29}$$

where for each $j \in [K]$, $\mathcal{S}_j \subseteq [d]$, $\mathcal{S}_j$ has q distinct indices, and $\mathcal{S}_j \neq \mathcal{S}_{j'}$ for $j \neq j'$. Here $\mathbf{x}|_{\mathcal{S}_j}$ denotes the restriction of $\mathbf{x} \in \mathbb{R}^d$ onto $\mathcal{S}_j$.

The definition above is motivated by the fact that a lot of high-dimensional functions arising from physical systems are dominated by a few terms i.e., each of the terms depends only on a subset of the input variables, say q out of the d with $q \ll d$ [45,66,84]. We recall some key definitions and results from [66] corresponding to the SRFE model.

Definition 2.3.6 (Bounded $\rho-$norm functions from [66]). *Fix a probability density function $\rho : \mathbb{R}^d \to \mathbb{R}$ and function $\phi : \mathbb{R}^d \times \mathbb{R}^d \to \mathbb{C}$. A function $f : \mathbb{R}^d \to \mathbb{C}$ has finite ρ-norm with respect to $\phi(\mathbf{x}; \boldsymbol{\omega})$ if it belongs to the class defined by*

$$\mathcal{F}(\phi, \rho) = \left\{ f(\mathbf{x}) = \int_{\boldsymbol{\omega} \in \mathbb{R}^d} \alpha(\boldsymbol{\omega})\phi(\mathbf{x}; \boldsymbol{\omega})\, d\boldsymbol{\omega} \;\middle|\; \|f\|_\rho := \sup_{\boldsymbol{\omega}} \left|\frac{\alpha(\boldsymbol{\omega})}{\rho(\boldsymbol{\omega})}\right| < \infty \right\}.$$

Theorem 2.3.7 (Generalization bound for bounded ρ-norm functions from [66]). *Let $f \in F(\phi, \rho)$, where $\phi(\mathbf{x}; \boldsymbol{\omega}) = \exp(i\langle \mathbf{x}, \boldsymbol{\omega}\rangle)$ and $\rho(\boldsymbol{\omega})$ is the density corresponding to a spherical Gaussian with variance σ^2, $\mathcal{N}(0, \sigma^2\mathbf{I}_d)$. For a fixed γ, consider a set of data samples $\mathbf{x}_1, \cdots, \mathbf{x}_m \sim \mathcal{N}(0, \gamma^2\mathbf{I}_d)$ and frequencies $\boldsymbol{\omega}_1, \cdots, \boldsymbol{\omega}_N \sim \mathcal{N}(0, \sigma^2\mathbf{I}_d)$. The measurement noise e_k is either bounded by $E = 2\nu$ or to be drawn i.i.d. from $\mathcal{N}(0, \nu^2)$. Let $\mathbf{A} \in \mathbb{C}^{m\times N}$ denote the associated random feature matrix where $a_{k,j} = \phi(\mathbf{x}_k; \boldsymbol{\omega}_j)$. Let $f^\sharp(\mathbf{x}) = \sum_{j=1}^{N} c_j^\sharp \phi(\mathbf{x}; \boldsymbol{\omega}_j)$, where c_j is the j^{th} component of $\mathbf{c}$ obtained by solving*

$$\mathbf{c}^\sharp = \operatorname*{argmin}_{\mathbf{c} \in \mathbb{C}^N} \|\mathbf{c}\|_1 \;\; s.t. \;\; \|\mathbf{A}\mathbf{c} - \mathbf{y}\| \leq \eta\sqrt{m},$$

with $\eta = \sqrt{2(\epsilon^2\|f\|_\rho^2 + E^2)}$ and with the additional pruning step

$$f^\sharp(\mathbf{x}) = \sum_{j \in \mathcal{S}^\sharp} c_j^\sharp \phi(\mathbf{x}; \boldsymbol{\omega}_j),$$

where $S^\sharp$ is the support set of the s largest (in magnitude) coefficients of $\mathbf{c}^\sharp$. For a given s, if the feature parameters σ and N, the confidence parameter δ, and the accuracy parameter ϵ are chosen so that the following conditions hold:

1. *$\gamma - \sigma$ principle*
$$\gamma^2\sigma^2 \geq \frac{1}{2}(13s)^{\frac{2}{d}}, \tag{2.30}$$

2. *Number of features*
$$N = \frac{4}{\epsilon^2}\left(1 + 4\gamma\sigma d\sqrt{1+\sqrt{\frac{12}{d}\log\frac{m}{\delta}}} + \sqrt{\frac{1}{2}\log\frac{1}{\delta}}\right)^2, \tag{2.31}$$

3. *Number of measurements*
$$m \geq 4(2\gamma^2\sigma^2+1)^d \log\frac{N^2}{\delta}, \tag{2.32}$$

4. *Dimensionality*
$$d \geq \frac{4\log\left(\frac{N^2}{\delta}\right)}{\log\left(\frac{\gamma^2\sigma^2}{\mathrm{e}\log(2\gamma^2\sigma^2+1)}\right)}. \tag{2.33}$$

Then with probability at least $1-6\delta$ the following error bound holds

$$\sqrt{\int_{\mathbb{R}^d} |f(\mathbf{x}) - f^\sharp(\mathbf{x})|^2 d\mu} \leq C'\left(1 + N^{\frac{1}{2}}s^{\frac{-1}{2}}m^{\frac{-1}{4}}\log^{\frac{1}{4}}\left(\frac{1}{\delta}\right)\right)\kappa_{1,s}(\mathbf{c}^\star) \tag{2.34}$$

$$+ C\left(1 + N^{\frac{1}{2}}m^{\frac{-1}{4}}\log^{\frac{1}{4}}\left(\frac{1}{\delta}\right)\right)\sqrt{\epsilon^2\|f\|_\rho^2 + 4\nu^2}, \tag{2.35}$$

where $C, C' > 0$ are constants and $\mathbf{c}^\star$ is the vector

$$\mathbf{c}^\star = \frac{1}{N}\left[\frac{\alpha(\boldsymbol{\omega}_1)}{\rho(\boldsymbol{\omega}_1)}, \cdots, \frac{\alpha(\boldsymbol{\omega}_N)}{\rho(\boldsymbol{\omega}_N)}\right]^T. \tag{2.36}$$

Theorem 2.3.8 (Generalization bound for order-q functions from [66])**.** *Let f be an order-q function of at most K terms as defined in Definition 2.3.5 such that each term g_ℓ, $\ell = 1, \cdots, K$ belongs to $\mathcal{F}(\phi, \rho)$, where $\phi(\mathbf{x};\boldsymbol{\omega}) = \phi(\langle \mathbf{x}, \boldsymbol{\omega}\rangle) = \exp(i\langle \mathbf{x}, \boldsymbol{\omega}\rangle)$ and $\rho : \mathbb{R}^q \to \mathbb{R}$ is the density corresponding to a spherical Gaussian with variance σ^2, $\mathcal{N}(0, \sigma^2\mathbf{I}_d)$. For a fixed γ, consider a set of data samples $\mathbf{x}_1, \cdots, \mathbf{x}_m \sim \mathcal{N}(0, \gamma^2\mathbf{I}_d)$. Let $\boldsymbol{\omega}_1, \cdots, \boldsymbol{\omega}_N \sim \mathcal{N}(0, \sigma^2\mathbf{I}_d)$ be a complete set of $q-$sparse feature weights drawn from density ρ. The measurement noise e_k is either bounded by $E = 2\nu$ or to be drawn i.i.d. from $\mathcal{N}(0, \nu^2)$. Let $\mathbf{A} \in \mathbb{C}^{m\times N}$ denote the associated random feature matrix where $a_{k,j} = \phi(\langle \mathbf{x}_k, \boldsymbol{\omega}_j\rangle)$. Suppose*

$\eta = \sqrt{2(\epsilon^2\binom{d}{q}|||f|||^2 + E^2)}$, where $|||f||| = \frac{1}{K}\sum_{j=1}^{K}\|g_j\|_\rho$. Let $f^\sharp(\mathbf{x}) = \sum_{j=1}^{N} c_j^\sharp \phi(\mathbf{x};\boldsymbol{\omega}_j)$, where $\mathbf{c}^\sharp$ is obtained by solving

$$\mathbf{c}^\sharp = \operatorname*{argmin}_{\mathbf{c}} \|\mathbf{c}\|_1 \ s.t.\ \|\mathbf{A}\mathbf{c} - \mathbf{y}\| \leq \eta\sqrt{m},$$

with the additional pruning step

$$f^\sharp(\mathbf{x}) = \sum_{j\in\mathcal{S}^\sharp} c_j^\sharp \phi(\mathbf{x};\boldsymbol{\omega}_j),$$

where $S^\sharp$ is the support set of the s largest (in magnitude) coefficients of $\mathbf{c}^\sharp$. For a given s, if the feature parameters σ and N, the confidence parameter δ, and the accuracy parameter ϵ are chosen so that the following conditions hold

1. *$\gamma - \sigma$ uncertainty principle*
$$\gamma^2\sigma^2 \geq \frac{1}{2}(13s)^{\frac{2}{d}}, \tag{2.37}$$

2. *Number of features*
$$N = n\binom{d}{q} = \frac{4}{\epsilon^2}\left(1 + 4\gamma\sigma d\sqrt{1 + \sqrt{\frac{12}{d}\log\frac{m}{\delta}}} + \sqrt{\frac{q}{2}\log\frac{d}{\delta}}\right)^2, \tag{2.38}$$

3. *Number of measurements*
$$m \geq 4(2\gamma^2\sigma^2 + 1)^{\max\{2q-d,0\}}(\gamma^2\sigma^2+1)^{\min\{2q,2d-2q\}}\log\frac{N^2}{\delta}, \tag{2.39}$$

4. *Dimensionality*
$$q \geq \frac{4\log\left(\frac{N^2}{\delta}\right)}{\log\left(\frac{\gamma^2\sigma^2}{\mathrm{e}\log(2\gamma^2\sigma^2+1)}\right)}. \tag{2.40}$$

Then with probability at least $1 - 6\delta$ the following error bound holds

$$\sqrt{\int_{\mathbb{R}^d}|f(\mathbf{x}) - f^\sharp(\mathbf{x})|^2 d\mu} \leq C'\left(1 + N^{\frac{1}{2}}s^{\frac{-1}{2}}m^{\frac{-1}{4}}\log^{\frac{1}{4}}\left(\frac{1}{\delta}\right)\right)\kappa_{1,s}(\tilde{\mathbf{c}}^\star) \tag{2.41}$$

$$+ C\left(1 + N^{\frac{1}{2}}m^{\frac{-1}{4}}\log^{\frac{1}{4}}\left(\frac{1}{\delta}\right)\right)\sqrt{\epsilon^2\binom{d}{q}|||f|||^2 + E^2}, \tag{2.42}$$

where $C, C' > 0$ are constants and $\tilde{\mathbf{c}}^\star = [\tilde{c}_1^\star, \cdots, \tilde{c}_N^\star]^T \in \mathbb{C}^N$ is the vector with

$$\tilde{c}_j^\star = \frac{1}{K}\sum_{\ell=1}^{K}\tilde{c}_{\ell,j}^\star,\ \text{where}\ \tilde{c}_{\ell,j}^\star = \begin{cases} \dfrac{\alpha_\ell(\boldsymbol{\omega}_j)}{n\rho(\boldsymbol{\omega}_j)} & \text{if}\ \operatorname{supp}(\boldsymbol{\omega}_j) = S_\ell, \\ 0 & \text{otherwise.} \end{cases} \tag{2.43}$$

The function $\alpha_\ell(\boldsymbol{\omega})$ is the transform of g_ℓ from Definitions 2.3.5 and 2.3.6.

In general, it is difficult to fully quantify and understand the theoretical bounds due to its dependency on the interactions between the number of features, measurements, etc. The above two theorems demonstrate that although the generalization bounds consist of several terms, the order q assumption on the function can significantly reduce the bound in terms of the dimension d.

Theorem 2.3.7 can be proved using Lemma 2.3.9 and Lemma 2.3.10 stated below. These are Lemma 1 and Lemma 2 stated and proved in [66], respectively.

Lemma 2.3.9. *Fix the confidence parameter $\delta > 0$ and accuracy parameter $\epsilon > 0$. Suppose $f \in \mathcal{F}(\phi, \rho)$ where $\phi(\mathbf{x}, \boldsymbol{\omega}) = \exp(i\langle \mathbf{x}, \boldsymbol{\omega}\rangle)$, the data samples $\mathbf{x}_k$ have probability measure $\mu(\mathbf{x})$ and ρ is a probability distribution with finite second moment used for sampling the random weights $\boldsymbol{\omega}$. Suppose $N \geq \frac{1}{\epsilon^2}\left(1 + \sqrt{2\log\left(\frac{1}{\delta}\right)}\right)^2$, then with probability at least $1-\delta$, the following holds with respect to the draw of $\boldsymbol{\omega}_j$ for $j \in [N]$:*

$$\sqrt{\int_{\mathbb{R}^d} |f(\mathbf{x}) - f^\star(\mathbf{x})|^2 d\mu} \leq \epsilon \|f\|_\rho, \tag{2.44}$$

where $f^\star(\mathbf{x}) = \sum_{j=1}^{N} c_j^\star \exp(i\langle \mathbf{x}, \boldsymbol{\omega}_j\rangle)$, with $c_j^\star = \dfrac{\alpha(\boldsymbol{\omega}_j)}{N\rho(\boldsymbol{\omega}_j)}$.

Lemma 2.3.10. *Let $f \in \mathcal{F}(\phi, \rho)$, where the basis function is $\phi(\mathbf{x}; \boldsymbol{\omega}) = \exp(i\langle \mathbf{x}, \boldsymbol{\omega}\rangle)$. For a fixed γ and q, consider a set of data samples $\mathbf{x}_1, \cdots, \mathbf{x}_m \sim \mathcal{N}(\mathbf{0}, \gamma^2 \mathbf{I}_d)$ with $\mu(\mathbf{x})$ denoting the associated probability measure and weights $\boldsymbol{\omega}_1, \cdots, \boldsymbol{\omega}_N$ drawn from $\mathcal{N}(\mathbf{0}, \sigma^2\mathbf{I}_d)$. Assume that the noise is bounded by $E = 2\nu$ or that the noise terms e_j are drawn i.i.d from $\mathcal{N}(0, \nu^2)$. Let $\mathbf{A} \in \mathbb{C}^{m\times N}$ denote the associated random feature matrix where $a_{k,j} = \phi(\mathbf{x}_k; \boldsymbol{\omega}_j)$. Suppose $\eta = \sqrt{2(\epsilon^2\|f\|_\rho^2 + E^2)}$ and let $f^\sharp(\mathbf{x}) = \sum_{j=1}^{N} c_j^\sharp \phi(\mathbf{x}; \boldsymbol{\omega}_j)$, where $\mathbf{c}^\sharp$ is obtained by solving*

$$\mathbf{c}^\sharp = \operatorname*{argmin}_{\mathbf{c}} \|\mathbf{c}\|_1 \text{ s.t. } \|\mathbf{A}\mathbf{c} - \mathbf{y}\| \leq \eta\sqrt{m},$$

with the additional pruning step

$$f^\sharp(\mathbf{x}) := \sum_{j \in \mathcal{S}^\sharp} c_j^\sharp \phi(\mathbf{x}; \boldsymbol{\omega}_j),$$

where $S^\sharp$ is the support set of the s largest (in magnitude) coefficients of $\mathbf{c}^\sharp$. Let the random feature approximation $f^\star$ be defined as

$$f^\star(\mathbf{x}) := \sum_{j=1}^{N} c_j^\star \exp(i\langle \mathbf{x}, \boldsymbol{\omega}_j\rangle), \tag{2.45}$$

where

$$\mathbf{c}^\star = \frac{1}{N}\left[\frac{\alpha(\boldsymbol{\omega}_1)}{\rho(\boldsymbol{\omega}_1)}, \cdots, \frac{\alpha(\boldsymbol{\omega}_N)}{\rho(\boldsymbol{\omega}_N)}\right]^T. \tag{2.46}$$

For a given s, if the feature parameters σ and N, the confidence δ, and accuracy ϵ are chosen so that the following conditions hold:

$$\begin{aligned}
&\gamma^2\sigma^2 \geq \frac{1}{2}(13s)^{\frac{2}{d}},\\
&N = \frac{4}{\epsilon^2}\left(1 + 4\gamma\sigma d\sqrt{1+\sqrt{\frac{12}{d}\log\frac{m}{\delta}}} + \sqrt{\frac{1}{2}\log\frac{1}{\delta}}\right)^2,\\
&m \geq 4(2\gamma^2\sigma^2+1)^d \log\frac{N^2}{\delta},\\
&d \geq \frac{4\log\left(\frac{N^2}{\delta}\right)}{\log\left(\frac{\gamma^2\sigma^2}{e\log(2\gamma^2\sigma^2+1)}\right)}.
\end{aligned}$$

Then with probability at least $1-5\delta$ the following error bound holds

$$\sqrt{\int_{\mathbb{R}^d} |f^\star(\mathbf{x}) - f^\sharp(\mathbf{x})|^2 d\mu} \leq C^{'}\left(1 + N^{\frac{1}{2}} s^{\frac{-1}{2}} m^{\frac{-1}{4}} \log^{\frac{1}{4}}\frac{1}{\delta}\right)\kappa_{1,s}(\mathbf{c}^\star) \tag{2.47}$$

$$+ C\left(1 + N^{\frac{1}{2}} m^{\frac{-1}{4}} \log^{\frac{1}{4}}\frac{1}{\delta}\right)\sqrt{\epsilon^2\|f\|_\rho^2 + 4\nu^2}, \tag{2.48}$$

where $C, C^{'} > 0$ are constants

The theorem below is stated from Theorem 4.3 in [36] and gives the conditions needed to find an upper bound for the restricted isometry constant for a random feature matrix.

Theorem 2.3.11. *Let the data $\{\mathbf{x}_k\}_{k\in[m]}$ be drawn from $\mathcal{N}(\mathbf{0}, \gamma^2\mathbf{I}_d)$, the weights $\{\boldsymbol{\omega}_j\}_{j\in[N]}$ be drawn from $\mathcal{N}(\mathbf{0}, \sigma^2\mathbf{I}_d)$, and the random feature matrix $\mathbf{A} \in \mathbb{C}^{m\times N}$ be defined component-wise by $a_{k,j} = \exp(i\langle\mathbf{x}_k, \boldsymbol{\omega}_j\rangle)$. For $\eta_1, \eta_2, \delta \in (0,1)$ and some integer $s \geq 1$, if*

$$\begin{aligned}
m &\geq C_1\eta_1^{-2}\, s\, \log(\delta^{-1}),\\
\frac{m}{\log(3m)} &\geq C_2\eta_2^{-2}\, s\, \log^2(s)\log\left(\frac{N}{9\log(2m)} + 3\right),\\
\sqrt{\delta}\,\eta_1\,(4\gamma^2\sigma^2+1)^{\frac{d}{4}} &\geq N,
\end{aligned}$$

where C_1 and C_2 are universal positive constants, then with probability at least $1-2\delta$, the s restricted isometry constant of $\frac{1}{\sqrt{m}}\mathbf{A}$ is bounded by

$$\delta_s\left(\frac{1}{\sqrt{m}}\mathbf{A}\right) < 3\eta_1 + \eta_2^2 + \sqrt{2}\eta_2.$$

2.4 Dynamical Systems in Epidemiology

In this section, we discuss some key definitions and concepts corresponding to dynamical systems in epidemiology. We summarize all the important definitions and concepts required to understand the contents of the thesis. For a more detailed understanding, we refer the reader to [83, 98].

Definition 2.4.1 (Ordinary Differential Equation)**.** *An ordinary differential equation (ODE) is an equality involving a function and its derivatives taken with respect to only one variable. An ODE of order n is defined as*

$$F(x, y, y', \cdots, y^{(n)}) = 0,$$

where y is a function of x, $y' = \dfrac{dy}{dx}$ is the first derivative with respect to x, and $y^{(n)} = \dfrac{d^n y}{dx^n}$ is the n^{th} derivative with respect to x.

Definition 2.4.2 (Dynamical System)**.** *It is a system of ODEs whose general evolution equations are given by*

$$\frac{d\mathbf{x}}{dt} = F(t, \mathbf{x}), \tag{2.49}$$

where $\mathbf{x} = \mathbf{x}(t) \in \mathbb{R}^d$ is the d-dimensional state vector and $F : \mathbb{R} \times \mathbb{R}^d \to \mathbb{R}^d$ is the functional operator.

Definition 2.4.3 (Delay Differential Equations with Constant Delay from [103])**.** *Let $\mathbf{x}(t) \in \mathbb{R}^d$ be the state vector. Then the delay differential equation is given by*

$$\begin{cases} \dfrac{d\mathbf{x}(t)}{dt} = f\left(t, \mathbf{x}(t), \mathbf{x}(t - \tau(t, \mathbf{x}(t)))\right), & t_0 \leq t \leq t_f \\ \mathbf{x}(t) = \phi(t), & t_0 - \tau \geq t \leq t_0 \end{cases} \tag{2.50}$$

where $\mathbf{x}(t) : [t_0, t_f] \longmapsto \mathbb{R}^d$, $f : [t_0, t_f] \times \mathbb{R}^d \times \mathbb{R}^d \to \mathbb{R}^d$ and τ is a positive constant.

For modeling in epidemiology, compartmental models are the most common techniques which are based on differential equations (deterministic or stochastic). The most common models involve a system of ODEs where the equations depend on how people may progress between the assigned compartments in the population. For example, the population may be stratified into compartments with labels S, I, or R, (Susceptible, Infectious, or Recovered, respectively), and the equations are based on how the rate of change of each of these compartments change with time as a disease progresses. Some well-known compartmental models used in this research study are defined below.

Definition 2.4.4 (SIR Compartmental Model)**.** *Susceptible-Infectious-Recovered (SIR) divides the population into three groups: Susceptible (healthy), Infectious (infected and infectious), and Recovered (recovered or died). Let S, I, and R be the population of Susceptible,* *Infected, and Recovered groups, respectively. Under the SIR model, we have*

$$\begin{aligned} \frac{dS}{dt} &= -\frac{\beta S I}{P} \\ \frac{dI}{dt} &= \frac{\beta S I}{P} - \gamma I \\ \frac{dR}{dt} &= \gamma I, \end{aligned} \tag{2.51}$$

where β is infection rate, γ is the removal rate, and P denotes the population.

Definition 2.4.5 (SEIR Compartmental Model)**.** *Susceptible-Exposed-Infectious-Recovered (SEIR) divides the population into four groups: Susceptible (healthy), Exposed (infected but not yet infectious), Infectious (infected and infectious), and Recovered (recovered or died). Let S, E, I, R be the population of Susceptible, Exposed, Infected, and Recovered groups, respectively. Under the SEIR model, we have*

$$\begin{aligned} \frac{dS}{dt} &= -\frac{\beta S I}{P} \\ \frac{dE}{dt} &= \frac{\beta S I}{P} - \sigma E \\ \frac{dI}{dt} &= \sigma E - \gamma I \\ \frac{dR}{dt} &= \gamma I, \end{aligned} \tag{2.52}$$

where β is infection rate, σ is the latency rate, γ is the removal rate, and P denotes the population.

Definition 2.4.6 (SμEIR Compartmental Model)**.** *SμEIR model described in [159] which takes into account unreported cases in the SEIR model defined in Definition 2.4.5. The dynamics of this model are given by*

$$\begin{aligned} \frac{dS}{dt} &= -\frac{\beta (I+E) S}{P} \\ \frac{dE}{dt} &= \frac{\beta (I+E) S}{P} - \sigma E \\ \frac{dI}{dt} &= \mu \sigma E - \gamma I \\ \frac{dR}{dt} &= \gamma I, \end{aligned} \tag{2.53}$$

where β is infection rate, σ is the latency rate, γ is the removal rate, and P denotes the population. μ represents the discovery rate which characterizes the ratio of the exposed cases that are confirmed and reported to the public. It reflects the unreported/undiscovered cases.

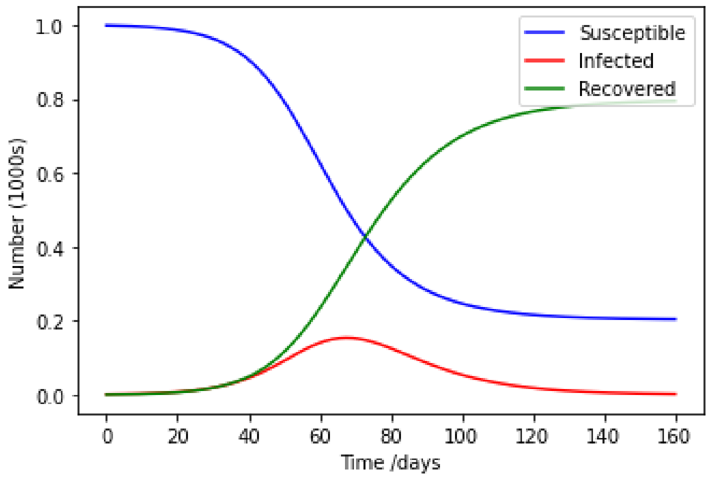


Figure 2.3: Plot of solution for the SIR model

Compartmental models (or variations around them) are the most widely used models for predictions in epidemiology owing to their ease of understanding and implementation. They can be numerically solved. In Figure 2.3, we plot the solutions of the SIR model for demonstration. We solve the SIR model numerically with $dt = 0.1$ for $t \in [0, 160]$, initial conditions $(S_0, I_0, R_0) = (999, 1, 0)$ and parameters $(\beta, \gamma) = (0.2, 0.1)$. Note that the system is normalized by dividing by the population $N = 1000$ and hence the solution plot in Figure 2.3 gives the proportion of the population on the y-axis. The plots demonstrate the expected behavior for each of the variables where the number of susceptible people comes down and the number of recovered goes up with time. As for the other variable, the number of infected people keeps going up and eventually reaches a peak after which it starts to drop as more and more people recover and develop immunity to the infection. The interactions of the variables depend a lot on the parameters of the model β and γ as well as the underlying assumptions. For example, in the example above we assume that the population is constant and the recovered people gain immunity against the disease.

When learning models, often one needs to tune hyperparameters so that the model can solve the machine-learning problem optimally. We use one such method in our research to tune hyperparameters called the Bayesian Information Criterion (BIC) defined below.

Definition 2.4.7 (Bayesian Information Criterion)**.** *The Bayesian information criterion (BIC) for a candidate model is defined as,*

$$BIC = -2 \log \mathcal{L} + p \log n,$$

where $\mathcal{L}$ is the loss value, p is the number of parameters in the model and n is the number of observations/measurements.

Predictions using dynamical systems may become additionally challenging if the data available is only corresponding to a single variable while the underlying system is multi-dimensional. In such situations, delay embedding theorems shed some light on systems recovered from delay time embeddings of a single variable belonging to a multidimensional system.

Definition 2.4.8 (Diffeomorphism)**.** *Given two manifolds $\mathcal{M}$ and $\mathcal{N}$, a differentiable map $f : \mathcal{M} \to \mathcal{N}$ is called a diffeomorphism if it is one-one onto and its inverse $f^{-1} : \mathcal{N} \to \mathcal{M}$ is differentiable as well.*

Theorem 2.4.9 (Takens' Theorem)**.** *Let $\mathcal{M}$ be a compact manifold of dimension d. For pairs (φ, y), with $\varphi : \mathcal{M} \to \mathcal{M}$ a smooth diffeomorphism and $y : \mathcal{M} \to \mathbb{R}$ a smooth function, it is generic property that the map $\psi_{(\varphi,y)} : \mathcal{M} \to \mathbb{R}^{2d+1}$, defined by*

$$\psi_{(\varphi,y)}(\mathbf{x}) = (y(\mathbf{x}), y(\varphi(\mathbf{x})), ..., y(\varphi^{2d})(\mathbf{x})) \tag{2.54}$$

is an embedding.

Takens' theorem gives the conditions under which a system can be recovered up to diffeomorphism from a single variable. The idea of using this theorem has already been explored in [26, 34, 76]. We elaborate on the application of this theorem via an example.

Example. A well-known example of a nonlinear dynamical system is the Lorenz system:

$$\begin{aligned} \frac{dx_1}{dt} &= \sigma(x_2 - x_1), \\ \frac{dx_2}{dt} &= x_1(\rho - x_3) - x_2, \\ \frac{dx_3}{dt} &= x_1 x_2 - \beta x_3. \end{aligned} \tag{2.55}$$

Takens' Theorem can be used to recover a system diffeomorphic to the above system through time embeddings of a single variable. Given the time measurement data, we are interested in finding the corresponding governing equations. Suppose m measurements corresponding to $x_1(t)$ is given for $t = t_1, \cdots, t_m$. Choose an embedding dimension p which is at least $2d+1$, where d denotes the dimension of the dynamical system. We build the matrix H using delay embeddings as illustrated below.

$$H = \begin{bmatrix} x_1(t_1) & x_1(t_2) & \cdots & x_1(t_{m-p+1}) \\ x_1(t_2) & x_1(t_3) & \cdots & x_1(t_{m-p+2}) \\ \vdots & \vdots & \vdots & \vdots \\ x_1(t_p) & x_1(t_{p+1}) & \cdots & x_1(t_m) \end{bmatrix}. \tag{2.56}$$

Then each row of H contains data corresponding to a p dimension system diffeomorphic to the original system. In Figure 2.4 (left) we plot the solution of the Lorenz system. The system is solved using the Python ODE solver with $dt = 0.01$, 10000 timesteps and parameters $(\sigma, \rho, \beta) = (10, 28, 2.667)$, initial conditions $(x_1(0), x_2(0), x_3(0)) = (0, 1, 1.05)$. We build the matrix H and to visualize the properties of the given theorem, three randomly

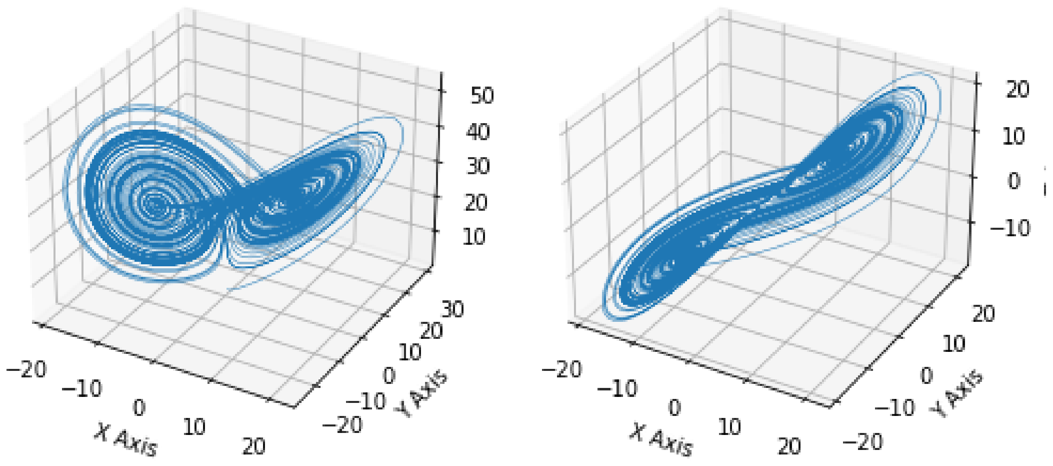


Figure 2.4: Plots depicting Takens' Delay Embedding theorem. Left: Solutions of the Lorenz system. Right: Plot of three (out of $2d + 1$) variables of a diffeomorphic system recovered from applying Takens' Delay Embedding theorem to the Lorenz system.

chosen rows (variables) are plotted i.e., we choose rows one, six, and seven from H as three variables with $m - p$ (we choose $p = 2d + 1 = 7$) points and plot them. From Figure 2.4 it can be seen that the plot of the three new variables is like a reshaped version of the original one.

2.5 Diffusion Models

This section explains the background for understanding diffusion models. We include some basic definitions from probability and statistics and include some basic derivations for the formulation of diffusion models. For a more detailed reading, we refer the readers to [70, 116, 152].

Definition 2.5.1 (Sample Space)**.** *Denoted as S, a sample space is the set that lists all possible outcomes of an unknown experiment.*

Definition 2.5.2 (Event)**.** *Any subset A of the sample space is called an event.*

Definition 2.5.3 (Probability Measure)**.** *A probability measure is a real-valued function that assigns to each event A in the sample space a probability $P(A)$ with the following properties:*

1. $P(A) \geq 0$ *and* $P(A) \in [0, 1]$.
2. $P(A) = 0$ *for* $A = \varphi$ *i.e., the empty set.*

3. $P(S) = 1$.

4. If $A_1, A_2, \cdots$ *is a countable sequence of disjoint events then*

$$P(A_1 \cup A_2 \cup \cdots) = P(A_1) + P(A_2) + \cdots$$

To understand the above definition consider an example of tossing a coin twice with H denoting "Heads" and T for "Tails". Then the sample space is $S = \{HH, HT, TH, TT\}$ any subset $A = \{HH, HT\}$ (or $A = \{HH, TT, TH\}$, $A = S$, $A = \varphi$) is an event.

Definition 2.5.4 (Random Variable)**.** *A function X from the sample space S to the set of all real numbers is called a random variable. (Note that the random variable X can be either discrete or continuous.)*

Definition 2.5.5 (Distribution)**.** *If X is a random variable, then the distribution of X is the collection of probabilities $P(X \in B)$ for all subsets B of the real numbers.*

Definition 2.5.6 (Density Function)**.** *A function $f : \mathbb{R} \to \mathbb{R}$ is a density function if $f(x) \geq 0 \ \forall x \in \mathbb{R}$ and $\int\limits_{-\infty}^{\infty} f(x)dx = 1$.*

Definition 2.5.7 (Random Vector)**.** *Also known as a multivariate random variable, it is a column vector $\mathbf{X} = [X_1, \cdots, X_n]^T$ such that each X_i for $1 \leq i \leq n$ are scalar-valued random variables on the same probability space.*

Definition 2.5.8 (Multivariate Normal Distribution)**.** *A random vector $\mathbf{X} = [X_1, \cdots, X_n]^T$ is said to follow the normal distribution written as $\mathbf{X} \sim \mathcal{N}(\boldsymbol{\mu}, \boldsymbol{\Sigma})$ if the probability density function is given by*

$$p(\mathbf{X}, \boldsymbol{\mu}, \boldsymbol{\Sigma}) = \frac{1}{(2\pi)^{n/2}|\boldsymbol{\Sigma}|^{\frac{1}{2}}} \exp\left[-\frac{1}{2}(\mathbf{X} - \boldsymbol{\mu})^T \boldsymbol{\Sigma}^T (\mathbf{X} - \boldsymbol{\mu})\right], \tag{2.57}$$

with mean vector $\boldsymbol{\mu} = E[\mathbf{X}] = [E[X_1], \cdots, E[X_n]]^T$ and $n \times n$ covariance matrix $\boldsymbol{\Sigma}$ such that $\boldsymbol{\Sigma}_{i,j} = E[(X_i - E[X_i])(X_j - E[X_j])]$ for $1 \leq i, j \leq n$.

Note that in the thesis, for ease of understanding and convenience we write $q(\mathbf{x}) = \mathcal{N}(\boldsymbol{\mu}, \boldsymbol{\Sigma})$ to mean that $q(\mathbf{x})$ is the probability density function of the normal distribution with mean $\boldsymbol{\mu}$ and covariance matrix $\boldsymbol{\Sigma}$. We state an elementary result below corresponding to the sum of two random variables following Gaussian distribution.

Lemma 2.5.9. *Let $\mathbf{X}$ and $\mathbf{Y}$ be two $d-$dimensional random variables such that $\mathbf{X} \sim \mathcal{N}(\boldsymbol{\mu}_X, \boldsymbol{\Sigma}_X)$ and $\mathbf{Y} \sim \mathcal{N}(\boldsymbol{\mu}_Y, \boldsymbol{\Sigma}_Y)$. Then $\mathbf{X} + \mathbf{Y}$ is also a random variable such that,*

$$\mathbf{X} + \mathbf{Y} \sim \mathcal{N}(\boldsymbol{\mu}_X + \boldsymbol{\mu}_Y, \boldsymbol{\Sigma}_X + \boldsymbol{\Sigma}_Y). \tag{2.58}$$

We state some important inequalities from statistical theory which will be used later in the proofs for the results derived in Chapters 3 and 5.

Definition 2.5.10 (Markov's Inequality)**.** *Let X be a non-negative random variable and $t > 0$ be a real number. Then,*

$$P(X \geq t) \leq \frac{\mathbb{E}[X]}{t}. \tag{2.59}$$

Definition 2.5.11 (Chebyshev's Inequality)**.** *Suppose X is a random variable with finite non-zero variance σ^2. Then for any real number $t > 0$,*

$$P(|X - \mathbb{E}[X] \geq t) \leq \frac{\sigma^2}{t^2}. \tag{2.60}$$

Definition 2.5.12 (McDiarmid's Inequality)**.** *Let $X_1, \cdots, X_n \in \mathcal{X}$ be independent random variables and $f : \mathcal{X}^n \to \mathbb{R}$ be a function. For all $i \in \{1, \cdots, n\}$, and $x_1, \cdots, x_n, x_i^{'} \in \mathcal{X}$, if the function ϕ satisfies*

$$|\phi(x_1, \cdots, x_{i-1}, x_i, x_{i+1}, \cdots, x_n) - \phi(x_1, \cdots, x_{i-1}, x_i^{'}, x_{i+1}, \cdots, x_n)| \leq c_i,$$

then for $t > 0$

$$P\left(\phi(X_1, \cdots, X_n) - \mathbb{E}[\phi] \geq t\right) \geq \exp\left(\frac{-2t^2}{\sum\limits_{i=1}^{n} c_i^2}\right). \tag{2.61}$$

Definition 2.5.13 (Markov Chain)**.** *A countable sequence of random variables $X_0, X_1, X_2, \cdots$ along with a countable set S (called the state space which is a countable set containing possible values of X_i) is a Markov Chain if*

$$P\{X_{n+1} = s_{n+1} | X_n = s_n, \cdots, X_0 = s_0\} = P\{X_n + 1 = s_{n+1} | X_n = s_n\},$$

where $P(X = j|Y = i)$ denotes the conditional probability that the random variable X takes value j given that the random variable Y takes value i.

The above property (often referred to as the Markov property) indicates that the conditional probability of X_{n+1} is only dependent on X_n and any additional knowledge of the past variables $\{X_j = s_j : j < n\}$ is irrelevant. The Markov property is one of the key concepts exploited in diffusion models. The essential idea is to systematically and slowly destroy the structure in a data distribution through an iterative forward diffusion process [132]. We then learn to reverse the diffusion process so that it restores the original data structure. We first discuss the forward and reverse processes of the diffusion model.

Definition 2.5.14 (Forward Process of Diffusion Model)**.** *Let $\mathbf{x}_0 \in \mathbb{R}^d$ be a $d-$ dimensional input from an unknown distribution with probability density function $q(\mathbf{x}_0)$. Given a variance schedule $0 < \beta_1 \leq \beta_2 \leq \cdots \leq \beta_K < 1$, we define the forward process as*

$$\mathbf{x}_t = \sqrt{1-\beta_t}\mathbf{x}_{t-1} + \sqrt{\beta_t}\boldsymbol{\epsilon}, \quad \textit{where } \boldsymbol{\epsilon} \sim \mathcal{N}(\mathbf{0}, \mathbf{I}_d). \tag{2.62}$$

Note that we use the notation $q(\mathbf{x}_t|\mathbf{x}_{t-1}) = \mathcal{N}(\mathbf{x}_t; \sqrt{1-\beta_t}\mathbf{x}_{t-1}, \beta_t \mathbf{I}_d)$ to mean that the probability density function of the the conditional distribution is given by

$$q(\mathbf{x}_t|\mathbf{x}_{t-1}) = \frac{1}{\sqrt{(2\pi\beta_t)^d}} \exp\left(-\frac{1}{2}\frac{(\mathbf{x}_t - \sqrt{1-\beta_t}\mathbf{x}_{t-1})^T(\mathbf{x}_t - \sqrt{1-\beta_t}\mathbf{x}_{t-1})}{\beta_t}\right).$$

The above process describes a Markov chain where only the output of the present is relevant for the next one. Any information prior to that is irrelevant. This property helps us to reparameterize the entire process such that the future output is dependent only on the initial input i.e., we do not need the predecessing $\mathbf{x}_t$'s to get the next one. Using the reparameterization trick, we can obtain $\mathbf{x}_t$ at any given time $t \in \{1, \dots, T\}$ from $\mathbf{x}_0$:

$$\begin{aligned}
\mathbf{x}_t &= \sqrt{\alpha_t}\ \mathbf{x}_{t-1} + \sqrt{1-\alpha_t}\ \boldsymbol{\epsilon}_{t-1} \\
&= \sqrt{\alpha_t\alpha_{t-1}}\ \mathbf{x}_{t-2} + \sqrt{1-\alpha_t\alpha_{t-1}}\ \widetilde{\boldsymbol{\epsilon}}_{t-2} \\
&\vdots \\
&= \sqrt{\prod_{i=1}^{t}\alpha_i}\ \mathbf{x}_0 + \sqrt{1-\prod_{i=1}^{t}\alpha_i}\ \widetilde{\boldsymbol{\epsilon}}_0 \\
&= \sqrt{\overline{\alpha}_t}\ \mathbf{x}_0 + \sqrt{1-\overline{\alpha}_t}\ \widetilde{\boldsymbol{\epsilon}}_0,
\end{aligned}$$

where $\widetilde{\boldsymbol{\epsilon}}_i \sim \mathcal{N}(\mathbf{0}, \mathbf{I}_d)$ for $i = 0, \dots, t-2$, $\alpha_t = 1 - \beta_t$ for $t = 1, \dots, T$ and $\overline{\alpha}_t = \prod_{i=1}^{t}\alpha_i$. Therefore, the conditional distribution $q(\mathbf{x}_{t+1}|\mathbf{x}_0)$ is

$$q(\mathbf{x}_{t+1}|\mathbf{x}_0) = \mathcal{N}(\mathbf{x}_{t+1}; \sqrt{\overline{\alpha}_{t+1}}\,\mathbf{x}_0, (1-\overline{\alpha}_{t+1})\mathbf{I}_d). \tag{2.63}$$

Lemma 2.5.15. *Let $\mathbf{x}_1, \cdots, \mathbf{x}_T$ be the vectors obtained from $\mathbf{x}_0$ by applying the forward process given in Definition 2.5.14. Then,*

$$q(\mathbf{x}_1, \dots \mathbf{x}_T|\mathbf{x}_0) = \prod_{t=1}^{T} q(\mathbf{x}_t|\mathbf{x}_{t-1}).$$

Proof. For $T = 2$, on the right-hand side, we have

$$q(\mathbf{x}_1|\mathbf{x}_0)q(\mathbf{x}_2|\mathbf{x}_1) = q(\mathbf{x}_1|\mathbf{x}_0)q(\mathbf{x}_2|\mathbf{x}_1, \mathbf{x}_0) = \frac{q(\mathbf{x}_1, \mathbf{x}_0)}{q(\mathbf{x}_0)}q(\mathbf{x}_2|\mathbf{x}_1, \mathbf{x}_0) = \frac{q(\mathbf{x}_0, \mathbf{x}_1, \mathbf{x}_2)}{q(\mathbf{x}_0)} = q(\mathbf{x}_1, \mathbf{x}_2|\mathbf{x}_0).$$

Note $\mathbf{x}_2$ is independent of $\mathbf{x}_0$ when $\mathbf{x}_1$ is given, hence $q(\mathbf{x}_2|\mathbf{x}_1) = q(\mathbf{x}_2|\mathbf{x}_1, \mathbf{x}_0)$.
For $T = n + 1$, we have

$$
\begin{aligned}
\prod_{t=1}^{n+1} q(\mathbf{x}_t|\mathbf{x}_{t-1}) &= \prod_{t=1}^{n} q(\mathbf{x}_t|\mathbf{x}_{t-1}) q(\mathbf{x}_{n+1}|\mathbf{x}_n) \\
&= q(\mathbf{x}_1, ..., \mathbf{x}_n|\mathbf{x}_0)\, q(\mathbf{x}_{n+1}|\mathbf{x}_n, ..., \mathbf{x}_0) \\
&= \frac{q(\mathbf{x}_0, ..., \mathbf{x}_n)}{q(\mathbf{x}_0)}\, q(\mathbf{x}_{n+1}|\mathbf{x}_n, ..., \mathbf{x}_0) \\
&= \frac{q(\mathbf{x}_0, ..., \mathbf{x}_n, \mathbf{x}_{n+1})}{q(\mathbf{x}_0)} \\
&= q(\mathbf{x}_1, ...\mathbf{x}_{n+1}|\mathbf{x}_0),
\end{aligned}
$$

where the second equality is obtained by using the induction hypothesis and the fact that $\mathbf{x}_{n+1}$ is independent of $\mathbf{x}_0, \mathbf{x}_1, \cdots, \mathbf{x}_{n-1}$ when $\mathbf{x}_n$ is given. The remaining equalities are obtained by using Bayes' rule. □

Once the forward process has destroyed the data structure, diffusion models aim to build a generative Markov chain that converts a simple known distribution (e.g. a Gaussian) into a target (data) distribution using a (reverse) diffusion process. Learning involves estimating small perturbations to the (forward) diffusion process. From Markovian theory, if β_t's are small, the reverse process is also Gaussian [51]. If the input distribution is known, then the reverse conditional distribution can be computed. We derive the conditional reverse mean and covariance in the following lemma.

Lemma 2.5.16. *Let $q(\mathbf{x}_0)$ denote the probability density function of an unknown distribution and $0 < \beta_1 \leq \beta_2 \leq \cdots \leq \beta_K < 1 < 1$ be a sequence of variances such that $q(\mathbf{x}_t|\mathbf{x}_{t-1}) = \mathcal{N}(\mathbf{x}_t; \sqrt{1-\beta_t}\mathbf{x}_{t-1}, \beta_t \mathbf{I}_d)$. Then for the reverse Markov chain conditioned on $\mathbf{x}_0$, denoted by $q(\mathbf{x}_{t-1}|\mathbf{x}_t, \mathbf{x}_0)$, the mean vector $\tilde{\boldsymbol{\mu}}_t \in \mathbb{R}^d$ and the covariance matrix $\tilde{\mathbf{\Sigma}}_t = \tilde{\beta}_t \mathbf{I}_d \in \mathbb{R}^{d\times d}$ are given by*

$$
\tilde{\boldsymbol{\mu}}_t = \frac{\sqrt{\alpha_t}(1-\bar{\alpha}_{t-1})}{1-\bar{\alpha}_t}\mathbf{x}_t + \frac{\sqrt{\bar{\alpha}_{t-1}}\beta_t}{1-\bar{\alpha}_t}\mathbf{x}_0 \text{ and } \tilde{\beta}_t = \frac{1-\bar{\alpha}_{t-1}}{1-\bar{\alpha}_t}\beta_t,
$$

where $\alpha_t = 1-\beta_t$ for $t = 1, \ldots, T$ and $\overline{\alpha}_t = \prod_{i=1}^{t} \alpha_i$.

Proof. In order to find the mean and covariance matrix of the reverse Markov chain (conditioned on $\mathbf{x}_0$), we use Bayes' rule to obtain

$$
q(\mathbf{x}_{t-1}|\mathbf{x}_t, \mathbf{x}_0) = q(\mathbf{x}_t|\mathbf{x}_{t-1}, \mathbf{x}_0)\, \frac{q(\mathbf{x}_{t-1}, \mathbf{x}_0)}{q(\mathbf{x}_t, \mathbf{x}_0)}\, \frac{q(\mathbf{x}_0)}{q(\mathbf{x}_0)} = q(\mathbf{x}_t|\mathbf{x}_{t-1}, \mathbf{x}_0)\, \frac{q(\mathbf{x}_{t-1}|\mathbf{x}_0)}{q(\mathbf{x}_t|\mathbf{x}_0)}. \tag{2.64}
$$

Since the conditional distributions for the forward process are all Gaussian distributions, we can write $q(\mathbf{x}_{t-1}|\mathbf{x}_t, \mathbf{x}_0)$ as

$$\frac{1}{\sqrt{(2\pi\beta_t)^d}} \exp\left(-\frac{1}{2}\frac{(\mathbf{x}_t - \sqrt{\alpha_t}\mathbf{x}_{t-1})^T(\mathbf{x}_t - \sqrt{\alpha_t}\mathbf{x}_{t-1})}{\beta_t}\right)$$
$$\frac{1}{\sqrt{(2\pi(1-\bar{\alpha}_{t-1}))^d}} \exp\left(-\frac{1}{2}\frac{(\mathbf{x}_{t-1} - \sqrt{\bar{\alpha}_{t-1}}\mathbf{x}_0)^T(\mathbf{x}_{t-1} - \sqrt{\bar{\alpha}_{t-1}}\mathbf{x}_0)}{1-\bar{\alpha}_{t-1}}\right)$$
$$\left(\sqrt{(2\pi(1-\bar{\alpha}_t))^d}\right) \exp\left(\frac{1}{2}\frac{(\mathbf{x}_t - \sqrt{\bar{\alpha}_t}\mathbf{x}_0)^T(\mathbf{x}_t - \sqrt{\bar{\alpha}_t}\mathbf{x}_0)}{1-\bar{\alpha}_t}\right)$$

$$= \frac{\sqrt{(1-\bar{\alpha}_t)^d}}{\sqrt{(2\pi\beta_t(1-\bar{\alpha}_{t-1}))^d}} \exp\left(-\frac{1}{2}\left[\frac{\mathbf{x}_t^T\mathbf{x}_t - 2\sqrt{\alpha_t}\mathbf{x}_t^T\mathbf{x}_{t-1} + \alpha_t\mathbf{x}_{t-1}^T\mathbf{x}_{t-1}}{\beta_t}\right]\right)$$
$$\exp\left(-\frac{1}{2}\left[\frac{\mathbf{x}_{t-1}^T\mathbf{x}_{t-1} - 2\sqrt{\bar{\alpha}_{t-1}}\mathbf{x}_0^T\mathbf{x}_{t-1} + \bar{\alpha}_{t-1}\mathbf{x}_0^T\mathbf{x}_0}{1-\bar{\alpha}_{t-1}}\right]\right)$$
$$\exp\left(\frac{1}{2}\left[\frac{\mathbf{x}_t^T\mathbf{x}_t - 2\sqrt{\bar{\alpha}_t}\mathbf{x}_t^T\mathbf{x}_0 + \bar{\alpha}_t\mathbf{x}_0^T\mathbf{x}_0}{1-\bar{\alpha}_t}\right]\right). \quad (2.65)$$

Simplifying Eq.(2.65), the terms inside the exponential function become,

$$-\frac{1}{2}\left[\mathbf{x}_{t-1}^T\mathbf{x}_{t-1}\left(\frac{\alpha_t}{\beta_t} + \frac{1}{1-\bar{\alpha}_{t-1}}\right) - 2\mathbf{x}_{t-1}^T\left(\frac{\sqrt{\alpha_t}}{\beta_t}\mathbf{x}_t + \frac{\sqrt{\bar{\alpha}_{t-1}}}{1-\bar{\alpha}_{t-1}}\mathbf{x}_0\right) + \mathbf{x}_t^T\mathbf{x}_t\left(\frac{1}{\beta_t} - \frac{1}{1-\bar{\alpha}_t}\right)\right.$$
$$\left. +2\frac{\sqrt{\bar{\alpha}_t}}{1-\bar{\alpha}_t}\mathbf{x}_t^T\mathbf{x}_0 + \mathbf{x}_0^T\mathbf{x}_0\left(\frac{\bar{\alpha}_{t-1}}{1-\bar{\alpha}_{t-1}} - \frac{\bar{\alpha}_t}{1-\bar{\alpha}_t}\right)\right]$$

$$= -\frac{1}{2}\left[\mathbf{x}_{t-1}^T\mathbf{x}_{t-1}\frac{1-\bar{\alpha}_t}{\beta_t(1-\bar{\alpha}_{t-1})} - 2\mathbf{x}_{t-1}^T\left(\frac{\sqrt{\alpha_t}}{\beta_t}\mathbf{x}_t + \frac{\sqrt{\bar{\alpha}_{t-1}}}{1-\bar{\alpha}_{t-1}}\mathbf{x}_0\right)\right.$$
$$\left. + \left(\mathbf{x}_t\sqrt{\frac{1}{\beta_t} - \frac{1}{1-\bar{\alpha}_t}} + \mathbf{x}_0\sqrt{\frac{\bar{\alpha}_{t-1}}{1-\bar{\alpha}_{t-1}} - \frac{\bar{\alpha}_t}{1-\bar{\alpha}_{t-1}}}\right)^T \left(\mathbf{x}_t\sqrt{\frac{1}{\beta_t} - \frac{1}{1-\bar{\alpha}_t}} + \mathbf{x}_0\sqrt{\frac{\bar{\alpha}_{t-1}}{1-\bar{\alpha}_{t-1}} - \frac{\bar{\alpha}_t}{1-\bar{\alpha}_{t-1}}}\right)\right]$$

$$= -\frac{1}{2}\left[\left(\mathbf{x}_{t-1}\sqrt{\frac{\alpha_t}{\beta_t} + \frac{1}{1-\bar{\alpha}_{t-1}}} - \left(\mathbf{x}_t\sqrt{\frac{1}{\beta_t} - \frac{1}{1-\bar{\alpha}_t}} + \mathbf{x}_0\sqrt{\frac{\bar{\alpha}_{t-1}}{1-\bar{\alpha}_{t-1}} - \frac{\bar{\alpha}_t}{1-\bar{\alpha}_{t-1}}}\right)\right)^T\right.$$
$$\left.\left(\mathbf{x}_{t-1}\sqrt{\frac{\alpha_t}{\beta_t} + \frac{1}{1-\bar{\alpha}_{t-1}}} - \left(\mathbf{x}_t\sqrt{\frac{1}{\beta_t} - \frac{1}{1-\bar{\alpha}_t}} + \mathbf{x}_0\sqrt{\frac{\bar{\alpha}_{t-1}}{1-\bar{\alpha}_{t-1}} - \frac{\bar{\alpha}_t}{1-\bar{\alpha}_{t-1}}}\right)\right)\right]$$

$$= -\frac{1}{2}\left[\left(\frac{\alpha_t}{\beta_t}+\frac{1}{1-\bar{\alpha}_{t-1}}\right)\left(\mathbf{x}_{t-1}-\left(\mathbf{x}_t\frac{\sqrt{\frac{1}{\beta_t}-\frac{1}{1-\bar{\alpha}_t}}}{\sqrt{\frac{\alpha_t}{\beta_t}+\frac{1}{1-\bar{\alpha}_{t-1}}}}+\mathbf{x}_0\frac{\sqrt{\frac{\bar{\alpha}_{t-1}}{1-\bar{\alpha}_{t-1}}-\frac{\bar{\alpha}_t}{1-\bar{\alpha}_{t-1}}}}{\sqrt{\frac{\alpha_t}{\beta_t}+\frac{1}{1-\bar{\alpha}_{t-1}}}}\right)\right)^T\right.$$
$$\left.\left(\mathbf{x}_{t-1}-\left(\mathbf{x}_t\frac{\sqrt{\frac{1}{\beta_t}-\frac{1}{1-\bar{\alpha}_t}}}{\sqrt{\frac{\alpha_t}{\beta_t}+\frac{1}{1-\bar{\alpha}_{t-1}}}}+\mathbf{x}_0\frac{\sqrt{\frac{\bar{\alpha}_{t-1}}{1-\bar{\alpha}_{t-1}}-\frac{\bar{\alpha}_t}{1-\bar{\alpha}_{t-1}}}}{\sqrt{\frac{\alpha_t}{\beta_t}+\frac{1}{1-\bar{\alpha}_{t-1}}}}\right)\right)\right]. \tag{2.66}$$

Thus combining Equations (2.65) and (2.66) we get,

$$q(\mathbf{x}_{t-1}|\mathbf{x}_t,\mathbf{x}_0) = \frac{\sqrt{(1-\overline{\alpha}_t)^d}}{\sqrt{(2\pi\beta_t(1-\overline{\alpha}_{t-1}))^d}}\exp\left(-\frac{1}{2}\frac{(\mathbf{x}_{t-1}-\tilde{\boldsymbol{\mu}}_t)^T(\mathbf{x}_{t-1}-\tilde{\boldsymbol{\mu}}_t)}{\tilde{\beta}_t}\right), \tag{2.67}$$

where

$$\tilde{\boldsymbol{\mu}}_t = \frac{\sqrt{\alpha_t}(1-\bar{\alpha}_{t-1})}{1-\bar{\alpha}_t}\mathbf{x}_t + \frac{\sqrt{\bar{\alpha}_{t-1}}\beta_t}{1-\bar{\alpha}_t}\mathbf{x}_0,$$

and

$$\tilde{\beta}_t\mathbf{I}_d = \frac{1}{\frac{\alpha_t}{\beta_t}+\frac{1}{1-\bar{\alpha}_{t-1}}}\mathbf{I}_d = \frac{1-\bar{\alpha}_{t-1}}{1-\bar{\alpha}_t}\beta_t\mathbf{I}_d.$$

□

Thus we see that the probability density function of the reverse distribution conditioned on $\mathbf{x}_0$ is also a Gaussian distribution with mean vector and covariance matrix given by $\tilde{\boldsymbol{\mu}}_t$ and $\tilde{\beta}_t\mathbf{I}_d$ respectively defined above. Since diffusion models deal with learning distributions in order to generate data, we define some commonly used terms from information theory that are used for learning distributions.

Definition 2.5.17 (Kullback–Leibler (KL) Divergence)**.** *Let $p(x)$ and $q(x)$ be two probability distributions. Then the KL Divergence denoted by $D_{KL}(q(x)||p(x))$ is defined as*

$$D_{KL}(q(x)||p(x)) = \mathbb{E}_{q(x)}\left[\log\left(\frac{q(x)}{p(x)}\right)\right].$$

Roughly speaking, KL Divergence is a measure of the information lost when $p(x)$ is approximated by $q(x)$.

Lemma 2.5.18 (Loss Function for Diffusion Models)**.** *Let $\mathbf{x}_0$ be data drawn from an unknown input distribution $q(\mathbf{x}_0)$. Suppose $\mathbf{x}_1,\cdots,\mathbf{x}_T$ are the degraded data obtained by applying the forward process given in Definition 2.5.14 and $p(\mathbf{x}_0,\cdots,\mathbf{x}_T)$ denote the reverse joint distribution. Then the cross entropy loss of approximating the true input distribution $q(\mathbf{x}_0)$ using $p(\mathbf{x}_0)$ (denoted by L_{CE}) satisfies the following inequality*

$$L_{CE} \leq \mathbb{E}_{q(\mathbf{x}_{0:T})}\left[\log\frac{q(\mathbf{x}_T|\mathbf{x}_0)}{p(\mathbf{x}_T)} + \sum_{t=2}^{T}\log\frac{q(\mathbf{x}_{t-1}|\mathbf{x}_t,\mathbf{x}_0)}{p(\mathbf{x}_{t-1}|\mathbf{x}_t)} - \log p(\mathbf{x}_0|\mathbf{x}_1)\right].$$

Proof.

$$
\begin{aligned}
L_{CE} &= -\mathbb{E}_{q(\mathbf{x}_0)}[\log p(\mathbf{x}_0)] = -\mathbb{E}_{q(\mathbf{x}_0)}\left[\log\left(\int p(\mathbf{x}_{0:T})d\mathbf{x}_{1:T}\right)\right] \\
&= -\mathbb{E}_{q(\mathbf{x}_0)}\left[\log\left(\int \frac{q(\mathbf{x}_{1:T}|\mathbf{x}_0)p(\mathbf{x}_{0:T})}{q(\mathbf{x}_{1:T}|\mathbf{x}_0)}d\mathbf{x}_{1:T}\right)\right] \\
&= -\mathbb{E}_{q(\mathbf{x}_0)}\left[\log\left(\mathbb{E}_{q(\mathbf{x}_{1:T}|\mathbf{x}_0)}\frac{p(\mathbf{x}_{0:T})}{q(\mathbf{x}_{1:T}|\mathbf{x}_0)}\right)\right] \\
&\leq -\mathbb{E}_{q(\mathbf{x}_0)}\mathbb{E}_{q(\mathbf{x}_{1:T}|\mathbf{x}_0)}\log\left(\frac{p(\mathbf{x}_{0:T})}{q(\mathbf{x}_{1:T}|\mathbf{x}_0)}\right) \quad \text{(Using, if } \phi \text{ is convex then } \phi(\mathbb{E}[X]) \leq \mathbb{E}[\phi(X)]) \\
&= -\int q(\mathbf{x}_0)q(\mathbf{x}_{1:T}|\mathbf{x}_0)\log\left(\frac{p(\mathbf{x}_{0:T})}{q(\mathbf{x}_{1:T}|\mathbf{x}_0)}\right)d\mathbf{x}_{1:T}d\mathbf{x}_0 \\
&= -\int q(\mathbf{x}_1,..,\mathbf{x}_T,\mathbf{x}_0)\log\left(\frac{p(\mathbf{x}_{0:T})}{q(\mathbf{x}_{1:T}|\mathbf{x}_0)}\right)d\mathbf{x}_{1:T}d\mathbf{x}_0 \\
&= -\mathbb{E}_{q(\mathbf{x}_{0:T})}\log\left(\frac{p(\mathbf{x}_{0:T})}{q(\mathbf{x}_{1:T}|\mathbf{x}_0)}\right) = \mathbb{E}_{q(\mathbf{x}_{0:T})}\log\left(\frac{q(\mathbf{x}_{1:T}|\mathbf{x}_0)}{p(\mathbf{x}_{0:T})}\right) = L_{VLB}.
\end{aligned}
$$

Now we need a computable form for L_{VLB}. Indeed,

$$
\begin{aligned}
L_{VLB} &= \mathbb{E}_{q(\mathbf{x}_{0:T})}\left[\log\left(\frac{q(\mathbf{x}_{1:T}|\mathbf{x}_0)}{p(\mathbf{x}_{0:T})}\right)\right] \\
&= \mathbb{E}_{q(\mathbf{x}_{0:T})}\left[\log\left(\frac{\prod_{t=1}^{T} q(\mathbf{x}_t|\mathbf{x}_{t-1})}{p(\mathbf{x}_T)\prod_{t=1}^{T} p(\mathbf{x}_{t-1}|\mathbf{x}_t)}\right)\right] \\
&= \mathbb{E}_{q(\mathbf{x}_{0:T})}\left[-\log p(\mathbf{x}_T) + \sum_{t=1}^{T}\log\frac{q(\mathbf{x}_t|\mathbf{x}_{t-1})}{p(\mathbf{x}_{t-1}|\mathbf{x}_t)}\right] \\
&= \mathbb{E}_{q(\mathbf{x}_{0:T})}\left[-\log p(\mathbf{x}_T) + \sum_{t=2}^{T}\log\frac{q(\mathbf{x}_t|\mathbf{x}_{t-1})}{p(\mathbf{x}_{t-1}|\mathbf{x}_t)} + \log\frac{q(\mathbf{x}_1|\mathbf{x}_0)}{p(\mathbf{x}_0|\mathbf{x}_1)}\right].
\end{aligned}
$$

Conditioning on $\mathbf{x}_0$, we get

$$
\begin{aligned}
L_{VLB} &= \mathbb{E}_{q(\mathbf{x}_{0:T})}\left[-\log p(\mathbf{x}_T) + \sum_{t=2}^{T} \log \frac{q(\mathbf{x}_t|\mathbf{x}_{t-1}, \mathbf{x}_0)}{p(\mathbf{x}_{t-1}|\mathbf{x}_t)} + \log \frac{q(\mathbf{x}_1|\mathbf{x}_0)}{p(\mathbf{x}_0|\mathbf{x}_1)}\right] \\
&= \mathbb{E}_{q(\mathbf{x}_{0:T})}\left[-\log p(\mathbf{x}_T) + \sum_{t=2}^{T} \log \frac{q(\mathbf{x}_{t-1}|\mathbf{x}_t, \mathbf{x}_0)}{p(\mathbf{x}_{t-1}|\mathbf{x}_t)} + \sum_{t=2}^{T} \log \frac{q(\mathbf{x}_t|\mathbf{x}_0)}{q(\mathbf{x}_{t-1}|\mathbf{x}_0)} + \log \frac{q(\mathbf{x}_1|\mathbf{x}_0)}{p(\mathbf{x}_0|\mathbf{x}_1)}\right] \\
&= \mathbb{E}_{q(\mathbf{x}_{0:T})}\left[\log \frac{q(\mathbf{x}_T|\mathbf{x}_0)}{p(\mathbf{x}_T)} + \sum_{t=2}^{T} \log \frac{q(\mathbf{x}_{t-1}|\mathbf{x}_t, \mathbf{x}_0)}{p(\mathbf{x}_{t-1}|\mathbf{x}_t)} - \log p(\mathbf{x}_0|\mathbf{x}_1)\right],
\end{aligned}
$$

where the second equality holds since $q(\mathbf{x}_t|\mathbf{x}_{t-1}, \mathbf{x}_0) = q(\mathbf{x}_{t-1}|\mathbf{x}_t, \mathbf{x}_0))\dfrac{q(\mathbf{x}_t|\mathbf{x}_0))}{q(\mathbf{x}_{t-1}|\mathbf{x}_0))}$. $\square$

Thus we see from the above result that the cross entropy loss of the distribution $p(\mathbf{x}_0)$ approximating $q(\mathbf{x}_0)$ is bounded above by

$$
\mathbb{E}_{q(\mathbf{x}_{0:T})}\left[\log \frac{q(\mathbf{x}_T|\mathbf{x}_0)}{p(\mathbf{x}_T)} + \sum_{t=2}^{T} \log \frac{q(\mathbf{x}_{t-1}|\mathbf{x}_t, \mathbf{x}_0)}{p(\mathbf{x}_{t-1}|\mathbf{x}_t)} - \log p(\mathbf{x}_0|\mathbf{x}_1)\right].
$$

Referring to Definition 2.5.17, we see that

$$
\mathbb{E}_{q(\mathbf{x}_{0:T})}\left[\log \frac{q(\mathbf{x}_T|\mathbf{x}_0)}{p(\mathbf{x}_T)}\right] = D_{KL}(q(\mathbf{x}_T|\mathbf{x}_0)\|p(\mathbf{x}_T)), \tag{2.68}
$$

$$
\mathbb{E}_{q(\mathbf{x}_{0:T})}\left[\log \frac{q(\mathbf{x}_{t-1}|\mathbf{x}_t, \mathbf{x}_0)}{p(\mathbf{x}_{t-1}|\mathbf{x}_t)}\right] = D_{KL}(q(\mathbf{x}_{t-1}|\mathbf{x}_t, \mathbf{x}_0)\|p(\mathbf{x}_{t-1}|\mathbf{x}_t)). \tag{2.69}
$$

Note that from the formulation of diffusion models, Eq. (2.68) is constant and hence often ignored when training a diffusion model. For Eq. (2.69), since it compares two Gaussian distributions, a closed form can be computed. We elaborate further about the closed form for Eq. (2.69) in Chapter 5. For the last term, $\mathbb{E}_{q(\mathbf{x}_{0:T})}[-\log p(\mathbf{x}_0|\mathbf{x}_1)]$, there are numerous ways to handle this term in practice. For example, the authors in [70] chose to model this term using a separate discrete decoder. The loss function derived in [70] can also be connected to the stochastic differential equations (SDEs). We discuss this connection further in Chapter 5. We end this chapter with a short discussion on Fréchet Inception Distance (FID), a popular technique used to measure the quality of samples generated in a generative model when the input distribution is unknown. We use the FID score in Chapter 5 to compare the quality of samples generated using different models.

Definition 2.5.19 (Fréchet Inception Distance (FID))**.** *For two probability distributions p and q over $\mathbb{R}^d$ with finite mean and variances, the FID is given by*

$$d_F(p, q) := \left(\inf_{\gamma \in \Gamma(p,q)} \int\limits_{\mathbb{R}^d \times \mathbb{R}^d} \|\mathbf{x} - \mathbf{y}\|^2 d\gamma(\mathbf{x}, \mathbf{y}) \right)^{\frac{1}{2}} \tag{2.70}$$

where where $\Gamma(p, q)$ is the set of all measures on $\mathbb{R}^d \times \mathbb{R}^d$ with marginals p and q on $\mathbf{x}$ and $\mathbf{y}$ respectively.

The exact form of the FID may be tractable for certain distributions. For example, let p and q be two univariate Gaussian distributions with mean μ_p and μ_q, and variance σ_p and σ_q respectively. Then the FID is given by,

$$d_F(p, q) = (\mu_p - \mu_q)^2 + (\sigma_p - \sigma_q)^2.$$

In practice for generative models like GANs, the FID is calculated using the Inception V3 model that is pre-trained on the Imagenet dataset. Both the real and generated images are passed through the Inception V3 model which helps extract the images' features. The mean and variance of the real and generated features are then calculated and compared. In general, the lower the FID scores, the better.

Chapter 3

Function Approximation using Hard-Ridge Random Feature Expansion

This chapter is based on the development of a fast algorithm for high-dimensional function approximation. The contents of this chapter are taken from the article [123], with modification. We propose a random feature model for approximating high-dimensional sparse additive functions called the Hard-Ridge Random Feature Expansion method (HARFE). This method utilizes a hard-thresholding pursuit-based algorithm applied to the sparse ridge regression (SRR) problem to approximate the coefficients with respect to the random feature matrix. The SRR formulation balances between obtaining sparse models that use fewer terms in their representation and ridge-based smoothing that tends to be robust to noise and outliers. In addition, we use a random sparse connectivity pattern in the random feature matrix to match the additive function assumption. We prove that the HARFE method is guaranteed to converge with a given error bound depending on the noise and the parameters of the sparse ridge regression model. In addition, we provide a risk bound on the learned model. Based on numerical results on synthetic data as well as on real datasets, the HARFE approach obtains lower (or comparable) error than other state-of-the-art algorithms. The chapter is organized as follows: the problem statement and algorithm are introduced in Section 3.1 followed by theoretical discussion in Section 3.2. Sections 3.3 and 3.4 give results for experiments on numerical and real datasets respectively.

3.1 Problem Statement and Algorithm

We are interested in approximating an unknown high dimensional function $f : \mathbb{R}^d \to \mathbb{C}, d \gg 1$, from a set of m samples $\{(\mathbf{x}_k, y_k)\}_{k=1}^{m}$ where the inputs $\mathbf{x}_k$ are drawn from an (unknown) probability measure $\mu(\mathbf{x})$ and the output data is (likely) corrupted by noise:

$$y_k = f(\mathbf{x}_k) + e_k, \qquad \text{for } k \in [m]. \tag{3.1}$$

We assume that the noise e_k is a Gaussian random variable or bounded by some constant E, that is, $|e_k| \le E \ \forall k \in [m]$. In addition, we assume that the target function f is an order-q additive function (see Definition 2.3.5) with $q \ll d$, $K \ll \binom{d}{q}$, and that we do not have prior knowledge of the terms g_j. Using the random feature method [66, 109], we approximate the target function f by:

$$f(\mathbf{x}) \approx f^{\#}(\mathbf{x}) = \mathbf{c}^T \phi(\mathbf{W}^T\mathbf{x}) = \sum_{j=1}^{N} c_j \phi(\langle \mathbf{x}, \boldsymbol{\omega}_j \rangle),$$

where $\mathbf{W} = [\boldsymbol{\omega}_{k,j}] \in \mathbb{R}^{d\times N}$ is a random weight matrix, $\boldsymbol{\omega}_j \in \mathbb{R}^d$ are the column vectors of the matrix $\mathbf{W}$, and $\mathbf{c} \in \mathbb{C}^N$ is the coefficient vector. The random weight matrix $\mathbf{W} \in \mathbb{R}^{d\times N}$ is fixed, while the coefficients $\mathbf{c} \in \mathbb{C}^N$ are trainable. The function $\phi : \mathbb{R}^d \times \mathbb{R}^d \to \mathbb{C}$ is the nonlinear activation function and can be chosen to be a trigonometric function, the sigmoid function, or the ReLU function. Unless otherwise stated, we use the sine activation function, i.e. $\phi(\cdot) = \sin(\cdot)$. This model is a two-layer neural network with the weights in the hidden layer being randomized but not trainable and thus the training problem relies on learning the coefficient vector $\mathbf{c}$. Theoretically, the random feature method has been shown to be comparable with shallow networks in terms of theoretical risk bounds [109–111, 120] where the population risk is defined as

$$R(f^{\#}) := ||f - f^{\#}||^2_{L^2(d\mu)} = \int_{\mathbb{R}^d} |f(\mathbf{x}) - f^{\#}(\mathbf{x})|^2 d\mu(\mathbf{x}), \tag{3.2}$$

which is also the L^2 squared error between the true function and its approximation. Suppose the entries of the random weight matrix $\mathbf{W}$ are i.i.d. random variables generated by the (one-dimensional) probability function $\rho(\boldsymbol{\omega})$, $\boldsymbol{\omega}_{k,j} \sim \rho(\boldsymbol{\omega})$. Let $\mathbf{A} \in \mathbb{C}^{m\times N}$ be the random feature matrix whose entries are defined as $a_{k,j} = \phi(\langle \mathbf{x}_k, \boldsymbol{\omega}_j \rangle)$. Then the general regression problem is to solve the following optimization problem:

$$\min_{\mathbf{c}\in\mathbb{C}^N} ||\mathbf{A}\mathbf{c} - \mathbf{y}||_2^2, \tag{3.3}$$

where $\mathbf{y} = [y_1, y_2, ..., y_m]^T$.

For sparse additive modeling, since f is an order-q function, each entry of the random feature matrix $\mathbf{A}$ should only depend on q entries of the input data $\mathbf{x}_k$. Therefore, we can instead generate a sparse random matrix $\mathbf{W}$ where each column of $\mathbf{W}$ has at most q nonzero entries following the probability function $\rho(\boldsymbol{\omega})$ [66]. One such way to generate $\mathbf{W}$ is to first generate N random vectors $\mathbf{v}^{(j)} = (v_1^{(j)}, \dots, v_q^{(j)})^T$ in $\mathbb{R}^q$ and use a random embedding that assigns $\mathbf{v}^{(j)}$ to $\boldsymbol{\omega}_j$, where $\boldsymbol{\omega}_j = (0, 0, \dots, 0, v_1^{(j)}, 0, \dots, v_2^{(j)}, 0, \dots, v_q^{(j)}, 0, \dots, 0)^T$. In particular, for each j, we select a subset of q indices from $[d]$ uniformly at random and then sample each nonzero entry using $\rho(\boldsymbol{\omega})$. The sparse random matrix $\mathbf{W}$ can also be obtained as $\mathbf{W} = \widetilde{\mathbf{W}} \odot \mathbf{M}$, where $\widetilde{\mathbf{W}} \in \mathbb{R}^{d\times N}$ is a dense matrix whose entries are sampled from $\rho(\boldsymbol{\omega})$, the mask $\mathbf{M} \in \mathbb{R}^{d\times N}$ is a sparse matrix whose non-zero entries are one and each column of $\mathbf{M}$ has q non-zero entries, and $\odot$ denotes the element-wise multiplication. Using this formulation, the general sparse regression problem becomes

$$\text{find } \mathbf{c} \in \mathbb{C}^N \text{ such that } ||\mathbf{Ac} - \mathbf{y}||_2 \leq \epsilon\sqrt{m} \quad \text{and} \quad \mathbf{c} \quad \text{is sparse,} \tag{3.4}$$

where ϵ is the parameter related to the noise level. The motivation for sparsity in $\mathbf{c}$ is due to the assumption that $K \ll \binom{d}{q}$, thus not all index subsets are needed.

In order to solve the sparse random feature regression problem, we propose a new greedy algorithm named hard-ridge random feature expansion (HARFE), which uses a hard thresholding pursuit (HTP) like algorithm to solve the random feature ridge regression problem. Specifically, we learn $\mathbf{c}$ from the following minimization problem:

$$\min_{\mathbf{c}\in\mathbb{C}^N} \|\mathbf{Ac} - \mathbf{y}\|_2^2 + m\lambda||\mathbf{c}||_2^2 \quad \text{such that } \mathbf{c} \text{ is } s\text{-sparse,} \tag{3.5}$$

where $\lambda > 0$ is the regularization parameter. Equation (3.5) can be rewritten as

$$\min_{\mathbf{c}\in\mathbb{C}^N} \|\mathbf{Bc} - \tilde{\mathbf{y}}\|_2^2 \quad \text{such that } \mathbf{c} \text{ is } s\text{-sparse,} \tag{3.6}$$

where $\mathbf{B} = \begin{bmatrix} \mathbf{A} \\ \sqrt{m\lambda}\mathbf{I}_N \end{bmatrix} \in \mathbb{C}^{(m+N)\times N}$ and $\tilde{\mathbf{y}} = \begin{bmatrix} \mathbf{y} \\ \mathbf{0} \end{bmatrix} \in \mathbb{C}^{m+N}$. To solve (3.6), we first start with an s-sparse vector $\mathbf{c}^0 \in \mathbb{C}^N$ (typically taken as $\mathbf{c}^0 = \mathbf{0}$) and update based on the HTP approach

$$\begin{aligned} S^{n+1} &= \{\text{indices of } s \text{ largest (in magnitude) entries of } \mathbf{c}^n + \mu\mathbf{B}^\star(\tilde{\mathbf{y}} - \mathbf{Bc}^n)\}, \\ \mathbf{c}^{n+1} &= \text{argmin}\{\|\tilde{\mathbf{y}} - \mathbf{Bc}\|_2^2, \quad \text{supp}(\mathbf{c}) \subseteq S^{n+1}\}, \end{aligned} \tag{3.7}$$

where $\mu > 0$ is the step-size and s is a user defined parameter. The idea is to solve for the coefficients using a much smaller number of model terms. The subset S given by the indices of the s largest entries of one gradient descent step applied on the vector $\mathbf{c}$ is a good candidate for the support set of $\mathbf{c}$. The HTP algorithm iterates between these two steps and leads to a stable and robust reconstruction of sparse vectors depending on the RIP constant. The Gram matrix $\mathbf{B}^\star\mathbf{B}$ is computed directly based on the ridge problem

$$\mathbf{c}^n + \mu\mathbf{B}^\star(\tilde{\mathbf{y}} - \mathbf{Bc}^n) = (1 - m\mu\lambda)\mathbf{c}^n + \mu\mathbf{A}^\star(\mathbf{y} - \mathbf{Ac}^n).$$

The approach is summarized in Algorithm 1. The relative residual at the iterate $\mathbf{c}^n$ is defined as

$$\text{Relative Residual} = \frac{||\mathbf{Ac}^n - \mathbf{y}||_2}{||\mathbf{y}||_2}.$$

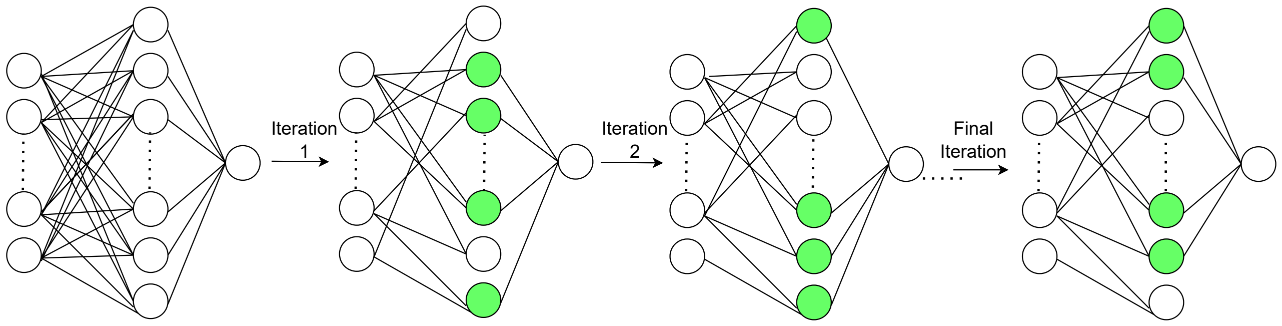


Figure 3.1: Schematic representation of HARFE. The active nodes at each iteration are given in green.

Algorithm 1 Hard-Ridge Random Feature Expansion (HARFE)

```
Input: Samples $\{\mathbf{x}_k, y_k\}_{k=1}^m$, non-linear function $\phi$, number of features $N$, sparsity level $s$,
    random weight sparsity $q$, step size $\mu$, regularization parameter $\lambda$, convergence thresh-
    old $\epsilon$ and total number of iterations tot_iter.
    Draw ($q$-sparse) $N$ random weights $\boldsymbol{\omega}_j$ whose non-zero entries are sampled from $\rho(\boldsymbol{\omega})$.
    Construct the random feature matrix $\mathbf{A} = [\phi(\langle \mathbf{x}_k; \boldsymbol{\omega}_j \rangle)] \in \mathbb{C}^{m \times N}$.
Algorithm:
    Initialization: Start with $s$-sparse $\mathbf{c}^0 \in \mathbb{C}^N$ ($\mathbf{c}^0 = \mathbf{0}$), $n = 0$
    while (Relative Residual> $\epsilon$) or (n<tot_iter) do
        $\tilde{\mathbf{c}}^{n+1} \leftarrow (1 - m\mu\lambda)\mathbf{c}^n + \mu \mathbf{A}^\star(\mathbf{y} - \mathbf{A}\mathbf{c}^n)$
        idx $\leftarrow$ indices of $s$ largest entries of $\tilde{\mathbf{c}}^{n+1}$    $\triangleright$ Choose the subset of features $S^{n+1}$
        $\bar{\mathbf{A}} \leftarrow \mathbf{A}[:, \text{idx}]$
        $\mathbf{c}^{n+1}[[N] \setminus \text{idx}] = 0$
        $\mathbf{c}^{n+1}[\text{idx}] \leftarrow \text{argmin}\{||\mathbf{y} - \bar{\mathbf{A}}\mathbf{c}||_2^2 + m\lambda||\mathbf{c}||_2^2\} = (\bar{\mathbf{A}}^\star\bar{\mathbf{A}} + m\lambda\mathbf{I}_s)^{-1}\bar{\mathbf{A}}^\star\mathbf{y}$
        $n = n + 1$
    end while
    Return: Sparse vector $\mathbf{c} = [c_1, c_2, ..., c_N]$ such that: $f^\sharp(\mathbf{x}) = \sum_{j=1}^{N} c_j \phi(\mathbf{x}; \boldsymbol{\omega}_j)$.
```

One of the motivations for including the ridge penalty is that for random feature regression, the sparsity level s can be large and thus the least squares step in (3.7) may be ill-conditioned. Numerically, we observed that even a small non-zero value of λ can be beneficial for ensuring convergence and good generalization. In Figure 3.1, we provide a schematic representation of the sequence generated by the algorithm, namely, over each step a sparse subset of nodes is obtained until a final configuration is achieved.

3.2 Theoretical Discussion

The error produced by the HARFE algorithm can be established by extending the results on the HTP algorithm to include the ridge regression term and by leveraging bounds on the restricted isometry constant for this type of random feature matrix. Let $\delta_s(\mathbf{A})$ denote the s-th restricted isometry constant of a matrix $\mathbf{A} \in \mathbb{C}^{m\times N}$ and $\kappa_{1,s}(\mathbf{x})$ denote the ℓ^1 distance to the best s-term approximation (see Definitions 2.1.3 and 2.1.7) which provides a measure for the compressibility of the vector $\mathbf{x}$ with respect to the ℓ^1 norm and is obtained by setting all but the s-largest in magnitude entries to zero.

The following result is restricted to the case when $q = d$ and $\mu = 1$. The $q \le d$ case follows from similar arguments [66]. From [53], the theorem below can be extended trivially for any step size μ by scaling the matrix $\mathbf{A}$ and the vector $\mathbf{y}$ with the ratio $\sqrt{\mu}$.

Theorem 3.2.1 (Convergence of the iterates of HARFE)**.** *Let the data $\{\mathbf{x}_k\}_{k\in[m]}$ be drawn from $\mathcal{N}(\mathbf{0},\gamma^2\mathbf{I}_d)$, the weights $\{\boldsymbol{\omega}_j\}_{j\in[N]}$ be drawn from $\mathcal{N}(\mathbf{0},\sigma^2\mathbf{I}_d)$, and the random feature matrix $\mathbf{A} \in \mathbb{C}^{m\times N}$ be defined component-wise by $a_{k,j} = \exp(i\langle \mathbf{x}_k, \boldsymbol{\omega}_j\rangle)$. Denote the ℓ_2-regularization parameter by λ and the sparsity level by s. If*

$$\begin{aligned} m &\ge C_1\,(1+\lambda)^{-2}\, s\,\log(\delta^{-1}),\\ \frac{m}{\log(3m)} &\ge C_2\,(1+\lambda)^{-1}\, s\,\log^2(6s)\log\left(\frac{N}{9\log(2m)}+3\right),\\ \frac{\sqrt{\delta}}{6\sqrt{3}}&\,(1+\lambda)\,(4\gamma^2\sigma^2+1)^{\frac{d}{4}} \ge N, \end{aligned}$$

where C_1 and C_2 are universal positive constants, then with probability at least $1-2\delta$, for all $\mathbf{c} \in \mathbb{C}^N$ and $\mathbf{e} \in \mathbb{C}^m$ with $\mathbf{y} = \mathbf{A}\mathbf{c} + \mathbf{e}$, the sequence $\mathbf{c}^n$ defined by the Algorithm 1 with $\mathbf{c}^0 = \mathbf{0}$, using $2s$ instead of s in the algorithm, satisfies

$$\|\mathbf{c}^n - \mathbf{c}\|_2 \le 2\beta^n\|\mathbf{c}\|_2 + \frac{D_1}{\sqrt{s}}\kappa_{1,s}(\mathbf{c}) + D_2\sqrt{\frac{m^{-1}\|\mathbf{y}-\mathbf{A}\mathbf{c}\|_2^2 + \lambda\|\mathbf{c}\|_2^2}{1+\lambda}},$$

for all $n \ge 0$ where the constants $\beta \in (0,1), D_1, D_2 > 0$ depend only on $\delta_{6s}(\mathbf{B})$. The matrix $\mathbf{B}$ is given by $\mathbf{B} = (m+m\lambda)^{-\frac{1}{2}}\begin{bmatrix}\mathbf{A}\\ \sqrt{m\lambda}I_N\end{bmatrix} \in \mathbb{C}^{(m+N)\times N}$.

Proof. The ridge regression problem:

$$\min_{\mathbf{c}\in\mathbb{R}^N} \|\mathbf{A}\mathbf{c}-\mathbf{y}\|_2^2 + m\lambda\|\mathbf{c}\|_2^2 \tag{3.8}$$

can be written in the form:

$$\min_{\mathbf{z}'\in\mathbb{R}^N} \|\mathbf{B}\mathbf{z}' - \tilde{\mathbf{y}}\|_2^2, \tag{3.9}$$

where $\mathbf{B} = (m + m\lambda)^{-\frac{1}{2}} \begin{bmatrix} \mathbf{A} \\ \sqrt{m\lambda} I_N \end{bmatrix} \in \mathbb{C}^{(m+N)\times N}$ and $\tilde{\mathbf{y}} = \begin{bmatrix} \mathbf{y} \\ \mathbf{0} \end{bmatrix} \in \mathbb{C}^{m+N}$. For equation (3.9), the error is defined as $\tilde{\mathbf{e}} = \tilde{\mathbf{y}} - \mathbf{B}\mathbf{z}'$ which is equivalent to

$$\tilde{\mathbf{e}} = \begin{bmatrix} \mathbf{e} \\ -\sqrt{m\lambda}\,\mathbf{c} \end{bmatrix} \in \mathbb{C}^{m+N}.$$

If $\mathbf{c}$ is the minimizer of (3.8) and $\mathbf{z}'$ is the minimizer of (3.9), then $\mathbf{z}' = (m + m\lambda)^{\frac{1}{2}}\,\mathbf{c}$. The scaling is needed for the matrix to satisfy a restricted isometry property.

To estimate the restricted isometry constant of $\mathbf{B}$, we first bound the restricted isometry constant of $m^{-\frac{1}{2}}\mathbf{A}$. By Theorem 2.3.11 and the assumptions, if

$$\begin{aligned} m &\geq 6C_1\eta_1^{-2}\, s\, \log(\delta^{-1}), \\ \frac{m}{\log(3m)} &\geq 6C_2\eta_2^{-2}\, s\, \log^2(6s)\log\left(\frac{N}{9\log(2m)} + 3\right), \\ \sqrt{\delta}\,\eta_1\,(4\gamma^2\sigma^2 + 1)^{\frac{d}{4}} &\geq N, \end{aligned}$$

where C_1 and C_2 are universal positive constants, then with probability at least $1 - 2\delta$, the $6s$-restricted isometry constant is bounded by

$$\delta_{6s}(\mathbf{A}) < 3\eta_1 + \eta_2^2 + \sqrt{2}\eta_2,$$

or equivalently

$$\left\| m^{-1}\mathbf{A}_S^\star \mathbf{A}_S - \mathbf{I}_S \right\|_{2\to 2} < 3\eta_1 + \eta_2^2 + \sqrt{2}\eta_2,$$

for all $S \subset [N]$ with $|S| = 6s$. Therefore, the $6s$-restricted isometry constant of B satisfies

$$\begin{aligned} \|\mathbf{B}_S^\star \mathbf{B}_S - \mathbf{I}_S\|_{2\to 2} &= \left\| (m + m\lambda)^{-1}\left(\mathbf{A}_S^\star\mathbf{A}_S + m\lambda\mathbf{I}_S\right) - \mathbf{I}_S \right\|_{2\to 2} \\ &= \frac{1}{1+\lambda}\left\| m^{-1}\mathbf{A}_S^\star\mathbf{A}_S - \mathbf{I}_S \right\|_{2\to 2} \\ &< \frac{3\eta_1 + \eta_2^2 + \sqrt{2}\eta_2}{1+\lambda}. \end{aligned}$$

Setting the parameters to $\eta_1 = \frac{1+\lambda}{6\sqrt{3}}$ and $\eta_2 = \frac{\sqrt{1+\lambda}}{4\sqrt{3}}$, then $\|\mathbf{B}_S^\star\mathbf{B}_S - \mathbf{I}_S\|_{2\to 2} < \frac{1}{\sqrt{3}}$ if

$$\begin{aligned} m &\geq 648C_1\,(1+\lambda)^{-2}\, s\, \log(\delta^{-1}), \\ \frac{m}{\log(3m)} &\geq 288C_2\,(1+\lambda)^{-1}\, s\, \log^2(6s)\log\left(\frac{N}{9\log(2m)} + 3\right), \\ \frac{\sqrt{\delta}}{6\sqrt{3}}\,(1+\lambda)\,(4\gamma^2\sigma^2+1)^{\frac{d}{4}} &\geq N. \end{aligned}$$

By Equation (2.10) of Theorem 2.1.18, the sequence generated by the hard thresholding pursuit algorithm produces a solution with the following bound

$$\|\mathbf{z}'^n - \mathbf{z}'\|_2 \leq 2\beta^n\|\mathbf{z}'\|_2 + \frac{C}{\sqrt{s}}\kappa_{1,s}(\mathbf{z}') + D\|\tilde{\mathbf{e}}\|_2, \tag{3.10}$$

for all $n \geq 0$ where the constants $\beta \in (0,1), C, D > 0$ depend only on $\delta_{6s}(\mathbf{B})$. Transforming the variables to the original variables yields the following bound

$$\begin{aligned}
\|\mathbf{c}^n - \mathbf{c}\|_2 &\leq 2\beta^n \|\mathbf{c}\|_2 + \frac{C}{\sqrt{s}}\kappa_{1,s}(\mathbf{c}) + D\ (1+\lambda)^{-\frac{1}{2}}\ m^{-\frac{1}{2}}\ \|\tilde{\mathbf{e}}\|_2 && (3.11)\\
&\leq 2\beta^n \|\mathbf{c}\|_2 + \frac{C}{\sqrt{s}}\kappa_{1,s}(\mathbf{c}) + D\ (1+\lambda)^{-\frac{1}{2}}\ m^{-\frac{1}{2}}\ \sqrt{\|\mathbf{e}\|_2^2 + m\lambda\|\mathbf{c}\|_2^2} && (3.12)\\
&\leq 2\beta^n \|\mathbf{c}\|_2 + \frac{C}{\sqrt{s}}\kappa_{1,s}(\mathbf{c}) + D\sqrt{\frac{m^{-1}\|\mathbf{e}\|_2^2 + \lambda\|\mathbf{c}\|_2^2}{1+\lambda}}, && (3.13)
\end{aligned}$$

which concludes the proof. □

It is worth noting that, in practice, $\lambda > 0$ is small, i.e. we would like $m\lambda = \mathcal{O}(1)$. Thus the third term in the iterative bound is smaller than the second term and does not contribute significantly to the overall error.

Theorem 3.2.2 (Risk bound for HARFE)**.** *Let the data $\{\mathbf{x}_k\}_{k\in[m]}$ be drawn from $\mathcal{N}(\mathbf{0}, \gamma^2\mathbf{I}_d)$, the weights $\{\boldsymbol{\omega}_j\}_{j\in[N]}$ be drawn from $\mathcal{N}(\mathbf{0}, \sigma^2\mathbf{I}_d)$, and the random feature matrix $\mathbf{A} \in \mathbb{C}^{m\times N}$ be defined component-wise by $a_{k,j} = \exp(i\langle \mathbf{x}_k, \boldsymbol{\omega}_j\rangle)$. Denote the ℓ_2 regularization parameter by λ, the accuracy parameter by $\epsilon > 0$, and the sparsity level by s. If the assumptions of Theorem 3.2.1 are satisfied, then with probability at least $1 - 3\delta$, the following risk bounds hold:*

$$R(f^{\#}) \leq \|f\|_\rho^2 \left(\epsilon^2 + D\frac{\lambda}{N}\left(3^{-\frac{1}{2}} + \left(2m\log\left(\frac{1}{\delta}\right)\right)^{\frac{1}{2}}\right)\right)$$
$$+ C\frac{1+\lambda}{s}\left(3^{-\frac{1}{2}} + N\left(2m\log\left(\frac{1}{\delta}\right)\right)^{\frac{1}{2}}\right)\kappa_{s,1}(\|\mathbf{c}^\star\|)^2 + DE^2\left(3^{-\frac{1}{2}}m^{-1} + N\left(\frac{2}{m}\log\left(\frac{1}{\delta}\right)\right)^{\frac{1}{2}}\right)$$

where,

$$\epsilon = \frac{1}{\sqrt{N}}\left(1 + \sqrt{2\log\left(\frac{1}{\delta}\right)}\right).$$

and $\mathbf{c}^\star$ denotes the coefficient vector corresponding to the best $\phi-$based approximation $f^\star$ defined in Eq. (3.18)*.*

Proof. We use the L^2 norm, which can be decomposed into two parts using the triangle inequality:

$$||f - f^\sharp||_{L^2(d\mu)} \leq ||f - f^\star||_{L^2(d\mu)} + ||f^\star - f^\sharp||_{L^2(d\mu)}.$$

The approximation $f^\sharp$ is defined in equation (3.17) and the best ϕ-based approximation $f^\star$ is given in equation (3.18).

Following the proof in Section 6 of [36], if $N \geq \frac{1}{\epsilon^2}\left(1+\sqrt{2\log\left(\frac{1}{\delta}\right)}\right)^2$, then with probability at least $1-\delta$, we have

$$||f - f^{\star}||_{L^2(d\mu)} \leq \epsilon ||f||_{\rho}. \tag{3.14}$$

We use McDiarmid's Inequality to bound $||f^{\star} - f^{\sharp}||_{L^2(d\mu)}$, arguing similarly as in Lemma 2 from [66] or Section 6 of [36]. Let $\{\mathbf{z}_j\}_{j\in[m]}$ be i.i.d. random variables sampled from the distribution μ which are independent from the sampled points $\{\mathbf{x}_j\}_{j\in[m]}$ and the frequencies $\{\boldsymbol{\omega}\}_{k\in[N]}$. This independence assumption makes $\{\mathbf{z}_j\}_{j\in[m]}$ also independent of coefficients $\mathbf{c}^{\sharp}$ and $\mathbf{c}^{\star}$ and thus allows for the following argument. Let v be a random variable defined by

$$v(\mathbf{z_1}, ..., \mathbf{z_m}) = \|f^{\star} - f^{\sharp}\|^2_{L^2(d\mu)} - \frac{1}{m}\sum_{j=1}^{m} |f^{\star}(\mathbf{z}_j) - f^{\sharp}(\mathbf{z}_j)|^2.$$

Then $\mathbb{E}_{\mathbf{z}}[v] = 0$ as

$$\mathbb{E}_{\mathbf{z}}[|f^{\star}(\mathbf{z}_j) - f^{\sharp}(\mathbf{z}_j)|^2] = \mathbb{E}_{\mathbf{z_1},...,\mathbf{z_m}}[|f^{\star}(\mathbf{z}_j) - f^{\sharp}(\mathbf{z}_j)|^2] = \|f^{\star} - f^{\sharp}\|^2_{L^2(d\mu)}.$$

We perturb the $k-$th component of v to get,

$$|v(\mathbf{z_1}, ..., \mathbf{z_k}, ..., \mathbf{z_m}) - v(\mathbf{z_1}, ..., \tilde{\mathbf{z}}_\mathbf{k}, ..., \mathbf{z_m})| \leq \frac{1}{m}\left| |f^{\star}(\mathbf{z}_k) - f^{\sharp}(\mathbf{z}_k)|^2 - |f^{\star}(\tilde{\mathbf{z}}_k) - f^{\sharp}(\tilde{\mathbf{z}}_k)|^2 \right|.$$

Using Cauchy-Schwarz inequality, for any $\mathbf{z}$, we have,

$$|f^{\star}(\tilde{\mathbf{z}}_k) - f^{\sharp}(\tilde{\mathbf{z}}_k)|^2 \leq N\|\tilde{\mathbf{c}}^{\star} - \tilde{\mathbf{c}}^{\sharp}\|_2^2,$$

which holds since $|\phi(\mathbf{z}, \boldsymbol{\omega})| = 1$. Hence,

$$|v(\mathbf{z_1}, ..., \mathbf{z_k}, ..., \mathbf{z_m}) - v(\mathbf{z_1}, ..., \tilde{\mathbf{z}}_\mathbf{k}, ..., \mathbf{z_m})| \leq \frac{2N}{m}\|\tilde{\mathbf{c}}^{\star} - \tilde{\mathbf{c}}^{\sharp}\|_2^2 := \Delta.$$

Therefore, we can apply McDiarmid's inequality to the random variable v, i.e. $P_{\mathbf{z}}(v - \mathbb{E}_{\mathbf{z}}[v] \geq t) \leq \exp(-\frac{2t^2}{m\Delta^2})$ where $t := \Delta\sqrt{\frac{m}{2}\log\left(\frac{1}{\delta}\right)}$. Following the results from Theorem 3.2.1, we have that $\delta_{6s}(\mathbf{B}) < \frac{1}{\sqrt{3}}$ (the matrix $\mathbf{B}$ is as obtained in equation (3.9)), then with s replaced by $2s$, with probability at least $1-3\delta$ (2δ for the coherence bound and δ from the 3.14), we have:

$$\|f^{\star} - f^{\sharp}\|^2_{L^2(d\mu)} \leq \frac{1}{m}\sum_{j=1}^{m} |f^{\star}(\mathbf{z}_j) - f^{\sharp}(\mathbf{z}_j)|^2 + N\left(\sqrt{\frac{2}{m}\log\left(\frac{1}{\delta}\right)}\right)\|\tilde{\mathbf{c}}^{\star} - \tilde{\mathbf{c}}^{\sharp}\|_2^2$$

$$= \frac{1}{m}\|\tilde{\mathbf{B}}(\tilde{\mathbf{c}}^\star - \tilde{\mathbf{c}}^\sharp)\|_2^2 + N\left(\sqrt{\frac{2}{m}\log\left(\frac{1}{\delta}\right)}\right)\|\tilde{\mathbf{c}}^\star - \tilde{\mathbf{c}}^\sharp\|_2^2$$
$$\leq \frac{1}{\sqrt{3}m}\|(\tilde{\mathbf{c}}^\star - \tilde{\mathbf{c}}^\sharp)\|_2^2 + N\left(\sqrt{\frac{2}{m}\log\left(\frac{1}{\delta}\right)}\right)\|\tilde{\mathbf{c}}^\star - \tilde{\mathbf{c}}^\sharp\|_2^2.$$

From Equation (2.11) of Theorem 2.1.18, $\|(\tilde{\mathbf{c}}^\star - \tilde{\mathbf{c}}^\sharp)\|_2 \leq \frac{C}{\sqrt{s}}\kappa_{s,1}(\tilde{\mathbf{c}}^\star) + D\|\tilde{\mathbf{e}}\|_2$, where $C, D > 0$ depend only on $\delta_{6s}(\mathbf{B})$. Since $\tilde{\mathbf{c}} = \sqrt{m + m\lambda}\mathbf{c}$, transforming to original variables, we have:

$$\|f^\star - f^\sharp\|_{L^2(d\mu)} \leq \left(\frac{1}{\sqrt{3}m} + N\left(\sqrt{\frac{2}{m}\log\left(\frac{1}{\delta}\right)}\right)\right)^{\frac{1}{2}}\left(\sqrt{m + m\lambda}\frac{C}{\sqrt{s}}\kappa_{s,1}(\|\mathbf{c}^\star\|) + D\sqrt{\|\mathbf{e}\|^2 + m\lambda\|\mathbf{c}^\star\|_2^2}\right). \tag{3.15}$$

Note $|\mathbf{c}_k^\star| = \frac{1}{N}\left|\frac{\alpha(\boldsymbol{\omega}_k)}{\rho(\omega)_k}\right| \leq \frac{1}{N}\|f\|_\rho$ and $\|\mathbf{e}\|_2 \leq E$. Thus, combining Equations (3.14) and (3.15) yields

$$\begin{aligned}
&\|f - f^\sharp\|_{L^2(d\mu)} \\
&\leq \epsilon\|f\|_\rho + \left(\frac{1}{\sqrt{3}m} + N\left(\sqrt{\frac{2}{m}\log\left(\frac{1}{\delta}\right)}\right)\right)^{\frac{1}{2}}\left(\sqrt{m + m\lambda}\frac{C}{\sqrt{s}}\kappa_{s,1}(\|\mathbf{c}^\star\|) + D\sqrt{E^2 + m\lambda N^{-1}\|f\|_\rho^2}\right) \\
&\leq \epsilon\|f\|_\rho + \left(3^{-\frac{1}{4}}m^{-\frac{1}{2}} + N^{\frac{1}{2}}\left(\frac{2}{m}\log\left(\frac{1}{\delta}\right)\right)^{\frac{1}{4}}\right)\left(\sqrt{m}\sqrt{1+\lambda}\frac{C}{\sqrt{s}}\kappa_{s,1}(\|\mathbf{c}^\star\|) + D(E + \sqrt{m\lambda}N^{-\frac{1}{2}}\|f\|_\rho\right) \\
&= \|f\|_\rho\left(\epsilon + D\sqrt{\frac{\lambda}{N}}\left(3^{-\frac{1}{4}} + \left(2m\log\left(\frac{1}{\delta}\right)\right)^{\frac{1}{4}}\right)\right) \\
&\quad + C\sqrt{\frac{1+\lambda}{s}}\left(3^{-\frac{1}{4}} + N^{\frac{1}{2}}\left(2m\log\left(\frac{1}{\delta}\right)\right)^{\frac{1}{4}}\right)\kappa_{s,1}(\|\mathbf{c}^\star\|) + DE\left(3^{-\frac{1}{4}}m^{-\frac{1}{2}} + N^{\frac{1}{2}}\left(\frac{2}{m}\log\left(\frac{1}{\delta}\right)\right)^{\frac{1}{4}}\right).
\end{aligned}$$

□

Theorem 3.2.1 highlights a theoretical purpose of $\lambda > 0$ in terms of convergence. The constants $D_1, D_2 > 0$ in Theorem 3.2.1 depend on $\delta_{6s}(\mathbf{B})$. As λ approaches zero, the value of $\delta_{6s}(\mathbf{B})$ approaches the larger value of $\delta_{6s}(\mathbf{A})$ and thus β increases, which can lead to slower convergence. As λ becomes large, the solution approaches zero and the error bounds in Theorem 3.2.1 become trivial. In practice, we found that a small non-zero value is useful for convergence and for mitigating the effects of noise and outliers.

Theorem 3.2.1 is stated for any vector $\mathbf{c}$. We can consider two potential vectors $\mathbf{c}$ depending on the scaling of N and m. First, if N is sufficiently large, then by the results of [36, 66] the matrix $\mathbf{A}$ will be well-conditioned with high probability (for any $\lambda > 0$) and thus there exists a $\mathbf{c}$ such that $\mathbf{e} = \mathbf{0}$. Therefore, for small $\lambda > 0$ the relative error is dominated by the compressibility of $\mathbf{c}$:

$$\frac{\|\mathbf{c}^n - \mathbf{c}\|_2}{\|\mathbf{c}\|_2} \leq 2\beta^n + \frac{C}{\sqrt{s}} \frac{\kappa_{1,s}(\mathbf{c})}{\|\mathbf{c}\|_2} + D\sqrt{\frac{\lambda}{1+\lambda}}. \tag{3.16}$$

In this setting, the HARFE algorithm can be seen as a pruning approach that generates a subnetwork with s connections that is an approximation to the full N-parameter network, see [54, 158].

Alternatively, we can consider the function approximation results found in [66,109–111]. Suppose we are given a probability density ρ used to sample the entries of the random weights $\boldsymbol{\omega} \in \mathbb{R}^d$ and a function $\phi : \mathbb{R}^d \times \mathbb{R}^d \to \mathbb{C}$. A function $f \in \mathcal{F}(\phi, \rho)$ if $f : \mathbb{R}^d \to \mathbb{C}$ has finite ρ-norm with respect to ϕ defined by

$$\mathcal{F}(\phi,\rho) = \left\{ f(\mathbf{x}) = \int_{\mathbb{R}^d} \alpha(\boldsymbol{\omega})\phi(\mathbf{x};\boldsymbol{\omega})\, d\boldsymbol{\omega} \;\middle|\; \|f\|_\rho := \sup_{\boldsymbol{\omega}} \left|\frac{\alpha(\boldsymbol{\omega})}{\rho(\boldsymbol{\omega})}\right| < \infty \right\},$$

where $\rho(\boldsymbol{\omega}) = \rho(\omega_1)\dots\rho(\omega_d)$. The random feature approximation of $f \in \mathcal{F}(\phi,\rho)$ is denoted by $f^\sharp$ and defined as

$$f^\sharp(\mathbf{x}) = \sum_{j=1}^N c_j^\sharp\, \phi(\mathbf{x}, \boldsymbol{\omega}_j), \tag{3.17}$$

where the weights $\{\boldsymbol{\omega}_j\}_{j\in[N]}$ are sampled i.i.d. from the density ρ. Following [109–111], the best ϕ-based approximation of $f \in \mathcal{F}(\phi,\rho)$ is given by $f^\star$

$$f^\star(\mathbf{x}) = \frac{1}{N}\sum_{j=1}^N \frac{\alpha(\boldsymbol{\omega}_j)}{\rho(\boldsymbol{\omega}_j)}\, \phi(\mathbf{x}, \boldsymbol{\omega}_j), \tag{3.18}$$

where the coefficients with respect to the random features are defined as $c_j^\star := \frac{\alpha(\boldsymbol{\omega}_j)}{N\rho(\boldsymbol{\omega}_j)}$ for all $j \in [N]$ and thus $\|\mathbf{c}^\star\|_2 \leq N^{-\frac{1}{2}}\,\|f\|_\rho$. For example, let ρ be the density associated with $\mathcal{N}(0,\sigma^2)$ and assume that the conditions of Theorem 3.2.1 hold. In this setting, it was shown in [66] that $\|\mathbf{e}^\star\|_\infty = \|A\mathbf{c}^\star - \mathbf{y}\|_\infty \leq \epsilon\|f\|_\rho$, where

$$\epsilon := \frac{1}{\sqrt{N}}\left(1 + 4\gamma\sigma d\sqrt{1 + \sqrt{\frac{12}{d}\log\frac{m}{\delta}}} + \sqrt{\frac{1}{2}\log\left(\frac{1}{\delta}\right)}\right).$$

Therefore, the bound in Theorem 3.2.1 becomes

$$\|\mathbf{c}^n - \mathbf{c}^\star\|_2 \leq \left(2\beta^n\, N^{-\frac{1}{2}} + \frac{C}{\sqrt{s}}\left(1 - sN^{-1}\right) + D\sqrt{\frac{\epsilon^2 + \lambda N^{-1}}{1+\lambda}}\right)\|f\|_\rho, \tag{3.19}$$

which scales like $N^{-\frac{1}{2}}$. Equation (3.19) could be refined, since in multiple places an ℓ^2 bound is replaced by an ℓ^∞ norm (noting that $\|f\|_\rho$ is essentially an infinity-like norm).

For the HARFE results in the following two sections, we observed that the value of μ (see Algorithm 1) does not have a significant impact on the generalization. Therefore, we set $\mu = 0.1$ for all experiments, which can be shown to produce a convergent sequence by extending the proofs found in [52]. In addition, we fix the sparsity ratio (s/N) to be 5% or 10%. Since the parameter λ depends on the dataset and the noise level, it should be tuned, for example, using cross-validation. In our experiments, we found that the optimal regularization parameter can range from 10^{-12} to 10^{-1} depending on the input data (since the data is not normalized). Thus, we optimize our results over a set of possible values for λ in order to obtain good generalization.

3.3 Numerical Results on Synthetic Data

In this section, we test Algorithm 1 for approximating sparse additive functions including the benchmark examples discussed in [15, 17, 66, 102, 107]. The experiments show that the HARFE model outperforms several existing methods in terms of testing errors.

The step-size parameter is set to $\mu = 0.1$ for all experiments. Other hyperparameters will be specified for each experiment. The relative test error is calculated as

$$\mathrm{Rel}(f, f^\sharp) = \sqrt{\frac{\sum\limits_{\mathbf{x}\in X_{\text{test}}} |f(\mathbf{x}) - f^\sharp(\mathbf{x})|^2}{\sum\limits_{\mathbf{x}\in X_{\text{test}}} |f(\mathbf{x})|^2}}, \tag{3.20}$$

and the mean-squared test error is defined as

$$\mathrm{MSE}(f, f^\sharp) = \frac{1}{|X_{\text{test}}|} \sum_{\mathbf{x}\in X_{\text{test}}} |f(\mathbf{x}) - f^\sharp(\mathbf{x})|^2, \tag{3.21}$$

where f is the target function and $f^\sharp$ is the trained function.

3.3.1 Low-Order Function Approximation

In the first example, we show the advantage of using a greedy approach over an ℓ^1 optimization problem and the benefit of the additional ridge term. The input data is sampled from a uniform distribution $\mathcal{U}[-1, 1]^d$ and the activation function is set to $\phi(\cdot) = \sin(\cdot)$. The nonzero entries of the random weights $\boldsymbol{\omega}_j$ are sampled from $\mathcal{N}(0, 1)$. We introduce a set of random bias terms $b_j \in \mathbb{R}$ for $j \in [N]$ so that the random feature matrix is now defined as $a_{k,j} = \sin(\langle \mathbf{x}_k, \boldsymbol{\omega}_j\rangle + b_j)$. The bias is sampled from $\mathcal{U}[0, 2\pi]$ to cover all phase angles. The number of random weights is $N = 10^4$, the sparsity level is $s = 500$, and the number of training and testing data are $m_{\text{train}} = 500$ and $m_{\text{test}} = 500$, respectively. In all experiments in this section, we set the maximal number of iterations for our method to 50.

	q	$\frac{1}{\sqrt{1+\|\mathbf{x}\|_2^2}}$	$\sqrt{1+\|\mathbf{x}\|_2^2}$	$\frac{x_1x_2}{1+x_3^6}$	$\sum_{i=1}^{d}\exp(-\lvert x_i\rvert)$
d		5	5	5	100
SRFE	1	3.30	1.29	102.1	1.20
HARFE, $\lambda = 0$	1	3.30	1.40	105.6	1.50
HARFE, $\lambda > 0$	1	3.20	1.00	100	1.10
SRFE	3	0.80	1.0	8.0	1.80
HARFE, $\lambda = 0$	3	1.00	0.25	3.20	2.70
HARFE, $\lambda > 0$	3	0.73	0.18	3.40	2.01
SRFE	5	0.56	1.10	9.24	2.04
HARFE, $\lambda = 0$	5	1.80	1.08	11.20	3.00
HARFE, $\lambda > 0$	5	0.57	1.00	7.70	2.20

Table 3.1: Relative test errors (as a percentage) for approximating various nonlinear functions using different q values. For each function, the two smallest errors are highlighted (in blue). The ridge paramater λ using in the HARFE approach is set to $10^{-4}, 10^{-10}, 10^{-10}$, and 10^{-1} (going left to right). In all experiments, $m_{train} = m_{test} = 500$, $N = 10^4$, $\mathbf{x} \sim \mathcal{U}[-1,1]^d$, the nonzero entries of $\boldsymbol{\omega}$ are drawn from $\mathcal{N}(0,1)$ and bias is drawn from $\mathcal{U}[0,2\pi]$.

Table 3.1 shows the median relative error (as a percentage) over 10 randomly generated test sets. In the experiments, we compared the results using $q \in \{1,3,5\}$. In each case, the HARFE approach with $\lambda > 0$ is more accurate than HARFE with $\lambda = 0$ and the SRFE [66]. We observe that when the exact order q is known, the error is lower (see Column 5 of Table 3.1). Although not included in the table, it is worth mentioning that the Elastic Net model performs comparably to the SRFE model, although it does not have the same generalization theory [66].

3.3.2 Approximation of Friedman Functions

In this example, we test the HARFE method on the Friedmann functions, which are used as benchmark examples for certain approximation techniques [15, 17, 102, 107]. The three Friedman functions are defined as, $f_1 : [0,1]^{10} \to \mathbb{R}$

$$f_1(x_1, ..., x_{10}) = 10\sin(\pi x_1 x_2) + 20\left(x_3 - \frac{1}{2}\right)^2 + 10x_4 + 5x_5,$$

$f_2 : [0,1]^4 \to \mathbb{R}$

$$f_2(x_1,x_2,x_3,x_4) = \sqrt{(100x_1)^2 + \left(x_3(520\pi x_2 + 40\pi) - \frac{1}{(520\pi x_2 + \pi)(10x_4+1)}\right)^2},$$

Method	f_1	f_2 ($\times 10^3$)	f_3 ($\times 10^{-3}$)
svm	4.36	18.13	23.15
lm	7.71	36.15	45.42
mnet	9.21	19.61	18.12
rForst	6.02	21.50	22.21
ANOVA	1.43	17.21	20.69
SRFE, $q = 2$	2.35	5.15	18.88
HARFE, $q = 2$	1.52	1.31	10.90
HARFE, q	3.01	1.90	13.28

Table 3.2: Mean-squared test errors of different methods when approximating Friedman functions. The values for SRFE and HARFE are obtained by training the model on 100 randomly generated training sets and validating them on 100 randomly generated test sets. We test for different q values. In the last row, $q = 5$ for f_1 and $q = d = 4$ for f_2 and f_3. For the HARFE, $\lambda = 1 \times 10^{-3}, 5 \times 10^{-3}$, and 1×10^{-5} for the functions f_1, f_2, and f_3, respectively. The two best values for every function are highlighted in blue.

and $f_3 : [0,1]^4 \to \mathbb{R}$

$$f_3(x_1, x_2, x_3, x_4) = \arctan\left(\frac{x_3(520\pi x_2 + 40\pi) - (520\pi x_2 + 40\pi)^{-1}(10x_4 + 1)^{-1}}{100x_1}\right).$$

We follow the setup from [107], where both the training and testing input datasets are randomly generated from the uniform distribution $\mathcal{U}[0,1]^d$, $m_{train} = 200$, and $m_{\text{test}} = 1000$. In addition, Gaussian noise with zero mean and standard deviations of $\sigma = 1.0, 125.0$, and 0.1, are added to the output data. The MSE of various methods when approximating Friedman functions are displayed in Table 3.2. We include the results of various methods found in [107] and compare them against the results of SRFE [66] and HARFE. For HARFE, the nonzero entries of the random weight vectors $\boldsymbol{\omega}$ and the bias terms are sampled from $\mathcal{U}[-1,1]$. We use $N = 10^4$ features for f_1 and $N = 2 \times 10^3$ features for f_2 and f_3, $s = 200$, and 50 iterations in total. Our proposed method achieves the smallest errors when approximating Friedman functions f_2 and f_3 even when the value of q is unknown (in that case, we assign $q = d$).

It is worth noting that although the first Friedman function has $d = 10$, it is a sparse additive model of order-2, and thus is better approximated by the ANOVA, SRFE, and HARFE models. This is verified by the results in Table 3.2 with $q = 2$. Note that HARFE with $q = 2$ yields a comparable result with ANOVA for f_1 while outperforming ANOVA in the other two examples. For the second Friedman function, HARFE with $q = 2$ is significantly better than the other methods by almost a factor of 14 times. For the third Friedman function, when the scale of the data is taken into consideration, all methods produce slightly worse results, with HARFE producing the lowest error overall. In Figure

3.2, scatter plots show the true data compared to the predicted values using the HARFE model over a test set for functions f_1, f_2, and f_3. The HARFE model produces lower variances for f_1 and f_2, while some bias occurs in f_3 near zero.

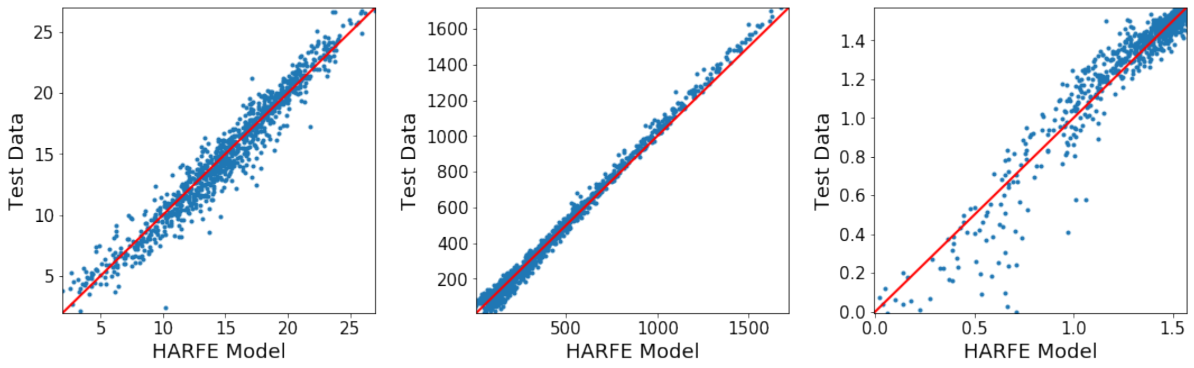


Figure 3.2: Scatter plots of the true data versus the predicted values using the HARFE model over the test set for functions f_1, f_2 and f_3 (from left to right).

In Figure 3.3, we plot the runtimes (in seconds) of both HARFE and SRFE during the training phase for $f(\mathbf{x}) = \sqrt{1 + \|\mathbf{x}\|_2^2}$ with different m and d. The stopping criteria used for both algorithms were the same. We can see clearly that HARFE is almost 2.5 times faster than SRFE as the number of features N increases.

In the next experiment, we would like to test HARFE for feature selection. Figure 3.4 displays a histogram illustrating the distribution of indices retained by the HARFE approach for the Friedman function $g : [0, 1]^{20} \to \mathbb{R}$,

$$g(x_1, ..., x_{20}) = 10\sin(\pi x_1 x_2) + 20\left(x_3 - \frac{1}{2}\right)^2 + 10x_4 + 5x_5.$$

In this experiment, we choose $q = 2$. From the histogram in Figure 3.4, we observe that HARFE can redistribute the weights based on active input variables especially when applied for a q-order additive function satisfying $q \ll d$ and the number of active variables (five in this case) is much less than the input dimension (which is twenty) of the function. Specifically, the histogram is based on the occurrence rate (as a percentage) of all twenty variables obtained from the HARFE model. The top 5 indices correspond exactly with the correct set.

3.4 Numerical Results on Real Datasets

We compare the performance of models obtained by the HARFE algorithm with other state-of-the-art sparse additive models [66,77,93] when applied to eleven real datasets. An overview of the datasets and the hyperparameters $s, q, m\lambda$ used in the HARFE model are presented in Table 3.3.

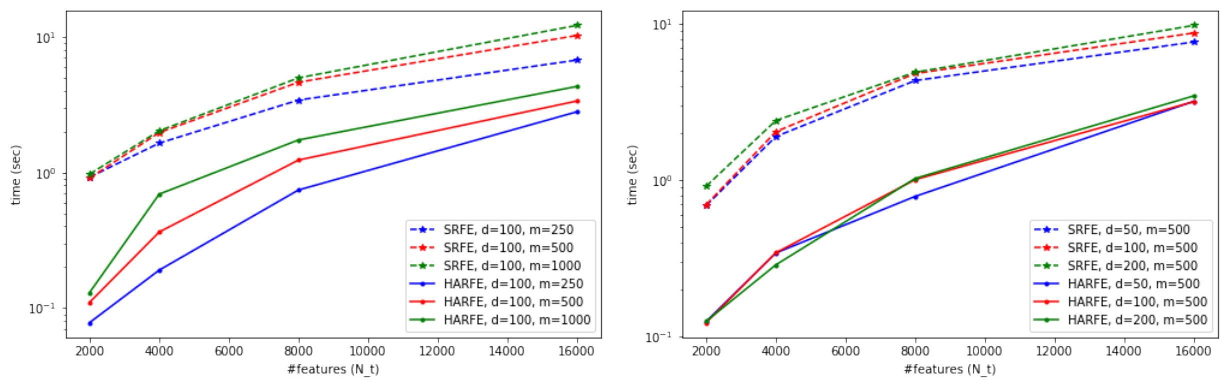


Figure 3.3: Plots showing the time required (in seconds) for optimizing the trainable weights $\mathbf{c}$ using HARFE compared with SRFE for $m \in \{250, 500, 1000\}$ with $d = 100$ (left) and $d \in \{50, 100, 200\}$ with $m = 500$ (right) for the function $f(\mathbf{x}) = \sqrt{1 + \|\mathbf{x}\|_2^2}$.

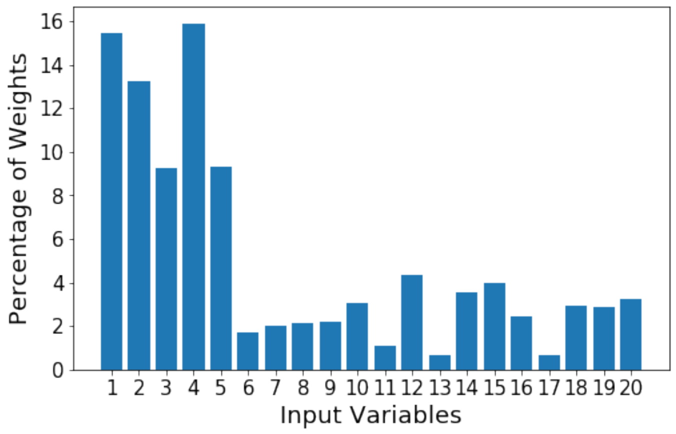


Figure 3.4: A histogram plot displaying the distribution of indices retained by the HARFE approach applied to the Friedman function $g(x_1, ..., x_{20}) = 10\sin(\pi x_1 x_2) + 20\left(x_3 - \frac{1}{2}\right)^2 + 10x_4 + 5x_5$ using $q = 2$. The histogram is based on the occurrence rate (as a percentage) of the input variables obtained from the HARFE model. The HARFE model correctly identifies the dominating index set.

The results of COSSO, Lasso, SALSA, SpAM, and SSAM are obtained from [77,93,113] and we include the results of the SRFE and HARFE model. The experiments follow the setup from [77,93,113], where the training data is normalized so that the input and output values have zero mean and unit variance along each dimension. Each dataset is divided in half to form the training and testing sets. The results and comparisons are shown in Table 3.4. The HARFE approach produces the lowest errors on eight datasets and achieves comparable results on the remaining datasets. Specifically, we significantly outperform other methods on the Propulsion, Airfoil, and Forestfire datasets.

In Figure 3.5, we plot the histogram of the percentage of weights corresponding to each

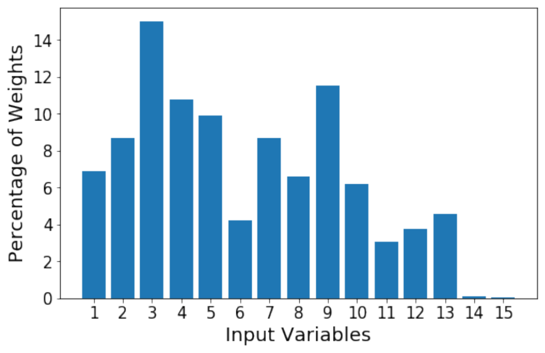
Dataset	dim	train	val	N	s	$m\lambda$	q
Propulsion	15	200	200	3k	300	10^{-10}	2
Galaxy	20	2000	2000	10k	1k	10^{-7}	2
Skillcraft	18	1700	1630	20k	1k	1.0	2
Airfoil	41	750	750	80k	5k	1.0	2
Forestfire	10	211	167	4220	422	0.5	2
Housing	12	256	250	10k	1k	0.1	2
Music	90	1000	1000	10k	666	2.5	3
Insulin	50	256	250	2560	170	2.75	2
Speech	21	520	520	20k	1k	0.1	2
Telemonitor	19	1000	867	15k	937	0.1	5
CCPP	59	2000	2000	10k	1k	0.05	1

Table 3.3: Overview of eleven datasets and the values of $s, m\lambda$, and q used in the HARFE model. The experimental setup and datasets for each test follow from [47, 77, 93, 113].

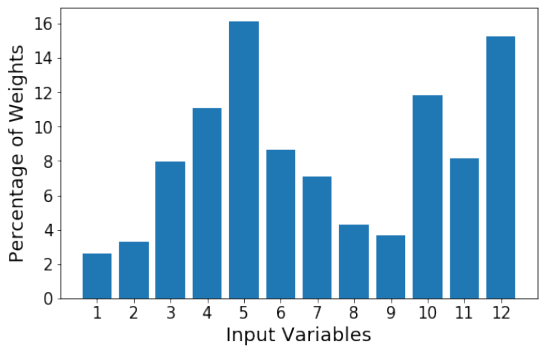
	HARFE	COSSO	Lasso	SALSA	SpAM	SRFE	SSAM
Propulsion	0.0000417	0.00094	0.0248	0.0088	1.1121	0.0154	-
Galaxy	0.0001024	0.00153	0.0239	0.00014	0.9542	0.0012	-
Skillcraft	0.5368	0.5551	0.6650	0.5470	0.9055	0.8730	0.5432
Airfoil	0.4492	0.5178	0.5199	0.5176	0.9623	0.5702	0.4866
Forestfire	0.2937	0.3753	0.5193	0.3530	0.9694	0.4067	0.3477
Housing	0.2636	1.3097	0.4452	0.2642	0.8165	0.6395	0.3787
Music	0.6134	0.7982	0.6349	0.6251	0.7683	1.0454	0.6295
Insulin	1.0137	1.1379	1.1103	1.0206	1.2035	1.6456	1.0146
Speech	0.0238	0.3486	0.0730	0.0224	0.6600	0.0246	-
Telemonitor	0.0370	5.7192	0.0863	0.0347	0.8643	0.0336	0.0689
CCPP	0.0677	0.9684	0.07395	0.0678	0.0647	0.07440	0.0694

Table 3.4: Average MSE on real datasets using various sparse additive models including COSSO, Lasso, SALSA, SpAM, SRFE, SSAM, and HARFE. The lowest error for each dataset is highlighted in blue.

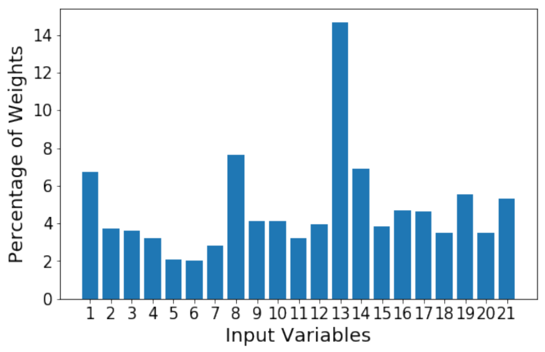
variable based on the (sparse) coefficient vector approximated using HARFE (from Table 3.4) for the Propulsion, Housing, Speech, and Telemonitor datasets, respectively. For the Propulsion test in Figure 3.5a, we see that the 14th variable (Gas turbine compressor decay state coefficient) and the 15th variable (Gas turbine decay state coefficient) have the least contribution to the predictor (Lever Position), while the 3rd variable (Gas turbine rate of revolutions) is the most relevant variable to the predictor. From the random sampling, the histograms are initiated uniformly (at least in expectation). This experiment shows that



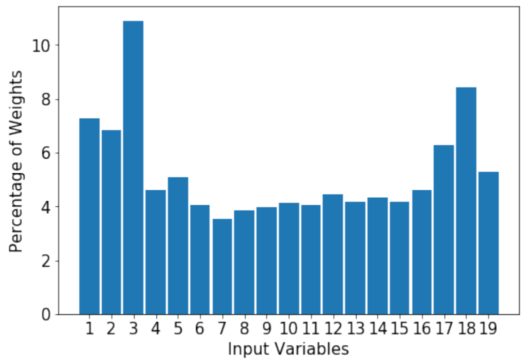
(a) Propulsion dataset



(b) Housing dataset



(c) Speech dataset



(d) Telemonitoring dataset

Figure 3.5: The histogram plots the percentage of weights corresponding to each variable based on the (sparse) coefficient vector approximated using HARFE for the Propulsion, Housing, Speech, and the Telemonitor datasets, respectively.

the HARFE algorithm will redistribute the weights and identify important variables as a benefit of the sparsity-promoting aspect.

From Figure 3.5b, the HARFE variable selection suggests that the predictor of the House dataset (per capita crime rate by town) is most affected by the 5th variable (proportion of owner-occupied units built prior to 1940) and the 12th variable (median value of owner-occupied homes in \$1000's). In Figure 3.5c, the plot shows that the 13th (Noise-to-Harmonic or NTH parameter) significantly contributes to the output of the predictor (median pitch) of the speech dataset. Lastly, for the Telemonitoring, the experiment in Figure 3.5d shows several significant contributors to HARFE trained predictor.

3.5 Summary

The proposed method HARFE aims to provide a fast algorithm for learning sparse additive functions. Theoretical results derived for HARFE indicate that the formulation of the algorithm using ridge regression with an HTP-based solution benefits the model in terms of both accuracy and convergence speed.

Chapter 4

Learning Epidemic Models from Missing Data

This chapter is based on predicting variables of epidemic models with missing data. The contents of this chapter are taken from the article [122], with slight modifications. The main motivation of this research work is to build a method that could be used to predict the future value of a particular variable belonging to a multidimensional dynamical system assuming that the availability of data for learning is only given for that particular variable. Such problems are frequently encountered in the field of epidemiology where one needs to quickly get short-term predictions of a particular variable (for example, the number of people infected) without the knowledge of any other variable or parameters. Not only is the data availability scarce, it is also incomplete. Continuing with our exploration of random feature models and sparsity, we now build a method for predicting the solution of an ODE from incomplete data. We make use of Takens' delay embedding theorem to embed the input data into a higher dimension space so that it represents a system diffeomorphic to the original one, which is then learned using random feature-based methods. We show that our proposed model outperforms existing benchmark methods of parameter learning in short-term predictions on simulated data as well as real datasets of various diseases like COVID-19, Zika, Ebola, etc. In this chapter, Section 4.1 gives the problem setting followed by the numerical results on synthetic and real data in Sections 4.2 and 4.3, respectively.

4.1 Motivation and Problem Setting

Given time-dependent observations $\{y(t_k)\}_{k=1}^{m}$ from an unknown multidimensional dynamical system, we propose a new inference method motivated by Takens' Theorem (stated in Theorem 2.4.9) where we aim to forecast the values of $y(t)$ over a given forecast horizon (generally one-week-ahead forecast). First proposed in 1981 in [138], given a dynamical system $\varphi : M \to M$, (M is a compact manifold of dimension d) and an observable $y : M \to \mathbb{R}$, Takens' theorem aims to obtain information about the original dynamical system. Motivated by Takens' theorem, we assume that the rate of change in the observable $y(t)$ is a function of its time-delayed mapping i.e.,

$$\dot{y}(t_k) = f(y(t_k), y(t_{k-1}), ..., y(t_{k-(p-1)})), \tag{4.1}$$

where p is the embedding dimension. Note that the variable y is part of an unknown multidimensional dynamical system. We would like to forecast the value of $y(t)$ while preserving the properties of the corresponding multidimensional dynamical system. Takens' theorem provides a theoretical guarantee to preserve the topological structure of the multidimensional dynamical system while solving the delayed equation, under certain conditions of the dynamics. Specifically, we wish to learn the function $f : \mathbb{R}^p \to \mathbb{R}$ in Equation (4.1) of the form

$$f(\mathbf{h}) \approx \sum_{j=1}^{N} c_j \phi(\langle \mathbf{h}, \boldsymbol{\omega}_j \rangle), \tag{4.2}$$

where $\boldsymbol{\omega}_j \in \mathbb{R}^p$ are the random weights, ϕ is a nonlinear activation function, and $\mathbf{c} = \left[c_1 \dots c_N\right]^T \in \mathbb{R}^N$ is the trainable coefficient vector. The nonlinear activation function $\phi : \mathbb{R}^p \times \mathbb{R}^p \to \mathbb{R}$ can be chosen to be a trigonometric function, the sigmoid function, or the ReLU function. Here, we use the ReLU activation function, i.e. $\phi(\cdot) = \max\{0, \cdot\}$. Entries of the random weight vector are i.i.d. random variables generated by a probability function $\rho(\boldsymbol{\omega})$, while the coefficients $\mathbf{c} \in \mathbb{R}^N$ are trainable. Also known as random features model proposed in [109–111], this model can be considered as a wide two-layer neural network where the weights in the first hidden layer are generated (following a distribution) and frozen while training problem relies on learning the coefficient vector $\mathbf{c}$. Since we assume limited data availability, we wish to have a sparse representation of the function f by learning the coefficient vector $\mathbf{c}$ with a sparsity constraint. Theoretically, when N is very large, the random feature methods have been shown to be comparable with shallow networks in terms of theoretical risk bounds [109–111, 120].

Given m measurements of the obervable y, we first build the input-output pairs $\{(\mathbf{h}_k, \dot{y}(t_k))\}_{k=p}^{m}$, where $\mathbf{h}_k = [y(t_k), y(t_{k-1}), ..., y(t_{k-p+1})]^T$ for $k = p, p+1, .., m$. We approximate the output data $\{\dot{y}(t_k)\}_{k=1}^{m}$ from given input data $\{y(t_k)\}_{k=1}^{m}$ using finite difference methods. In all our numerical simulations, we build the output data as follows:

$$\begin{aligned} \dot{y}(t_1) &= \frac{y(t_2) - y(t_1)}{(t_2 - t_1)}; \\ \dot{y}(t_k) &= \frac{y(t_{k+1}) - y(t_{k-1})}{(t_{k+1} - t_{k-1})}, \qquad \text{for } k = 2, .., m-1; \\ \dot{y}(t_m) &= \frac{y(t_m) - y(t_{m-1})}{(t_m - t_{m-1})}. \end{aligned} \tag{4.3}$$

We used central Euler discretizations for the intermediate points to have second-order accuracy and minimize the errors arising from using discrete methods. Other derivative approximation techniques could also be used, such as forward or backward discretizations. If the data is noisy, then an additional step would be required after getting the output data since finite difference methods may amplify the noise in the input dataset making recovery difficult. We apply a convolution-based averaging filter on $\{\dot{y}(t_k)\}_{k=1}^m$ with a smoothing parameter s as given below,

$$\dot{y}(t_k) = \frac{1}{s} \sum_{n=(k+1)-(s-1)}^{k+1} \dot{y}(t_n), \tag{4.4}$$

where $\dot{y}(t_k) = 0 \ \forall \ k \leq 0$ and $k \geq m+1$ and s denotes the strength of the smoothing filter. The value of s chosen is dependent upon the noise level present in the dataset. The value of s used for each of the experiments has been specified in their respective sections. Let $\mathbf{z} = [\dot{y}(t_p), \dot{y}(t_{p+1}), ..., \dot{y}(t_m)]^T$ and $\mathbf{A} = (\phi(\langle \mathbf{h}_k, \boldsymbol{\omega}_j \rangle)) \in \mathbb{R}^{(m-p+1)\times N}$. The matrix $\mathbf{A}$ is given by:

$$\mathbf{A} = \begin{bmatrix} \phi(\langle \mathbf{h}_p, \boldsymbol{\omega}_1 \rangle) & \phi(\langle \mathbf{h}_p, \boldsymbol{\omega}_2 \rangle) & \phi(\langle \mathbf{h}_p, \boldsymbol{\omega}_3 \rangle) & \dots & \phi(\langle \mathbf{h}_p, \boldsymbol{\omega}_N \rangle) \\ \phi(\langle \mathbf{h}_{p+1}, \boldsymbol{\omega}_1 \rangle) & \phi(\langle \mathbf{h}_{p+1}, \boldsymbol{\omega}_2 \rangle) & \phi(\langle \mathbf{h}_{p+1}, \boldsymbol{\omega}_3 \rangle) & \dots & \phi(\langle \mathbf{h}_{p+1}, \boldsymbol{\omega}_N \rangle) \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ \phi(\langle \mathbf{h}_m, \boldsymbol{\omega}_1 \rangle) & \phi(\langle \mathbf{h}_m, \boldsymbol{\omega}_2 \rangle) & \phi(\langle \mathbf{h}_m, \boldsymbol{\omega}_3 \rangle) & \dots & \phi(\langle \mathbf{h}_m, \boldsymbol{\omega}_N \rangle) \end{bmatrix} \in \mathbb{R}^{(m-p+1)\times N}. \tag{4.5}$$

The problem (4.2) becomes,

$$\text{find } \mathbf{c} \in \mathbb{R}^N \text{ such that } \mathbf{z} \approx A\mathbf{c} \quad \text{and} \quad \mathbf{c} \quad \text{is sparse}, \tag{4.6}$$

which can be solved by the following minimization problem:

$$\mathbf{c}^\# = \operatorname*{argmin}_{\mathbf{c}\in\mathbb{R}^N} \|\mathbf{A}\mathbf{c} - \mathbf{z}\|_2^2 + \lambda \|\mathbf{c}\|_1. \tag{4.7}$$

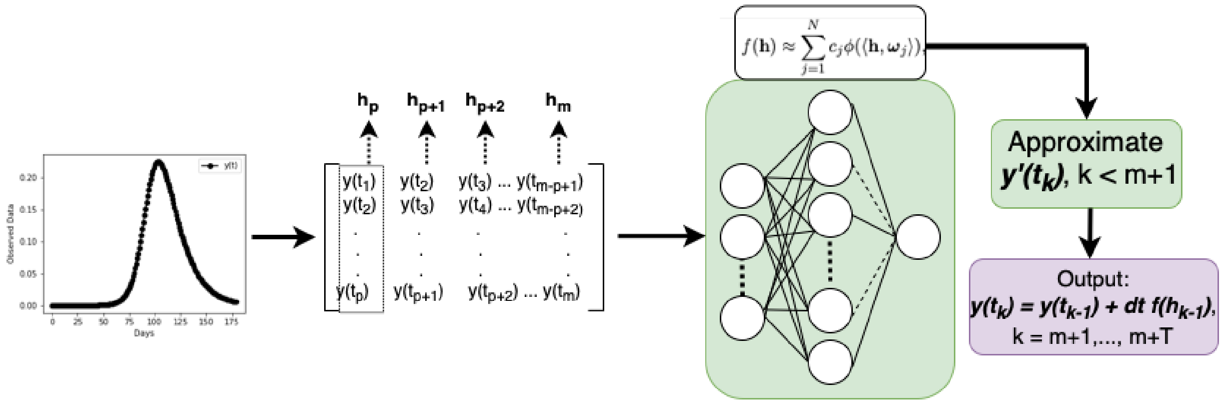
Here, $\lambda > 0$ is the regularization parameter. To forecast T future values of the given trajectory i.e., $\{y(t_{m+i})\}_{i=1}^T$, we use the Euler method:

$$y(t_{m+i}) = y(t_{m+i-1}) + (t_{m+i} - t_{m+i-1}) f(\mathbf{h}_{m+i-1}), \tag{4.8}$$

for all $i = 1, ..., T$ and $\mathbf{h}_k = [y(t_k), y(t_{k-1}), ..., y(t_{k-p+1}))]^T$ for any integer $k = m, m+1, ...m+T$. The summary of our SPADE4 algorithm is given in Algorithm 2 and schematically represented in Figure 4.1.

4.2 Numerical Experiments on Synthetic Data

In this section, we compare our proposed method and relevant benchmark methods, including the popular SEIR model and the most state-of-the-art SμEIR model [159] on synthetic data simulated from the SμEIR model. In this simulation study, the input data consists



$f(\mathbf{h}) \approx \sum_{j=1}^{N} c_j \phi(\langle \mathbf{h}, \boldsymbol{\omega}_j \rangle)$
h_p
h_{p+1}
h_{p+2}
h_m
y(t)
Observed Data
Days
$y(t_1)$ $y(t_2)$ $y(t_3)$ ... $y(t_{m-p+1})$
$y(t_2)$ $y(t_3)$ $y(t_4)$ ... $y(t_{m-p+2})$
$y(t_p)$ $y(t_{p+1})$ $y(t_{p+2})$ ... $y(t_m)$
Approximate $y'(t_k)$, k < m+1
Output: $y(t_k) = y(t_{k-1}) + dt\, f(h_{k-1})$, k = m+1,..., m+T

Figure 4.1: Schematic representation of SPADE4 algorithm

of time series $\{I(t_k)\}_{k=1}^{m}$ corresponding to the infectious variable $I(t)$ in the SμEIR model. Our goal is to obtain a one-week-ahead forecast horizon of this infectious variable. In all experiments, we display the predicted values of the mentioned methods versus the ground truth when varying the size of the training data as the observed trends change (before the peak, after the peak, and around the peak of the epidemic). We also study the robustness of SPADE4 with respect to noise and compare it with the benchmark models.

Data Generation. We first solve the SμEIR model given in Eq. (2.53) numerically on the time interval $[0, 180]$ (days) with the timestep $\Delta t = 0.01$ and the parameters of the SμEIR model:

$$\beta = 3/14, \quad \sigma = 0.25, \quad \mu = 0.75, \quad \gamma = 1/14, \quad P = S(0) + E(0) + I(0) + R(0),$$

and initial conditions are

$$S(0) = 10^6, \quad E(0) = 0, \quad I(0) = 1, \quad R(0) = 0.$$

We normalize the solution by the population P before building the training dataset. The training dataset is either noiseless $\{I(t_k)\}_{k=1}^{m}$, or noisy $\{I(t_k) + \varepsilon_k\}_{k=1}^{m}$, which resembles the reported daily counts of active cases in an outbreak. Note that the inputs are proportions instead of counts in this study. The size of the training data m will be specified in each experiment.

Prediction using SPADE4. For our proposed method, the random weights $\boldsymbol{\omega}_j \in \mathbb{R}^p$ are sampled from $\mathcal{N}(0, 1)$ and bias terms $b_j (j \in [N])$ are sampled from $\mathcal{U}(0, 2\pi)$. The number of random weights is $N = 50m$, where m is the number of training points. The activation function is set to $\phi(\cdot) = \text{ReLU}(\cdot)$ and the time delay $\tau = 1$. We choose the embedding dimension $p = 2d + 1$, where $d = 4$ is the dimension of the multi-dimensional dynamical system. The ℓ_1-regularization parameter λ is selected by the model from a range of possible values, $\lambda \in [10^{-6}, 5 \times 10^{-6}, 10^{-7}, 5 \times 10^{-7}, 10^{-8}, 5 \times 10^{-8}, 10^{-9}, 5 \times 10^{-9}]$ using the Bayesian Information Criterion (BIC) (see Definition 2.4.7). Using the training

Algorithm 2 Algorithm for SPADE4

```
Input: Input data $\{y(t_k)\}_{k=1}^m$; smoothing parameter $s$; function $\phi(\cdot,\boldsymbol{\omega}) = \phi(\langle\cdot,\boldsymbol{\omega}\rangle)$, distribution $\rho$ to sample $\boldsymbol{\omega}$, number of features $N$, regularization parameter $\lambda$.
Algorithm:
 1: Approximate $\{\dot{y}(t_k)\}_{k=1}^m$:
      $\dot{y}(t_1) \leftarrow \frac{y(t_2)-y(t_1)}{(t_2-t_1)}, \quad \dot{y}(t_m) \leftarrow \frac{y(t_m)-y(t_{m-1})}{(t_m-t_{m-1})},$
      $\dot{y}(t_k) \leftarrow \frac{y(t_{k+1})-y(t_{k-1})}{(t_{k+1}-t_{k-1})}, \quad \text{for} \quad k = 2,3,\dots,m-1.$
 2: if $\{y(t_k)\}_{k=1}^m$ is noisy then
      $\dot{y}(t_k) \leftarrow \frac{1}{s}\sum_{n=k+2-s}^{k+1} \dot{y}(t_n),$
    where $\dot{y}(t_k) = 0 \ \forall \ k \le 0$ and $k \ge m+1$.
 3: end if
 4: Define $\mathbf{z} = [\dot{y}(t_p), \dot{y}(t_{p+1}), ..., \dot{y}(t_m)]^T$.
 5: Draw $N$ random weights $\boldsymbol{\omega}_j \sim \rho(\boldsymbol{\omega})$.
 6: Define the time-delay data
      $\mathbf{h}_k = [y(t_k), y(t_{k-1}), ..., y(t_{k-p+1)})]^T, \quad \text{for} \quad k = p,\dots,m.$
 7: Construct random feature matrix $\mathbf{A} = (\phi(\langle \mathbf{h}_k, \boldsymbol{\omega}_j\rangle)) \in \mathbb{R}^{(m-p+1)\times N}$.
 8: Solve
      $\mathbf{c}^{\#} = \underset{\mathbf{c}\in\mathbb{R}^N}{\operatorname{argmin}} \|\mathbf{Ac}-\mathbf{z}\|_2^2 + \lambda\|\mathbf{c}\|_1.$
 9: for $i = 1,\dots,T$ do
10:     $\hat{f}(\mathbf{h}_{m+i-1}) = \sum_{j=1}^{N} c_j^{\#}\phi(\langle \mathbf{h}_{m+i-1}, \boldsymbol{\omega}_j\rangle)$.
11:     $y(t_{m+i}) = y(t_{m+i-1}) + (t_{m+i} - t_{m+i-1})\hat{f}(\mathbf{h}_{m+i-1})$.
12: end for
13: Output: $\{y(t_k)\}_{k=m+1}^{m+T}$.
```

data, we obtain the learned coefficients $\mathbf{c}$ by solving the optimization problem (4.7). In our paper, we choose the LASSO package in Python for obtaining $\mathbf{c}$. We forecast the proportion of daily active cases denoted by $\{\hat{I}_{ours}(t_k)\}_{k=m+1}^{m+T}$ (T denotes the prediction window) using Equation (4.8). Note that we may also choose to use the ridge-regression optimization problem and apply the proposed HARFE algorithm from Chapter 3. However, the performance of HARFE was not found to be superior to the proposed method as the underlying function may not satisfy the assumption of being a sparse high-dimensional additive function.

Prediction using benchmark models. For benchmark models, we use the least squares method to learn the models' parameters (from 100 possible random initializations) by fitting given input data with the variable $I(t)$ from SμEIR and SEIR models as the target function. The parameters learnt are $(\beta, \sigma, \mu, \gamma)$ for the SμEIR model (Equation (2.53)) and (β, σ, γ) for SEIR model (Equation (2.52)).

Prediction accuracy. For all the methods, the accuracy of predicting the trajectory of a novel pathogen outbreak (such as COVID-19) is measured using relative test error which is calculated as

$$\text{Error}(I, \hat{I}_{model}) = \sqrt{\frac{\sum\limits_{k=m+1}^{m+7} [I(t_k) - \hat{I}_{model}(t_k)]^2}{\sum\limits_{k=m+1}^{m+7} [I(t_k)]^2}}, \tag{4.9}$$

where I denotes the true values and $\hat{I}_{model}$ are the predicted values of a model.

4.2.1 Results on Noiseless Simulated Data

We first present numerical results on simulated data without noise where we evaluate model performance during key moments of the epidemic, i.e., before, around, and after the peak of the curve. For before and after the peak, we consider data points up to 81st and 125th days (given by the dashed vertical lines in the top left plot of Figure 4.2) out of 180 days, respectively, as the training set, and predict for the next seven days. The corresponding predicted curves of the one-week-ahead-forecast horizon are shown in the middle first row and the top right plot of Figure 4.2. For assessing model performance around the changing slope of the trajectory, we consider the number of training points $m \in \{97, 100, 104, 108, 111\}$ and plot the predicted curves of our proposed method and the two benchmark models for the one-week-ahead forecast in the second row of Figure 4.2. We observe from Figure 4.2 that the best performance is given by the benchmark model fitted with SμEIR (green curves), closely by our proposed method (blue curves) and the SEIR model (magenta curves). This is an expected outcome considering that we are fitting the benchmark model with the same model from which data was simulated. However, in real-world applications, there are irregularities in data which can make the noiseless case too idealistic. So in the following section, we add noise to the input data and then compare the performance of benchmark models to our SPADE4.

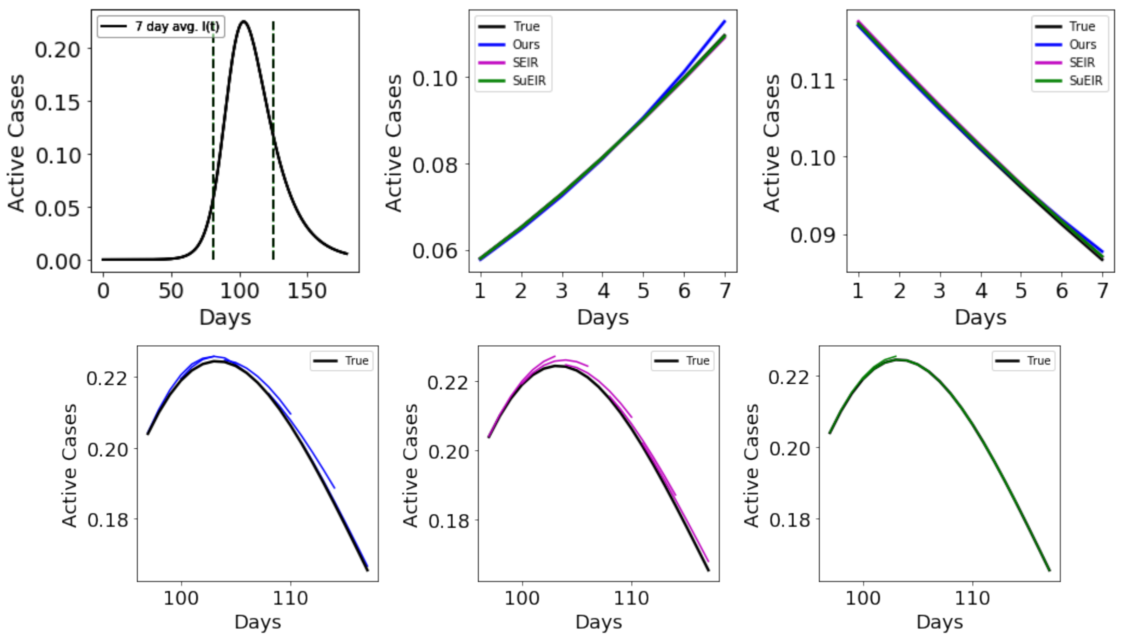


Figure 4.2: Results on noiseless simulated data. First row: Noiseless training datasets from 0 up to 81st and 125th days out of 180 days (top left). The next two figures are the predicted values of the infectious variable $I(t)$ in the next seven days using SPADE4 (blue), SEIR model (magenta), and SμEIR model (green) correspond to those two training datasets versus ground truth (black). Second row: Prediction of $I(t)$ for the next seven days around the peak of the wave using SPADE4 (blue), SEIR model (magenta), and SμEIR model (green) versus ground truth (black).

4.2.2 Results on Noisy Simulated Data

In this section, we consider noisy input data. As before, the data is simulated from Equation (2.53) with the parameters described above. The input data with noise is given by:

$$I_{\text{noisy}}(t) = I(t) + \varepsilon \max \mid I(t) \mid, \tag{4.10}$$

where $I(t)$ is the clean data, $\varepsilon \sim \mathcal{N}(0, \eta)$, and η is the noise level. For all experiments involving noisy input data, we pre-process the input data by considering the seven-day average of $I(t)$. This is done to take care of data irregularities and to resemble setup in practice. For our SPADE4, an additional convolution based smoothing filter with parameter $s = 15$ is also used on the output vector $\{\dot{I}(t_k)\}_{k=1}^{m}$. This is done to minimize noise amplification from the use of finite difference approximations to obtain $\{\dot{I}(t_k)\}_{k=1}^{m}$. Similar to the setup in Section 4.2.1, we consider the number of training points $m \in \{81, 97, 100, 104, 108, 111, 125\}$ days and plot the one-week-ahead forecast. We consider 5% noise level, i.e., $\eta = 0.05$, and plot the results in Figure 4.3. Experimental results with $\eta = 0.02$ are given in the appendix.

Before the peak, the one-week-ahead forecast of our SPADE4 outperforms the SEIR

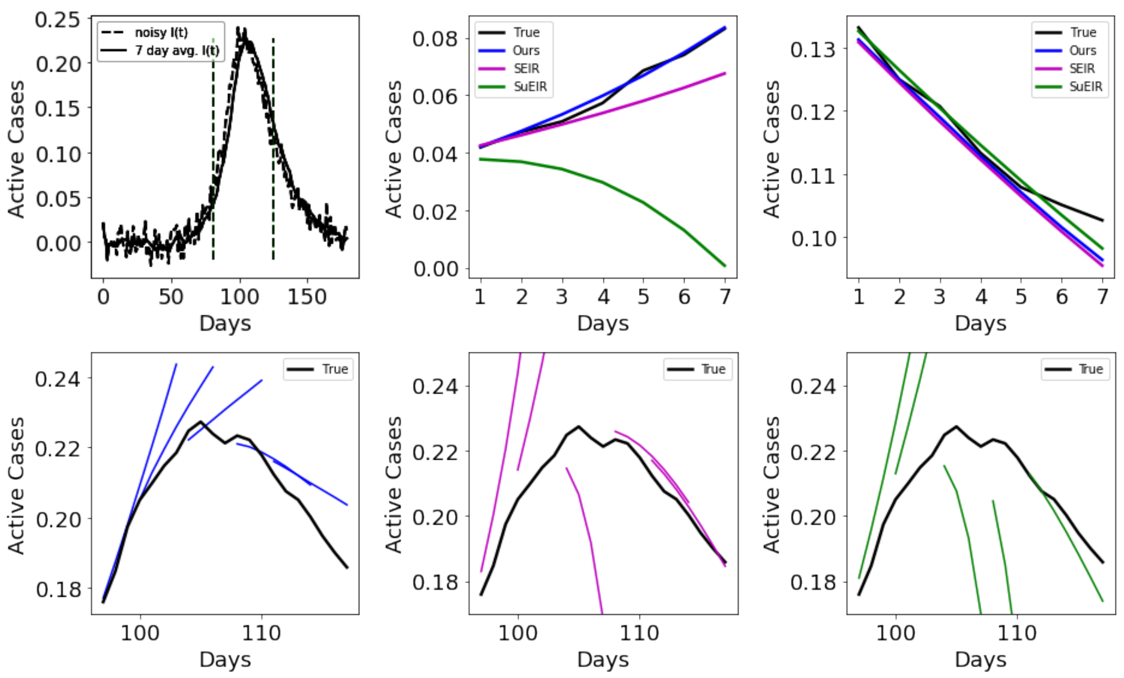


Figure 4.3: Results on simulated data with 5% noise added to input data. First row: Noisy training datasets from 0 up to 81^{st} and 125^{th} days out of 180 days (top left). The next two figures are the predicted values of the infectious variable $I(t)$ in the next seven days using SPADE4 (blue), SEIR model (magenta), and SμEIR model (green) correspond to those two training datasets versus ground truth (black). Second row: Prediction of $I(t)$ for the next seven days around the peak of the wave using SPADE4 (blue), SEIR model (magenta), and SμEIR model (green) versus ground truth (black).

and the SμEIR model in both cases of the noise level. After the peak, the prediction results of SPADE4 and the SEIR model are comparable and are better than those obtained from the SμEIR (see top row of Figure 4.3). Around the peak, (see bottom row of Figure 4.3), we observe that SPADE4 quickly picks up the changing slope in the wave. On the other hand, the benchmark models give unreliable results at some points in the curve. Both benchmark models overpredict the number of active cases for $m \in \{97, 100\}$ and underpredict the number of active cases for $m = 104$.

The superior performance of SPADE4 can be seen more clearly from Figure 4.4 where we plot the validation errors for our model and the benchmark models fitted different sizes of training data m and noise levels $\eta = 0.02$ and $\eta = 0.05$. We see that SPADE4 has low relative error consistently while the errors for the other models tend to fluctuate to a higher error at certain points. The performance of the benchmark models is highly influenced by the number of training points, noise level, as well as the nature of the curve while our proposed SPADE4 is robust to all these factors affecting input data.

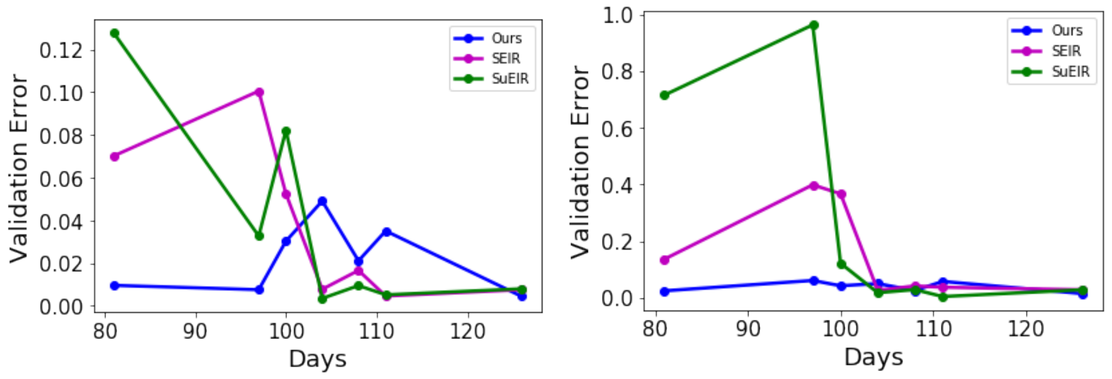


Figure 4.4: Validation errors based on various size of the training data $m \in \{81, 97, 101, 104, 108, 112, 126\}$ for the noise level $\eta = 0.02$ (left) and $\eta = 0.05$ (right).

4.3 Numerical Experiments on Real Datasets

In this section, we demonstrate the performance of SPADE4 on various real datasets, including daily active cases of 2019 Coronavirus (COVID-19) in Canada[1] as well as cumulative cases of Ebola in Guinea [2], Zika virus in Giradot [117], and influenza A/H7N9 in China [3]. The training data for the benchmark models is given by $I(t)$ for the COVID-19 dataset and by the variable $I(t) + R(t)$ for the remaining datasets. For each dataset, we compare the one-week-ahead forecasts of SPADE4 with benchmark methods including SEIR and SμEIR models across important moments in the epidemic. Predictions using SPADE4 use the same hyperparameters as described in Section 4.2 unless specified otherwise.

4.3.1 Data from Daily Active Cases of COVID-19 in Canada

In this section, we study the one-week-ahead forecast of the number of active COVID-19 cases in Canada (see Figure 4.5) up to the fifth wave of COVID-19. Each wave is approximately given by the black vertical lines in Figure 4.5. We notice that the first four waves are similar in terms of wavelength as well as amplitude. On the other hand, the fifth wave however is different since it is driven by a much more infectious Omicron variant and hence has a short wavelength with a higher amplitude. Therefore, we examine the forecast within the second (from day 200 to day 380) and the fifth waves (from day 650 to day 704) of COVID-19 in Canada.

The anomalies in how data were reported, such as the under-reporting of cases on weekends and the backlog cases reported later, can lead to a noisy dataset. Hence, we

[1]https://health-infobase.canada.ca/covid-19/epidemiological-summary-covid-19-cases.html
[2]https://www.kaggle.com/datasets/imdevskp/ebola-outbreak-20142016-complete-dataset
[3]https://datadryad.org/stash/dataset/doi:10.5061/dryad.2g43n

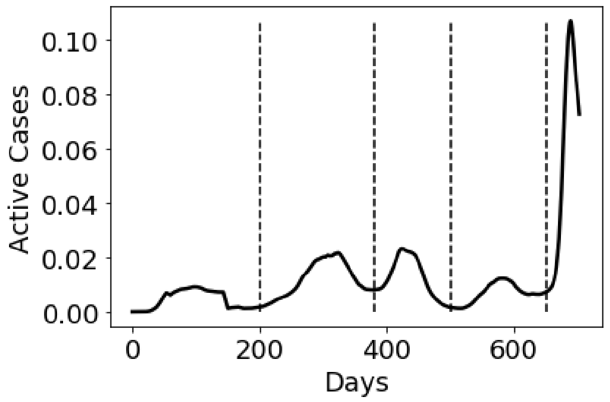


Figure 4.5: Proportion of daily active COVID-19 cases in Canada from the beginning of the pandemic to the end of the fifth wave.

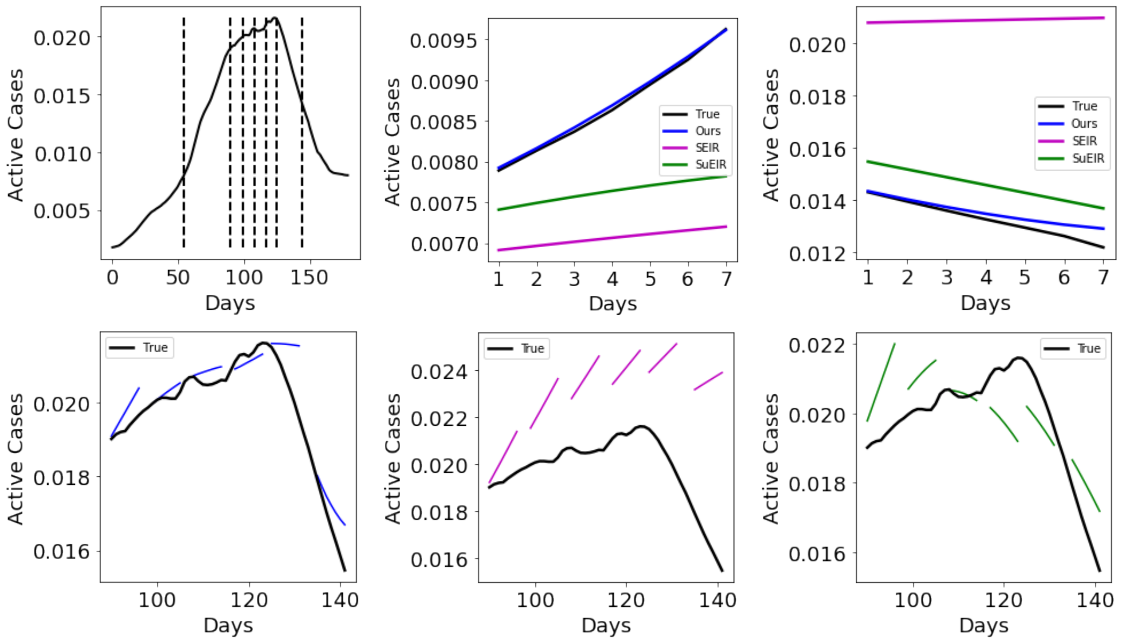


Figure 4.6: Results on second wave of COVID-19 in Canada. First row: Second wave from 0 up to 54^{th} and 144^{th} days out of 180 days (top left). The next two figures are the predicted values of the infectious variable $I(t)$ in the next seven days using SPADE4 (blue), SEIR model (magenta), and SμEIR model (green) correspond to those two training datasets versus ground truth (black). Second row: Prediction of $I(t)$ for the next seven days around the peak of the wave using SPADE4 (blue), SEIR model (magenta), and SμEIR model (green) versus ground truth (black).

consider the seven-day average of the given data as the ground truth. For the initial conditions required by the benchmark models, we choose the initial values of the variables $I(t_0) = I_0$, $R(t_0) = 0$, where I_0 is the first data point in the training set. Since the initial

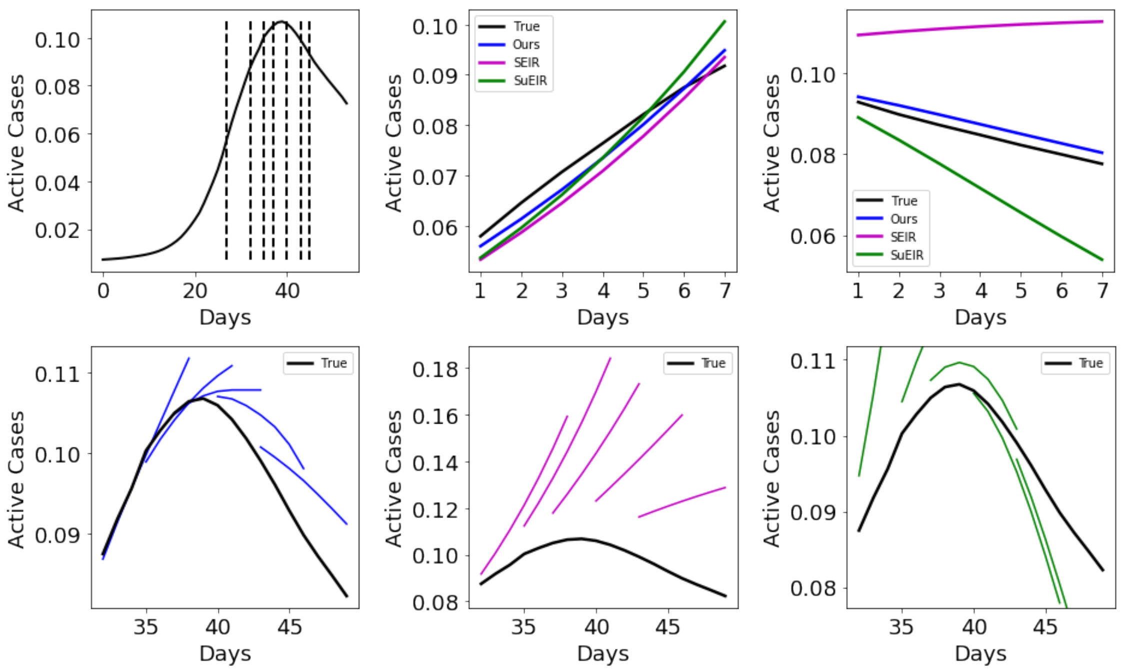


Figure 4.7: Results on fifth wave of COVID-19 in Canada. First row: Second wave from 0 up to 27^{th} and 46^{th} days out of 54 days (top left). The next two figures are the predicted values of the infectious variable $I(t)$ in the next seven days using SPADE4 (blue), SEIR model (magenta), and SμEIR model (green) correspond to those two training datasets versus ground truth (black). Second row: Prediction of $I(t)$ for the next seven days around the peak of the wave using SPADE4 (blue), SEIR model (magenta), and SμEIR model (green) versus ground truth (black).

exposed population is not known, we let the model choose the best estimate of $E(t_0)$ from $\{kI_0 : k \in \{0, 1, 5, 10, 15, 20, 25, 50, 80\}\}$, which is the value of $E(t_0)$ that gives the smallest ℓ_2-squared error over the training set. Finally, the initial susceptible variable is given by $S(t_0) = P - E(t_0) - I(t_0)$, where P is the approximate population of Canada, $P = 3.8 \times 10^7$. To avoid negligible values when the population P is extremely large compared to the number of active cases, we normalize the training data by dividing the entire dataset by $c * P$, where $c \in (0, 1]$. Here, we choose $c = 0.1$. Finally, for both the second and the fifth waves, we consider the number of training data m as the observed trends change, including the trends before the peak, around the peak, and after the peak. More precisely, m is chosen from the set $\{54, 90, 99, 108, 117, 126, 135, 144\}$ for the second wave and from the set $m \in \{27, 32, 35, 38, 41, 43, 46\}$ for the fifth wave (see the top left figures in Figure 4.6 and Figure 4.7). Note that m corresponds to the number of days starting from the starting date of a wave.

The forecast result illustrates the advantage of our method compared to the benchmark models. More precisely, the forecasts suggested by SPADE4 (blue curves in Figures 4.6 and 4.7) are closest to the true curve (in black) and it successfully picks up the changing nature of the curve around the peak quickly and adapts to the downward trend of the true curve. With the SEIR model, as the training data varies across before, around to after the peak (second row and top right plots in Figures 4.6 and 4.7), it tends to move from underprediction to overprediction to an extent where it is completely away from the true curve after the peak. The SμEIR model starts with underprediction and moves on to a mix of overprediction and underprediction inconsistently without picking up the changing slope of the trajectory as the size of training data varies (second row and top right plots in Figures 4.6 and 4.7).

4.3.2 Data from Cumulative Cases of Ebola in Guinea, Zika in Giradot and Influenza A/H7N9 in China

In this section, we use data based on cumulative cases of Ebola in Guinea, Zika in Giradot and influenza A/H7N9 in China. For each dataset, the parameters based on the size of the dataset, population P used for normalization of the dataset, preprocessing constant, size of the training set and forecast horizon have been summarized in Table 4.1. The initial conditions are chosen as $S(t_0) = P - E(t_0) - I(t_0)$, $I(t_0) = I_0$, $R(t_0) = 0$, where I_0 is the first data point in each dataset and we estimate $E(t_0)$ from $\{kI_0 : k \in \{0, 1, 5, 10, 15, 20, 25, 50, 80\}\}$, which is the value of $E(t_0)$ that gives the smallest ℓ_2-squared error over the training set. As in Section 4.3.1, we let the model choose the best estimate of $E(t_0)$. A convolution filter with the parameter $s = 10$ (see Equation (4.4)) is used on the derivative vector. We test SPADE4 on different sizes of training sets with cardinality m (see Table 4.1 for values of m considered). Comparison with the benchmark method is done using a one-week-forecast horizon of the proportion of cumulative cases.

Data	Total Data	Population	c	m (days)	Prediction Interval
Ebola Data from Guinea	572	135×10^6	10^{-3}	172 286	173-179 287-293
Zika Data from Giradot	93	95×10^3	1	27 65	28-34 66-72
Influenza Data from China	128	7×10^8	10^{-5}	44 64	45-51 65-71

Table 4.1: Parameters corresponding to datasets for Ebola, Zika and influenza A/H7N9.

The one-week-ahead forecasts of the Ebola, Zika, and flu datasets are given in Figure 4.8. From the second column in Figure 4.8, we can see that the benchmark methods with SEIR and SμEIR have a tendency to underpredict for Zika and flu datasets when the epidemic curve has a sharp slope and the number of training points is less. For the Ebola dataset, the SμEIR model learns well with less training data, however, the SEIR model overpredicts far away from the true curve. At this point, however, forecasts suggested

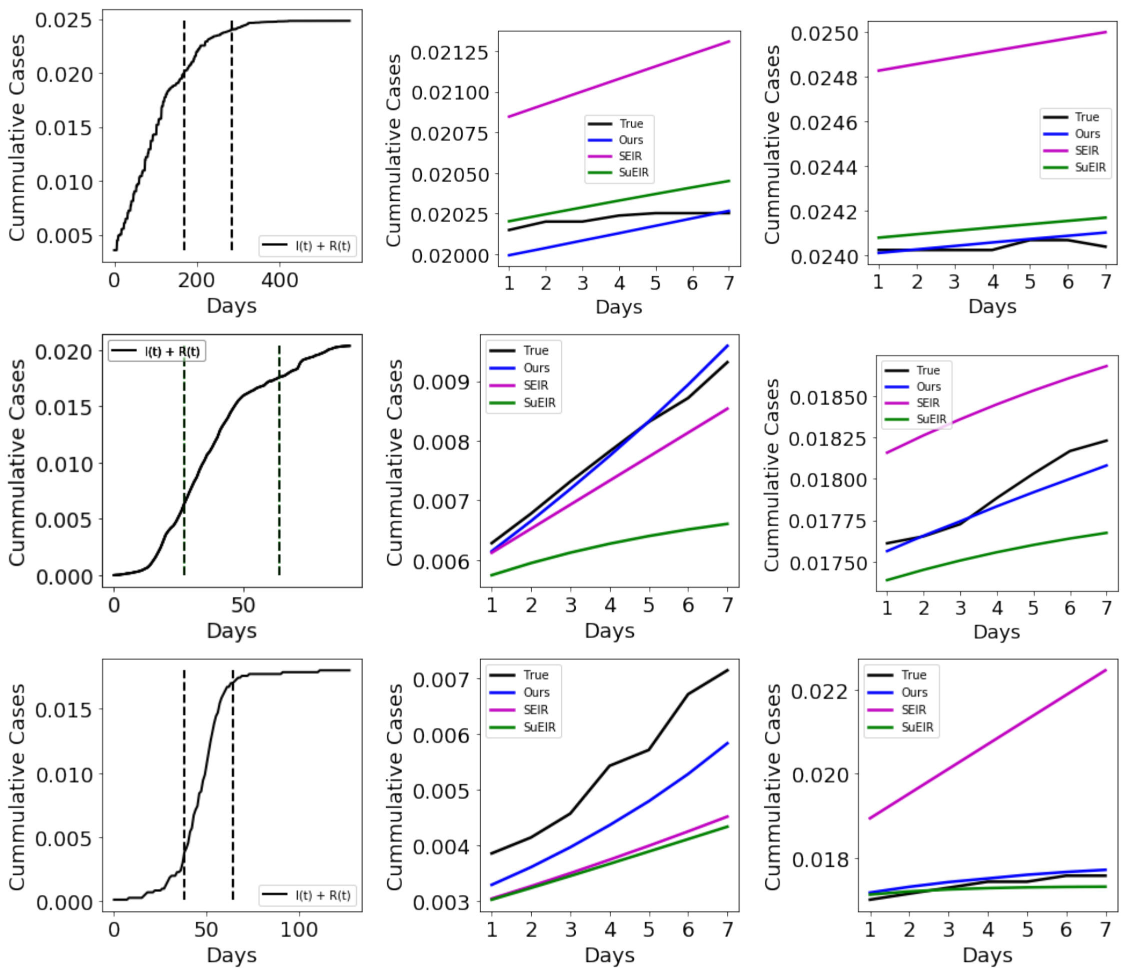


Figure 4.8: First column presents Ebola (top), Zika (center), and influenza A/H7N9 (bottom) datasets where the training data is from day 0th to the dashed vertical lines. The next two columns present the corresponding one-week ahead forecasts of our SPADE4 (in blue), SEIR (in magenta), and SμEIR (in green) versus ground truth (in black).

by SPADE4 are closest to the true values. Towards the end of the curve given in the third column of Figure 4.8, an increase in training data improves the SμEIR benchmark method (green curves) predictions which are much closer to the true values, especially for Ebola and flu datasets. The SEIR method predictions are still far away irrespective of the increase in training points as the method struggles to learn the changing nature of the trajectory. Predictions of SPADE4 remain consistently close to the true values with slight improvement with an increase in the size of training data. This can be seen more clearly in Table 4.2 in Section 4.3.3 where relative errors on the one-week-ahead horizon have been summarized. The results demonstrate that the performance of the benchmark method is dependent on the knowledge of the target model, the nature of the epidemic, and the number of data points. Fitting to an incorrect target model or scarce training data can affect prediction accuracy. However, SPADE4 can predict close to the true values without any prior knowledge of the underlying model.

4.3.3 Comparison with SEIR Models with Time-Varying Transmission Rate

In this section, we provide a comparison analysis between the performance of our SPADE4 against the time-varying transmission rate SEIR models [58, 130], denoted by SEIR$_{\beta(t)}$. In [58, 130], the authors present $\beta(t)$ using a fixed basis and learn $\beta(t)$ along with the other parameters of the model. In our paper, we compare with SEIR$_{\beta(t)}$, where $\beta(t)$ has a Legendre polynomial basis representation of order q, i.e., given $t \in [a, b]$,

$$\beta(t) = \sum_{k=0}^{q} \xi_k P_k(x),$$

where $x = \dfrac{2t - b - a}{b - a}$ and $P_k(x)$ are Legendre polynomials of order $0 \leq k \leq q$ given by

$$\begin{aligned}
&P_0(x) = 1, \quad P_1(x) = x, \\
&(2k + 1)P_{k+1}(x) = (k + 1)xP_k(x) - kP_{k-1}(x).
\end{aligned}$$

We use non-linear least squares to learn the parameters γ, σ and the weights ξ_k's representing $\beta(t)$ for each $q \in \{1, 2, 3, 4, 5, 6\}$ and let the model choose the best q using Bayesian Information Criterion (BIC). To compare the performance of our SPADE4 with SEIR$_{\beta(t)}$, we plot the seven-day validation errors with various sizes of the training data for the second and fifth wave of COVID-19 in Canada in Figure 4.9. Additionally, we also report in Table 4.2 the seven-day validation errors for the datasets with cumulative data explored in Section 4.3.2.

From Figure 4.9, one can conclude that SPADE4 consistently outperforms SEIR, SμEIR, and SEIR$_\beta(t)$ models. Especially, the performance gap is significant for the fifth wave when the peak is sharp and the amount of available data is limited.

A similar comparison has been investigated to datasets with cumulative data, including Ebola in Guinea, Zika in Giradot, and influenza A/H7N9 in China. The seven-day forecasting errors (in terms of relative validation error) are given in Table 4.2. We notice that SPADE4 performs the best in most cases except one with the influenza A/H7N9 dataset from China.

4.3.4 Prediction Interval

We propose a simple method to construct the prediction interval for SPADE4. The main idea is to use part of the training data to estimate the variance $\hat{\sigma}(t)$ of $\hat{I}_{ours}(t)$. Then,

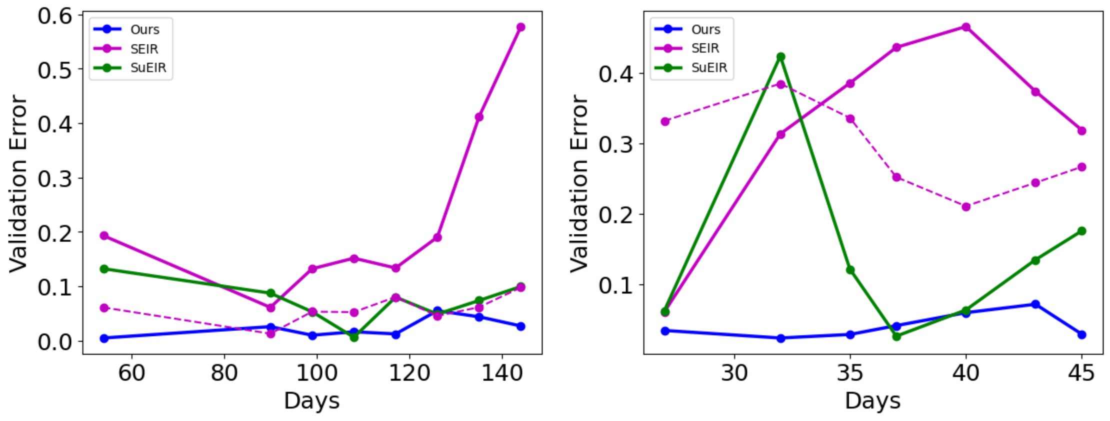


Figure 4.9: Validation error plot with a various number of training points for our SPADE4 (blue), SEIR (solid magenta), SEIR$_{\beta(t)}$ (broken magenta), and SμEIR model (green) models on second (left) and fifth (right) wave of COVID-19 dataset.

Data	m	SPADE4	SEIR	SEIR$_{\beta(t)}$	SμEIR
Ebola Data from Guinea	172	**0.0053**	0.0428	0.0811	0.0058
	286	**0.0012**	0.0365	0.00497	0.0036
Zika Data from Giradot	27	**0.0204**	0.0638	0.1168	0.2185
	65	**0.0055**	0.0302	0.0635	0.0213
Influenza Data from China	38	0.1783	0.3178	**0.0511**	0.3359
	64	**0.0079**	0.2017	0.0291	0.0097

Table 4.2: Relative error over one-week-ahead forecast horizon for different sizes of training data for the datasets corresponding to Ebola, Zika, and influenza A/H7N9.

the 95% prediction interval is $\hat{I}_{ours}(t) \pm 1.96\hat{\sigma}(t)$. The detail of our construction method is outlined in Algorithm 3. To illustrate the performance of this method, we construct the prediction interval for the Ebola, Zika, and influenza A/H7N9 datasets using the same setting as in Section 4.3.2. We can see that the true curve lies inside the shaded region of the 95% prediction interval (Figure 4.10).

4.3.5 Stability

Since our proposed method SPADE4 uses random features as a basis, in all our experiments above, we make use of a fixed seed to ensure the results are consistent over multiple executions of the simulations. However, in this section, we provide numerical evidence of the stability of SPADE4 with respect to the random basis generated for each simulation. Figure 4.11 gives all the possible predicted curves for the second wave of COVID-19 in Canada with different sizes of training data using SPADE4 (shaded in blue) over one

Algorithm 3 Prediction Interval of SPADE4

Given: Data $\{I(t_k)\}_{k=1}^{m_2}$; Forecast window T; a number $m_1 < m_2 - T$.

To find: 95% prediction interval for $\{I(t_k)\}_{k=m_2+1}^{m_2+T}$.

Algorithm:

```
1: Initialize $V = [\mathbf{v}_1, \dots, \mathbf{v}_{m_2-m_1-T}] \in \mathbb{R}^{T\times(m_2-m_1-T)}$ and $\hat{\Sigma} = [\hat{\sigma}_1, \dots, \hat{\sigma}_T] \in \mathbb{R}^T$.
2: for $i = 1, \dots, m_2 - m_1 - T$ do
       $\{\hat{I}_{ours}(t_k)\}_{k=m_1+i}^{m_1+i+T} \leftarrow \text{SPADE4}(\{I(t_k)\}_{k=1}^{m_1+i-1})$.
       $\mathbf{v}_i \leftarrow [\hat{I}_{ours}(t_{m_1+i}) - I(t_{m_1+i}), \dots, \hat{I}_{ours}(t_{m_1+i+T}) - I(t_{m_1+i+T})]^T$.
3: end for
4: for $j = 1, \dots, T$ do
       $$\hat{\sigma}_j = \sqrt{\frac{1}{m_2 - m_1 - T} \sum_{k=1}^{m_2-m_1-T} V[k,j]^2}.$$
5: end for
6: Find $\{\hat{I}_{ours}(t_k)\}_{k=m_2+1}^{m_2+T} \leftarrow \text{SPADE4}(\{I(t_k)\}_{k=1}^{m_2})$.
7: Prediction intervals: $[\hat{I}(t_{m_2+1}) \pm \hat{\sigma}_1 * 1.96, \dots, \hat{I}(t_{m_2+T}) \pm \hat{\sigma}_T * 1.96]$.
```

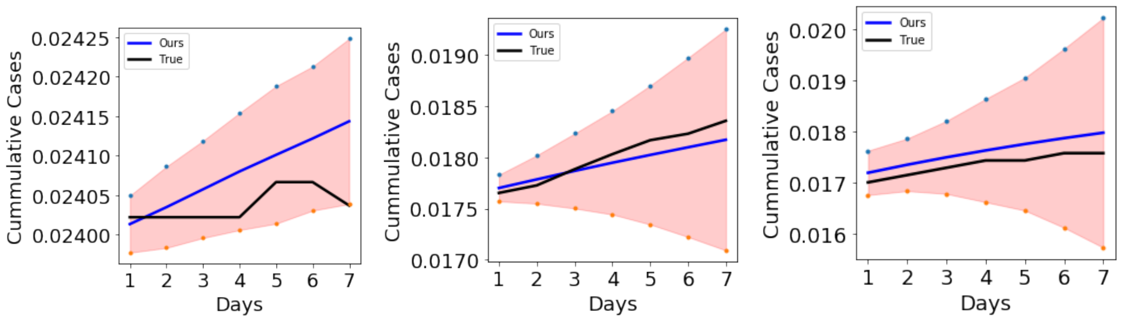


Figure 4.10: Prediction interval for the Ebola dataset with $m = 286$ ($m_1 = 266$ and $m_2 = 286$), Zika dataset with $m = 65$ ($m_1 = 45$ and $m_2 = 65$) and influenza A/H7N9 dataset with $m = 64$ ($m_1 = 44$ and $m_2 = 64$).

hundred randomly generated basis along with the predictions of the benchmark methods. We can see that all the predictions made by SPADE4 are closest to the true curve (in black) in comparison to the other methods depicting that SPADE4 performs well consistently irrespective of the random basis generated.

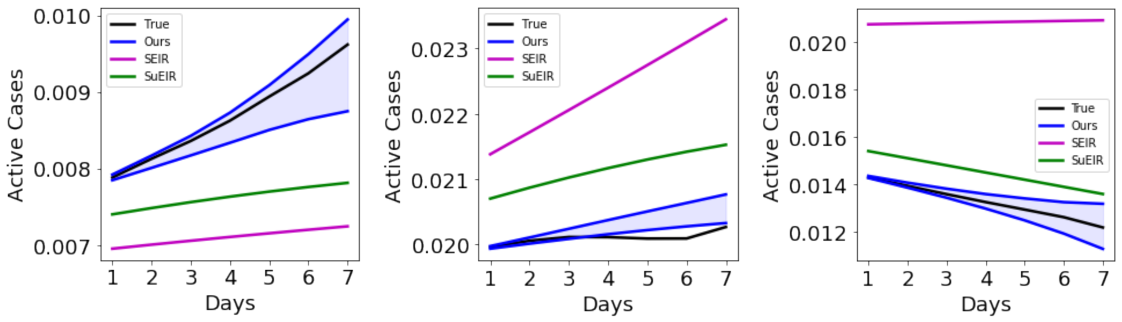


Figure 4.11: Predicted seven-day forecast curves using 100 runs of SPADE4 (shaded in blue) and the benchmark methods (SEIR model in magenta and SμEIR model in green) for the second wave of COVID-19 in Canada before the peak, at the peak and after the peak.

4.3.6 Varying Embedding Dimension

In this section, we demonstrate the stability of our method with respect to the choice of the embedding dimension p. We plot the validation error for one-week ahead forecasts for the second wave of COVID-19 in Canada with varying p and plot them in Figure 4.12. We use $p = 9$ in all our experiments since we follow the population division suggested by one of the most popular SEIR model in epidemiology. Since $d = 4$ in SEIR and SμEIR models, we choose $p = 2d+1 = 9$. The results in Figure 4.12 demonstrate the fact that using $p = 9$ gives consistently good results across different sizes of training data. While values of $p < 9$

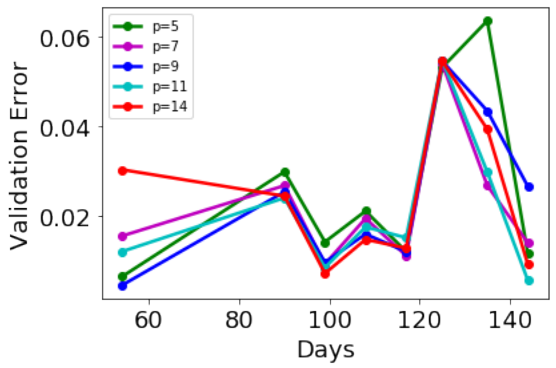


Figure 4.12: Validation error plot with various number of training points for SPADE4 with $p \in \{5, 7, 9, 11, 14\}$ for the second wave of COVID-19 in Canada.

do not perform the best, the error does improve for some cases when $p > 9$. However, for small input data, a large value of p is not feasible since it would further reduce the training data after the use of a time delay embedding map. Thus, while the embedding dimension p is a hyperparameter worthy of further exploration, our experiments suggest that choosing $p = 2d + 1$ outperforms other choices when both, the size of input data and the validation error are taken into account.

4.4 Other Function Approximation Methods

In this section, we discuss the performance of SPADE4 by considering alternative function-approximation models, namely polynomial approximation and neural networks.

In the first case, we replace the dictionary matrix $\mathbf{A}$ in SPADE4 with a polynomial basis. We compare with an orthogonal polynomial basis of orders 2 and 3. Given the input data $\{y(t_k)\}_{k=1}^{m}$, embedding dimension p and order of the polynomial basis n, the collection of trial functions using Legendre polynomials (as given in [126]) is given by

$$\mathbf{A}_{leg} = \begin{bmatrix} 1 & \sqrt{3}y(t_1) & \cdots & \sqrt{3}y(t_p) & \dfrac{\sqrt{5}(3[y(t_1)]^2 - 1)}{2} & 3y(t_1)y(t_2) & \cdots \\ 1 & \sqrt{3}y(t_2) & \cdots & \sqrt{3}y(t_{p+1}) & \dfrac{\sqrt{5}(3[y(t_2)]^2 - 1)}{2} & 3y(t_2)y(t_2) & \cdots \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ 1 & \sqrt{3}y(t_{m-p+1}) & \cdots & \sqrt{3}y(t_m) & \dfrac{\sqrt{5}(3[y(t_{m-p+2})]^2 - 1)}{2} & 3y(t_{m-p+2})y(t_{m-p+3}) & \cdots \end{bmatrix}.$$

For our experiments, we choose second and third-order Legendre polynomials as the dictionaries. We then solve the optimization problem as described in Eq. (4.7) and obtain predictions using Eq. (4.8).

For the sake of further comparison, a shallow neural network approximation of the function f in Eq. (4.2) is also used in place of the random feature approximation i.e.,

$$f(\mathbf{h}) = \mathbf{W}_2\phi(\mathbf{W}_1\mathbf{h} + \mathbf{b}_1), \tag{4.11}$$

where for $\mathbf{h} \in \mathbb{R}^p$, we have trainable weights and biases given by $\mathbf{W}_1 \in \mathbb{R}^{N\times p}$, $\mathbf{b}_1 \in \mathbb{R}^N$ and $\mathbf{W}_2 \in \mathbb{R}^{1\times N}$. We also adjust the number of neurons in the hidden layer to match the number of trainable parameters in SPADE4. We learn the parameters of the neural network model by minimizing the ℓ_2 norm of the true data with the approximated function given in Eq. (4.11) using gradient descent.

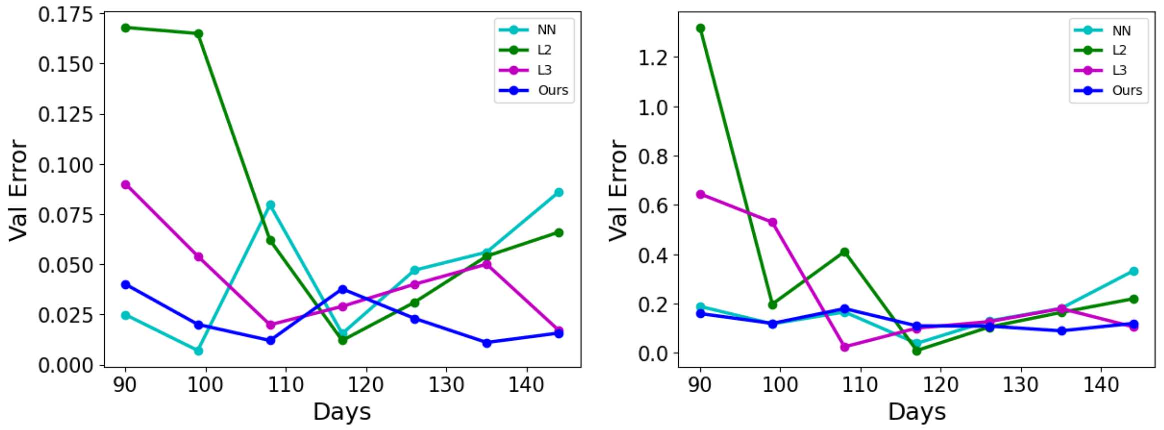


Figure 4.13: Comparison of SPADE4 against Legendre polynomial dictionary of orders 2,3 and an NN approximation using simulated data with 5% noise. Predictions are made for seven-day-ahead (left) and fourteen-day-ahead (right) intervals.

In the plots, we denote approximations using the Legendre polynomial of orders 2 and 3 by $L2$ and $L3$ respectively. NN denotes approximation using a (shallow) neural network. We compare our method with both one-week and two-week predictions. We can see that for each experiment, our proposed model is comparable to a different method. For example, in Figure 4.13, we see that for a two-week prediction, SPADE4 performs comparably well to an NN, while for a one-week prediction in Figure 4.14, the performance of SPADE4 is comparable to the dictionary of second order Legendre polynomials. We see that SPADE4 outperforms most of the methods when the training data is limited which can be noted in the first two points of the plots in all the Figures. A key observation is that while SPADE4 may not always give the lowest validation errors, it always performs comparably to the best-performing method for each of the datasets. To summarize, while the nature of the dataset may affect the performance of the other approximation methods, SPADE4 gives reliable predictions with low validation errors regardless of the data considered.

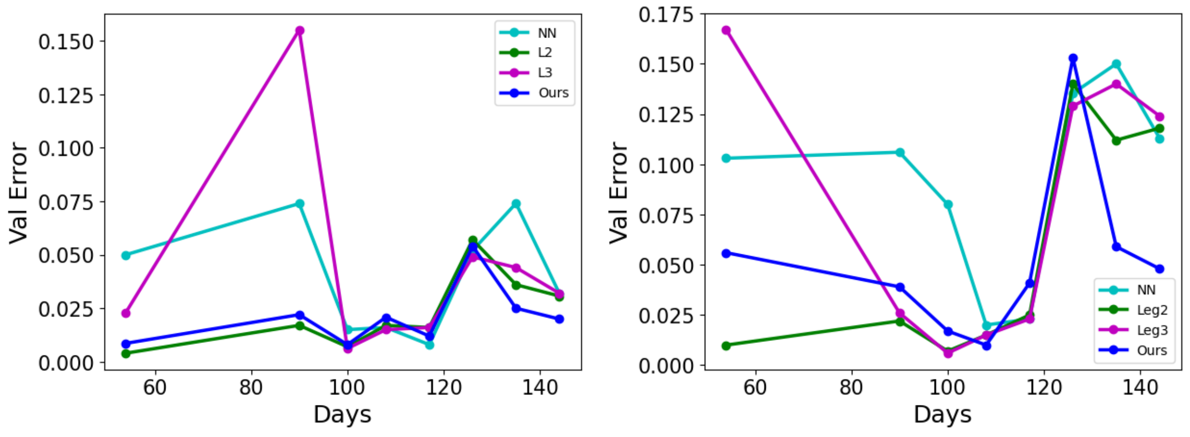


Figure 4.14: Comparison of SPADE4 against Legendre polynomial dictionary of orders 2,3 and an NN approximation using the second wave of the COVID-19 dataset. Predictions are made for seven-day-ahead (left) and fourteen-day-ahead (right) intervals.

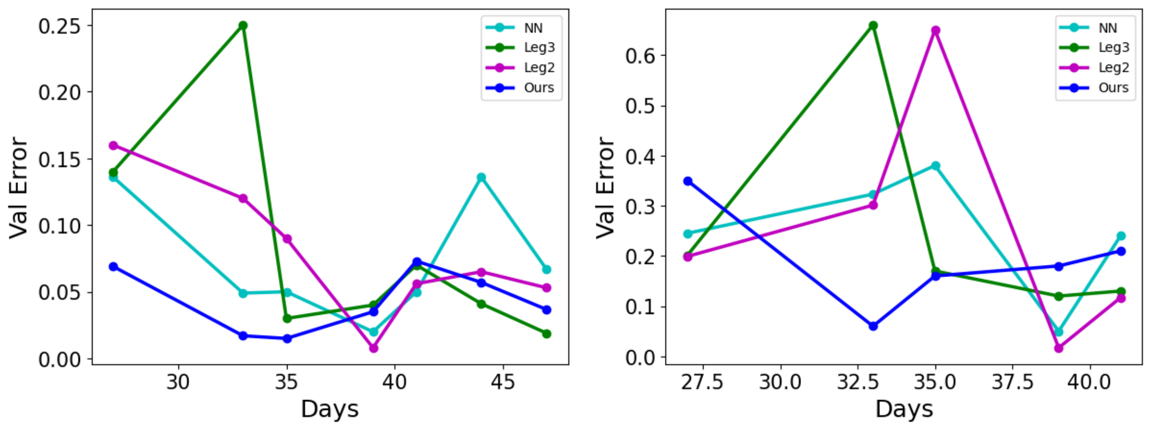


Figure 4.15: Comparison of SPADE4 against Legendre polynomial dictionary of orders 2,3 and an NN approximation using the fifth wave of the COVID-19 dataset. Predictions are made for seven-day-ahead (left) and fourteen-day-ahead (right) intervals.

4.5 Limitations

Since the performance of SPADE4 is partially dependent on an accurate estimation of the derivative, the method might not give accurate results if the data is extremely noisy. Although we do use simple convolution-based smoothing filters to address noisy datasets, more advanced denoising techniques might be required for higher levels of noise. Another drawback of the method is the hyperparameter choice of the embedding dimension. While Takens' theorem gives an estimate of the embedding dimension (at least $2d+1$), we need an estimate of the underlying dimension d in the first place to have the estimate of $2d+1$.

Chapter 5

Diffusion Random Features Model

In this chapter, we present the notion of learning distributions, in particular learning the mean and/or variance of a Gaussian distribution using a random feature-based method for diffusion models. Diffusion models have always been seen as complex models requiring a huge amount of data to perform well. Although they are well known for generating high-resolution images, diffusion models have high computational power requirements and are rarely interpretable. Our motivation to work with diffusion models and random features comes from an attempt to build an interpretable model for learning distributions through diffusion models. We build the model by considering a random feature model to be learned for each diffusion step. This architecture helps us to build a stacked-up model consisting of random feature models layered through time. The model is interpretable since the random feature model can be analyzed for error bounds for each fixed timestep, and as the number of timesteps is finite, the error bounds can be concatenated over time. We derive bounds for sampled data using existing theory from [35] and our proposed results. The contents of this chapter are taken from [124], with modification. the chapter is organized as follows: Section 5.1 gives the background and motivation followed by the algorithm and theoretical results in Section 5.2 and the experimental results in Section 5.3.

5.1 Diffusion Models and Related Works

We first recall some useful notations and terminologies corresponding to diffusion models. In this chapter, we denote $\mathcal{N}(\mathbf{0}, \mathbf{I}_d)$ the $d-$ dimensional Gaussian distribution with the zero mean vector and the identity covariance matrix. We use the notation $p(\mathbf{x}) = \mathcal{N}(\boldsymbol{\mu}, \boldsymbol{\Sigma})$ to mean that $p(\mathbf{x})$ is the p.d.f. of a random vector $\mathbf{x}$ following the multivariate normal distribution with mean vector $\boldsymbol{\mu}$ and covariance matrix $\boldsymbol{\Sigma}$.

A diffusion model consists of two Markov chains: a forward process and a reverse process. The goal of the forward process is to degrade the input sample by gradually adding noise to the data over a fixed number of timesteps. The reverse process involves learning to undo the added-noise steps using a parameterized model. Knowledge of the reverse process helps to generate new data starting with a random noisy vector followed by sampling through the reverse Markov chain [81, 152].

5.1.1 Forward Process

The forward process degrades the input data such that $q(\mathbf{x}_K) \approx \mathcal{N}(\mathbf{0}, \mathbf{I}_d)$. More precisely, let $\mathbf{x}_0 \in \mathbb{R}^d$ be input from an unknown distribution with p.d.f. $q(\mathbf{x}_0)$. Suppose $0 < \beta_1 \leq \beta_2 \leq \cdots \leq \beta_K < 1$ is a given variance schedule. As defined in Definition 2.5.14 in Chapter 2, the forward process generates a sequence of random variables $\mathbf{x}_1, \mathbf{x}_2, ..., \mathbf{x}_K$ with conditional distributions

$$q(\mathbf{x}_{k+1}|\mathbf{x}_k) = \mathcal{N}(\mathbf{x}_{k+1}; \sqrt{1-\beta_{k+1}}\mathbf{x}_k, \beta_{k+1}\mathbf{I}_d) \text{ for } k = 0, \cdots, K-1. \tag{5.1}$$

Let $\alpha_k = 1 - \beta_k$ for $k = 1, \ldots, K$ and $\overline{\alpha}_k = \prod_{i=1}^{k} \alpha_i$.

From Eq. (2.63) in Chapter 2, the conditional distribution $q(\mathbf{x}_{k+1}|\mathbf{x}_0)$ is

$$q(\mathbf{x}_{k+1}|\mathbf{x}_0) = \mathcal{N}(\mathbf{x}_{k+1}; \sqrt{\overline{\alpha}_{k+1}}\,\mathbf{x}_0, (1-\overline{\alpha}_{k+1})\mathbf{I}_d). \tag{5.2}$$

Note that, at $k = K$, we have

$$\mathbf{x}_K = \sqrt{\overline{\alpha}_K}\,\mathbf{x}_0 + \sqrt{1-\overline{\alpha}_K}\,\boldsymbol{\epsilon},$$

where $\boldsymbol{\epsilon} \sim \mathcal{N}(\mathbf{0}, \mathbf{I}_d)$. Since $0 < \beta_1 \leq \beta_2 \leq \cdots \leq \beta_K < 1$, $0 < \overline{\alpha}_K < \alpha_1^K < 1$. Therefore, $\lim\limits_{K\to\infty} \overline{\alpha}_K = 0$. Hence, $q(\mathbf{x}_K) = \int q(\mathbf{x}_K|\mathbf{x}_0)q(\mathbf{x}_0)d\mathbf{x}_0 \approx \mathcal{N}(\mathbf{0}, \mathbf{I}_d)$, i.e., as the number of timesteps becomes very large, the distribution $q(\mathbf{x}_K)$ will approach the Gaussian distribution with mean $\mathbf{0}$ and covariance $\mathbf{I}_d$.

5.1.2 Reverse Process

The reverse process aims to generate data from the input distribution by sampling from $q(\mathbf{x}_K)$ and gradually denoising for which one needs to know the reverse distribution $q(\mathbf{x}_{k-1}|\mathbf{x}_k)$. In general, computation of $q(\mathbf{x}_{k-1}|\mathbf{x}_k)$ is intractable without the knowledge of $\mathbf{x}_0$. Therefore, we condition the reverse distribution on $\mathbf{x}_0$ in order to obtain the mean and variance for the reverse process. From Lemma 2.5.16 in Chapter 2 we have,

$$q(\mathbf{x}_{k-1}|\mathbf{x}_k, \mathbf{x}_0) = \frac{\sqrt{(1-\overline{\alpha}_k)^d}}{\sqrt{(2\pi\beta_k(1-\overline{\alpha}_{k-1}))^d}} \exp\left(-\frac{1}{2}\frac{(\mathbf{x}_{k-1}-\tilde{\boldsymbol{\mu}}_k)^T(\mathbf{x}_{k-1}-\tilde{\boldsymbol{\mu}}_k)}{\tilde{\beta}_k}\right), \tag{5.3}$$

where,

$$\tilde{\boldsymbol{\mu}}_k = \frac{\sqrt{\alpha_k}(1-\overline{\alpha}_{k-1})}{1-\overline{\alpha}_k}\mathbf{x}_k + \frac{\sqrt{\overline{\alpha}_{k-1}}\beta_k}{1-\overline{\alpha}_k}\mathbf{x}_0 \text{ and } \tilde{\beta}_k = \frac{1-\overline{\alpha}_{k-1}}{1-\overline{\alpha}_k}\beta_k. \tag{5.4}$$

Thus the reverse distribution conditioned on $\mathbf{x}_0$ is $q(\mathbf{x}_{k-1}|\mathbf{x}_k, \mathbf{x}_0) = \mathcal{N}(\tilde{\boldsymbol{\mu}}_k, \tilde{\beta}_k \mathbf{I}_d)$, where $\tilde{\boldsymbol{\mu}}_k, \tilde{\beta}_k$ are obtained above. Our aim is to learn the reverse distribution from the obtained conditional reverse distribution. From Markovian theory, if β_k's are small, the reverse process is also Gaussian [132]. Let $p_\theta(\mathbf{x}_{k-1}|\mathbf{x}_k)$ be the learned reverse distribution, then Markovian theory tells us that $p_\theta(\mathbf{x}_{k-1}|\mathbf{x}_k) = \mathcal{N}(\boldsymbol{\mu}_\theta(\mathbf{x}_k, k), \boldsymbol{\Sigma}_\theta(\mathbf{x}_k, k))$, where $\boldsymbol{\mu}_\theta(\mathbf{x}_k, k)$ and $\boldsymbol{\Sigma}_\theta(\mathbf{x}_k, k)$ are the learned mean vector and variance matrix respectively. Since the derived covariance matrix $\tilde{\beta}_k \mathbf{I}_d$ for conditional reverse distribution is constant, $\boldsymbol{\Sigma}_\theta(\mathbf{x}_k, k)$ need not be learnt. In [70], the authors show that choosing $\boldsymbol{\Sigma}_\theta(\mathbf{x}_k, t)$ as $\beta_k \mathbf{I}_d$ or $\tilde{\beta}_k \mathbf{I}_d$ yield similar results and thus we fix $\boldsymbol{\Sigma}_\theta(\mathbf{x}_k, k) = \beta_k \mathbf{I}_d$ for simplicity. Furthermore, since $\mathbf{x}_k$ is also available as input to the model, the loss function derived in [70] as a KL divergence between $q(\mathbf{x}_{k-1}|\mathbf{x}_k, \mathbf{x}_0)$ and $p_\theta(\mathbf{x}_{k-1}|\mathbf{x}_k)$ can be simplified as

$$D_{KL}(q(\mathbf{x}_{k-1}|\mathbf{x}_k, \mathbf{x}_0) \| p_\theta(\mathbf{x}_{k-1}|\mathbf{x}_k) = \mathbb{E}_q \left[\frac{1}{2\beta_k} \left\| \tilde{\boldsymbol{\mu}}_k(\mathbf{x}_k, \mathbf{x}_0) - \boldsymbol{\mu}_\theta(\mathbf{x}_k, k) \right\|^2 \right] + \text{const}, \tag{5.5}$$

where for each timestep k, $\tilde{\boldsymbol{\mu}}_k(\mathbf{x}_k, \mathbf{x}_0)$ denotes the mean of the reverse distribution conditioned on $\mathbf{x}_0$ i.e., $q(\mathbf{x}_{k-1}|\mathbf{x}_k, \mathbf{x}_0)$ and $\boldsymbol{\mu}_\theta(\mathbf{x}_k, k)$ denotes the learned mean vector. Thus, the above equation predicts the mean of the reverse distribution when conditioned on $\mathbf{x}_0$. Substituting $\mathbf{x}_0 = \frac{1}{\sqrt{\bar{\alpha}_k}}(\mathbf{x}_k - \sqrt{1-\bar{\alpha}_k}\boldsymbol{\epsilon}_k)$ in Eq. (5.4) we can obtain $\tilde{\boldsymbol{\mu}}(\mathbf{x}_k, k) = \frac{1}{\sqrt{\alpha_k}}\left(\mathbf{x}_k - \frac{\beta_k}{\sqrt{1-\bar{\alpha}_k}}\boldsymbol{\epsilon}\right)$. Further, since $\mathbf{x}_k$ is known, we can use the formula for $\boldsymbol{\mu}_\theta(\mathbf{x}_k, k) = \frac{1}{\sqrt{\alpha_k}}\left(\mathbf{x}_k - \frac{\beta_k}{\sqrt{1-\bar{\alpha}_k}}\boldsymbol{\epsilon}_\theta(\mathbf{x}_k, k)\right)$. We can simplify Eq. (5.5) as:

$$\mathbb{E}_{k,\mathbf{x}_0,\boldsymbol{\epsilon}} \left[\frac{1}{2\alpha_k(1-\bar{\alpha}_k)} \|\boldsymbol{\epsilon} - \boldsymbol{\epsilon}_\theta(\mathbf{x}_k, k)\|^2 \right]. \tag{5.6}$$

where $\boldsymbol{\epsilon}_\theta$ now denotes a function approximator intended to predict the noise from $\mathbf{x}_k$. The above results show that we can either train the reverse process mean function approximator $\boldsymbol{\mu}_\theta$ to predict $\tilde{\boldsymbol{\mu}}_k$ or modify using its parameterization to predict $\boldsymbol{\epsilon}$. In our proposed algorithm, we choose to use the loss function from Eq. (5.6) since it is one of the simplest forms to train and understand. This formulation of DDPM also helps us to harness the power of SDEs in diffusion models through its connection to DSMs [18].

5.1.3 DDPM and DSM

We can also apply the DDPM algorithm for score matching by formulating the DDPM objective as a DSM objective.

$$
\begin{aligned}
L_{\text{DDPM}} &= \mathbb{E}_{k,\mathbf{x}_0,\boldsymbol{\epsilon}}\left[\frac{1}{2\alpha_k(1-\overline{\alpha}_k)}\|\boldsymbol{\epsilon} - \boldsymbol{\epsilon}_\theta(\mathbf{x}_k, k)\|^2\right] && (5.7)\\
&= \mathbb{E}_{k,\mathbf{x}_0,\boldsymbol{\epsilon}}\left[\frac{1}{2\alpha_k(1-\overline{\alpha}_k)}\left\|\frac{\mathbf{x}_k - \sqrt{\overline{\alpha}_k}\mathbf{x}_0}{\sqrt{1-\overline{\alpha}_k}} - \boldsymbol{\epsilon}_\theta(\mathbf{x}_k, k)\right\|^2\right] && (5.8)\\
&= \mathbb{E}_{k,\mathbf{x}_0,\mathbf{x}_k}\left[\frac{1}{2\alpha_k(1-\overline{\alpha}_k)}\left\|\frac{\mathbf{x}_k - \sqrt{\overline{\alpha}_k}\mathbf{x}_0}{1-\overline{\alpha}_k}\sqrt{1-\overline{\alpha}_k} - \frac{\sqrt{1-\overline{\alpha}_k}}{\sqrt{1-\overline{\alpha}_k}}\boldsymbol{\epsilon}_\theta(\mathbf{x}_k, k)\right\|^2\right] && (5.9)\\
&= \mathbb{E}_{k,\mathbf{x}_0,\mathbf{x}_k}\left[\frac{1}{2\alpha_k}\left\| - \nabla_{\mathbf{x}_k}\log q(\mathbf{x}_k|\mathbf{x}_0) - \frac{1}{\sqrt{1-\overline{\alpha}_k}}\boldsymbol{\epsilon}_\theta(\mathbf{x}_k, k)\right\|^2\right] && (5.10)\\
&= \mathbb{E}_{k,\mathbf{x}_0,\mathbf{x}_k}\left[\frac{1}{2\alpha_k}\left\|s_\theta(\mathbf{x}_k, k) - \nabla_{\mathbf{x}_k}\log q(\mathbf{x}_k|\mathbf{x}_0)\right\|^2\right] = L_{\text{DSM}}, && (5.11)
\end{aligned}
$$

where $s_\theta(\mathbf{x}_k, k) = -\frac{1}{\sqrt{1-\overline{\alpha}_k}}\boldsymbol{\epsilon}_\theta(\mathbf{x}_k, k)$. The above formulation is known as denoising score matching (DSM), which is equivalent to the objective of DDPM. Furthermore, the objective of DSM is also related to the objective of score-based generative models using SDEs [133]. We briefly discuss the connection between diffusion models, SDEs, and DSM in the upcoming section [70, 133, 152].

5.1.4 Diffusion Models and SDEs

The forward process can also be generalized to stochastic differential equations (SDEs) if infinite time steps or noise levels are considered (SDEs) as proposed in [152]. To formulate the forward process as an SDE, let $t = \frac{k}{K}$ and define functions $\mathbf{x}(t), \beta(t)$ and $\boldsymbol{\epsilon}(t)$ such that $\mathbf{x}(\frac{k}{K}) = \mathbf{x}_k$, $\beta(\frac{k}{K}) = K\beta_k$ and $\boldsymbol{\epsilon}(\frac{k}{K}) = \boldsymbol{\epsilon}_k$. Note that in the limit $K \to \infty$, we get $t \in [0, 1]$.

Using the derivations from [152], the forward process can be written as an SDE of the form,

$$d\mathbf{x} = \frac{-\beta(t)}{2}\mathbf{x}dt + \sqrt{\beta(t)}d\mathbf{w}, \tag{5.12}$$

where $\mathbf{w}$ is the standard Wiener process. The above equation now is in the form of an SDE

$$d\mathbf{x} = f(\mathbf{x}, t)dt + g(t)d\mathbf{w}, \tag{5.13}$$

where $f(\mathbf{x}, t)$ and $g(t)$ are diffusion and drift functions of the SDE respectively, and $\mathbf{w}$ is a standard Wiener process. The above process can be reversed by solving the reverse SDE,

$$d\mathbf{x} = [f(\mathbf{x}, t) - g(t)^2 \nabla_{\mathbf{x}(t)} \log q(\mathbf{x}(t))]dt + g(t)d\overline{\mathbf{w}}, \tag{5.14}$$

where $\overline{\mathbf{w}}$ is a standard Wiener process backwards in time, and dt denotes an infinitesimal negative time step and $q(\mathbf{x}(t))$ is the marginal distribution of $\mathbf{x}(t)$. Note that in particular for Eq.(5.12), the reverse SDE will be of the form

$$d\mathbf{x} = \left[\frac{\beta(t)}{2} - \beta(t)\nabla_{\mathbf{x}(t)} \log q(\mathbf{x}(t))\right] dt + \sqrt{\beta(t)}d\overline{\mathbf{w}}. \tag{5.15}$$

The unknown term $\nabla_{\mathbf{x}(t)} \log q(\mathbf{x}(t))$ is called the score function and is estimated by training a parameterized model $s_\theta(\mathbf{x}(t), t)$ via minimization of the loss given by

$$\mathbb{E}_{q(\mathbf{x}(t))}\left[\frac{1}{2}\left\|s_\theta(\mathbf{x}(t), t) - \nabla_{\mathbf{x}(t)} \log q(\mathbf{x}(t))\right\|_2^2\right]. \tag{5.16}$$

Note that since $q(\mathbf{x}(0))$ is unknown, therefore the distribution $q(\mathbf{x}(t))$ and subsequently the score function $\nabla_{\mathbf{x}(t)} \log q(\mathbf{x}(t))$ are also unknown. Referring to results from [147], we see that the loss in Eq. (5.16) is equivalent to the denoising score matching (DSM) objective given by,

$$\mathbb{E}_{q(\mathbf{x}(t),\mathbf{x}(0))}\left[\frac{1}{2}\left\|s_\theta(\mathbf{x}(t), t) - \nabla_{\mathbf{x}(t)} \log q(\mathbf{x}(t)|\mathbf{x}(0))\right\|_2^2\right]. \tag{5.17}$$

We can see that the above objective is the same as the objective of DSM in the discrete setting. Apart from the success of score-based models using SDEs, an additional advantage of formulating the diffusion model using SDEs is the theoretical analysis based on results from SDEs. In our paper, we aim to use this connection to build a theoretical understanding of our proposed model.

While there are remarkable results for improving training and sampling for diffusion models, little has been explored in terms of the model architectures. Since distribution learning and data generation is a complex task, it is unsurprising that conventional diffusion models are computationally expensive. From previous works in [70, 152, 155], U-Net (or variations on U-Net combined with ResNet, CNNs, etc.) architecture is still the most commonly used model for diffusion models. The architecture of U-Net is designed in a way that it preserves the input dimension of the data i.e., the output and input dimension are the same. Additionally, U-Nets generally consist of a contracting (downsampling) and expanding (upsampling) path. The contracting path consists of sequences of convolutional and max-pooling layers that downsample the input images and extract the layers. However, all these architectures have millions of parameters making training (and sampling) cumbersome. An alternative approach to reduce the complexity of machine learning algorithms is to use a random feature model (RFM) [109, 110] for approximating the kernels using a randomized basis. RFMs are derived from kernel-based methods which utilize a pre-defined nonlinear function basis called kernel $K(\mathbf{x}, \mathbf{y})$. From the neural network point of view, an RFM is a two-layer network with a fixed single hidden layer sampled randomly [109,110]. Not only do random feature-based methods give similar results to that of a shallow network, but the model in itself is also interpretable which makes it a favorable method to use. Some recent works which use random features for a variety of tasks are explored in [37,38,105,115,122,123]. However, random features can lack expressibility due to their structure and thus we aim to propose an architecture that can be more flexible in learning yet retaining properties of random features.

5.2 Algorithm and Theory

Our proposed model is a diffusion model inspired random feature model. The main idea of our model is to build an interpretable diffusion random feature architecture. Our work is inspired by *denoising diffusion probabilistic model* (DDPM) proposed in [70] and semi-random features proposed in [78]. Let $\mathbf{x}_0 \in \mathbb{R}^d$ be the input data belonging to an unknown distribution $q(\mathbf{x}_0)$. Let K denote the total number of timesteps in which the forward process is applied. Suppose N is the number of features. For each timestep k, we build a noise predictor function ϵ_θ of the form

$$\epsilon_\theta(\mathbf{x}_k, k) = \left(\sin\left(\mathbf{x}_k^T \mathbf{W} + \mathbf{b}^T\right) \odot \cos\left(\boldsymbol{\tau}_k^T \boldsymbol{\theta}^{(1)}\right)\right) \boldsymbol{\theta}^{(2)}, \tag{5.18}$$

where $\mathbf{x}_k \in \mathbb{R}^d$, $\mathbf{W} \in \mathbb{R}^{d\times N}$, $\mathbf{b} = \begin{bmatrix} b_1 & \dots & b_N \end{bmatrix}^T \in \mathbb{R}^N$, $\boldsymbol{\theta}^{(1)} = \begin{bmatrix} \theta_{ki}^{(1)} \end{bmatrix} \in \mathbb{R}^{K\times N}$, $\boldsymbol{\tau}_k \in \mathbb{R}^K$, $\boldsymbol{\theta}^{(2)} = \begin{bmatrix} \theta_{ij}^{(2)} \end{bmatrix} \in \mathbb{R}^{N\times d}$, and $\odot$ denotes element-wise multiplication. The vector $\boldsymbol{\tau}_k$ $(k \geq 1)$ is a one-hot vector with the position of one corresponding to the timestep k. The motivation to use trainable weights corresponding to the time parameter is twofold: first, we want to associate importance to the timestep being used when optimizing the weights; secondly, we aim to build a random feature model layered through time. The inspiration for using cosine as an activation comes from the idea of positional encoding used for similar tasks. In general, positional encoding remains fixed, but for our method, we wish to make the weights associated with timestep random and trainable. This is done so that the model learns the noise level associated with the timestep. Our aim is to train the parameters $\boldsymbol{\theta} = \{\boldsymbol{\theta}^{(1)}, \boldsymbol{\theta}^{(2)}\}$ while $\mathbf{W}$ and $\mathbf{b}$ are randomly sampled and fixed. The model is trained using Algorithm 4.

5.2.1 Theoretical Results

We provide theoretical results corresponding to our proposed model. We first formulate our proposed model as a time-dependent layered random features model, followed by the proof of obtaining sample generation bounds. The obtained bounds help to prove that DRFM

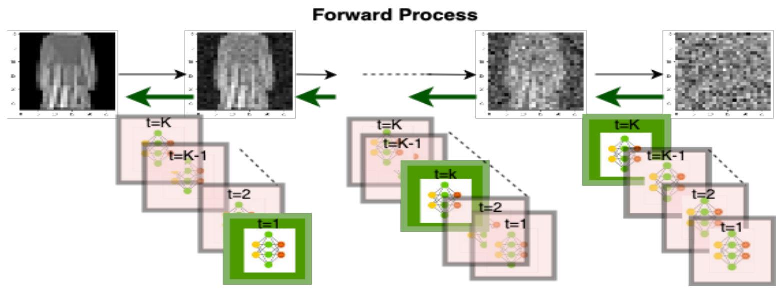


Figure 5.1: Representation of DRFM. The green boxes denote the random feature layer which is active and corresponds to the timestep selected, while the other remains fixed.

is capable of generating samples from the distribution on which it was trained using the results from [35].

For a fixed timestep k, let $\mathbf{x}_k = \mathbf{y}$ then Eq. (5.18) can be written as:

$$\begin{aligned}\epsilon_\theta(\mathbf{x}_k, k) &= \left(\sin\left(\mathbf{x}_k^T\mathbf{W} + \mathbf{b}^T\right) \odot \cos\left(\boldsymbol{\tau}_k\boldsymbol{\theta}^{(1)}\right)\right)\boldsymbol{\theta}^{(2)} \\ &= \left(\sin\left(\begin{bmatrix} y_1 & \dots & y_d\end{bmatrix}\begin{bmatrix}\omega_{11} & \dots & \omega_{1N} \\ \vdots & \dots & \vdots \\ \omega_{d1} & \dots & \omega_{dN}\end{bmatrix} + \begin{bmatrix} b_1 \\ \vdots \\ b_N\end{bmatrix}\right) \odot \begin{bmatrix}\cos\left(\theta_{k1}^{(1)}\right) & \dots & \cos\left(\theta_{kN}^{(1)}\right)\end{bmatrix}\right)\boldsymbol{\theta}^{(2)} \\ &= \sin\left(\mathbf{x}_k^T\mathbf{W} + \mathbf{b}^T\right)\begin{bmatrix}\cos\left(\theta_{k1}^{(1)}\right)\theta_{11}^{(2)} & \dots & \cos\left(\theta_{k1}^{(1)}\right)\theta_{1d}^{(2)} \\ \vdots & \dots & \vdots \\ \cos\left(\theta_{kN}^{(1)}\right)\theta_{N1}^{(2)} & \dots & \cos\left(\theta_{kN}^{(1)}\right)\theta_{Nd}^{(2)}\end{bmatrix}.\end{aligned} \tag{5.19}$$

For each $j = 1, ..., N$, let $a_j = \sin\left(\sum_{i=1}^{d} y_i\omega_{ij} + b_j\right)$. Note that a_j's are fixed. Then Eq. (5.19) becomes,

$$\epsilon_\theta(\mathbf{y}, k) = \begin{bmatrix} a_1\cos\left(\theta_{k1}^{(1)}\right) & \dots & a_N\cos\left(\theta_{kN}^{(1)}\right)\end{bmatrix}\begin{bmatrix}\theta_{11}^{(2)} & \dots & \theta_{1d}^{(2)} \\ \vdots & \dots & \vdots \\ \theta_{N1}^{(2)} & \dots & \theta_{Nd}^{(2)}\end{bmatrix} \tag{5.20}$$

$$= \begin{bmatrix} a_1 & \dots & a_N\end{bmatrix}\begin{bmatrix}\cos\left(\theta_{k1}^{(1)}\right)\theta_{11}^{(2)} & \dots & \cos\left(\theta_{k1}^{(1)}\right)\theta_{1d}^{(2)} \\ \vdots & \dots & \vdots \\ \cos\left(\theta_{kN}^{(1)}\right)\theta_{N1}^{(2)} & \dots & \cos\left(\theta_{kN}^{(1)}\right)\theta_{Nd}^{(2)}\end{bmatrix}. \tag{5.21}$$

Algorithm 4 Training and sampling using DRFM

```
Input: Sample $\mathbf{x}_0 \sim q(\mathbf{x}_0)$ where $q$ is an unknown distribution, variance schedule $\beta = \{\beta_1, ..., \beta_K\}$ such that $0 < \beta_1 \leq \beta_2 \leq \cdots \leq \beta_K < 1$, random weight matrix $\mathbf{W} = [\omega_{ij}]$ and bias vector $\mathbf{b}$ sampled from a distribution $\rho$ and total number of epochs epoch.
Algorithm:

    Training

 1: Choose random timestep $k \in \{1, 2, ..., K\}$ and build vector $\boldsymbol{\tau}_k = [0, ...0, 1, 0, ..., 0]^T$ where 1 is in $k^{th}$ position.
 2: Define the forward process for $k = 1, 2, ..., K$ as
        $$\mathbf{x}_k = \sqrt{1-\beta_k}\mathbf{x}_{k-1} + \sqrt{\beta_k}\boldsymbol{\epsilon}_k,$$
    where $\boldsymbol{\epsilon}_k \sim \mathcal{N}(\mathbf{0}, \mathbf{I}_d)$.
 3: for $j$ in epochs do
 4:     $k \sim \mathcal{U}\{1, 2, ..., K\}$.
 5:     Define $\boldsymbol{\tau}_k$ as in line 1.
 6:     $\epsilon_\theta(\mathbf{x}_k, k) \leftarrow (\sin(\mathbf{x}_k^T\mathbf{W} + \mathbf{b}) \odot \cos(\boldsymbol{\tau}_k^T\boldsymbol{\theta}^{(\mathbf{1})}))\boldsymbol{\theta}^{(\mathbf{2})}$.
 7:     Update $\boldsymbol{\theta} = [\boldsymbol{\theta}^{(1)}, \boldsymbol{\theta}^{(2)}]$ by minimizing the loss $L = \frac{1}{K}\sum_{k=1}^{K}\left\|\boldsymbol{\epsilon}_k - \epsilon_\theta(\mathbf{x}_k, k)\right\|_2^2$.
 8: end for

    Sampling

 9: Sample a point $\mathbf{x}_K \sim \mathcal{N}(\mathbf{0}, \mathbf{I}_d)$.
10: for $k = K-1, ..., 1$ do
11:     Sample $\boldsymbol{\epsilon} \sim \mathcal{N}(\mathbf{0}, \mathbf{I}_d)$.
12:     $\tilde{\mathbf{x}}_{k-1} = \frac{1}{\sqrt{1-\beta_k}}\left(\mathbf{x}_k - \frac{\sqrt{\beta_k}}{\sqrt{1 - \prod_{i=1}^{k}(1-\beta_i)}}\epsilon_\theta(\mathbf{x}_k, k))\right) + \beta_k\boldsymbol{\epsilon}$.
13: end for
    Output: Generated sample $\tilde{\mathbf{x}}_0$.
```

Thus, for a fixed time k, our proposed architecture is a random feature model with a fixed dictionary having N features denoted by $\phi(\langle\mathbf{x}_k, \boldsymbol{\omega}_i\rangle\rangle = \sin\left(\mathbf{x}_k^T\boldsymbol{\omega}_i + b_i\right), \forall i = 1, \ldots, N$ and learned coefficients $\mathbf{C} = (c_{ij}) \in \mathbb{R}^{N\times d}$ whose entries are $c_{ij} = \cos\left(\boldsymbol{\theta}_{ki}^{(1)}\right)\boldsymbol{\theta}_{ij}^{(2)}, \ \forall i = 1, \ldots, N$; $j = 1, \ldots, d$.

As depicted in Figure 5.1, DRFM can be visualized as K random feature models stacked up in a column. Each random feature model has associated weights corresponding to its timestep which also gets optimized implicitly while training. The reformulation of DRFM into multiple random feature models leads us to our first Lemma stated below. We show that the class of functions generated by our proposed model is the same as the class of functions approximated by random feature models.

Lemma 5.2.1. *Let $\mathcal{G}_{k,\boldsymbol{\omega}}$ denote the set of functions that can be approximated by DRFM at each timestep k defined as*

$$\mathcal{G}_{k,\boldsymbol{\omega}} = \left\{ \boldsymbol{g}(\mathbf{x}) = \sum_{j=1}^{N} \cos\left(\boldsymbol{\theta}_{kj}^{(1)}\right) \boldsymbol{\theta}_{j}^{(2)} \phi(\mathbf{x}_k^T \boldsymbol{\omega}_j) \middle| \left\| \boldsymbol{\theta}_{j}^{(2)} \right\|_\infty \leq \frac{C}{N} \right\}. \tag{5.22}$$

Then for a fixed k, $\mathcal{G}_{k,\boldsymbol{\omega}} = \mathcal{F}_{\boldsymbol{\omega}}$ where

$$\mathcal{F}_{\boldsymbol{\omega}} = \left\{ \boldsymbol{f}(\mathbf{x}) = \sum_{j=1}^{N} \boldsymbol{\alpha_j} \, \phi(\mathbf{x}^T \boldsymbol{\omega}_j) \middle| \|\boldsymbol{\alpha_j}\|_\infty \leq \frac{C}{N} \right\}. \tag{5.23}$$

Proof. The above equality can be proved easily.

Fix k. Consider $\boldsymbol{g}(\mathbf{x}) \in \mathcal{G}_{k,\boldsymbol{\omega}}$, then $\boldsymbol{g}(\mathbf{x}) = \sum_{j=1}^{N} \cos\left(\boldsymbol{\theta}_{kj}^{(1)}\right) \boldsymbol{\theta}_{j}^{(2)} \phi(\mathbf{x}_k^T \boldsymbol{\omega}_j)$. Clearly, $\boldsymbol{g}(\mathbf{x}) \in \mathcal{F}_{\boldsymbol{\omega}}$ as $\left\| \cos\left(\boldsymbol{\theta}_{kj}^{(1)}\right) \boldsymbol{\theta}_{j}^{(2)} \right\|_\infty \leq \left\| \boldsymbol{\theta}_{j}^{(2)} \right\|_\infty \leq \frac{C}{N}$. Thus $\mathcal{G}_{k,\boldsymbol{\omega}} \subseteq \mathcal{F}_{\boldsymbol{\omega}}$.

Conversely let $\boldsymbol{f}(\mathbf{x}) \in \mathcal{F}_{\boldsymbol{\omega}}$, then $\boldsymbol{f}(\mathbf{x}) = \sum_{j=1}^{N} \boldsymbol{\alpha_j} \, \phi(\mathbf{x}_k^T \boldsymbol{\omega}_j)$. Choose $\boldsymbol{\theta}_{j}^{(2)} = \boldsymbol{\alpha}_j$ and $\boldsymbol{\theta}_{kj}^{(1)} = [0, \cdots, 0]$.

As $\cos\left(\boldsymbol{\theta}_{kj}^{(1)}\right) \boldsymbol{\theta}_{j}^{(2)} = \boldsymbol{\alpha}_j$, thus $\boldsymbol{f}(\mathbf{x}) = \sum_{j=1}^{N} \cos\left(\boldsymbol{\theta}_{kj}^{(1)}\right) \boldsymbol{\theta}_{j}^{(2)} \phi(\mathbf{x}^T \boldsymbol{\omega}_j)$ and $\|\boldsymbol{\theta}_{j}^{(2)}\|_\infty = \|\boldsymbol{\alpha}_j\|_\infty \leq \frac{C}{N}$. Hence $\boldsymbol{f}(\mathbf{x}) \in \mathcal{G}(k, \boldsymbol{\omega})$ and $\mathcal{F}_{\boldsymbol{\omega}} \subseteq \mathcal{G}_{k,\boldsymbol{\omega}}$. □

In the next Lemma stated below, we extend results from [109–111] to find approximation error bounds for vector-valued functions.

Lemma 5.2.2. *Let $X \subset \mathbb{R}^d$ denote the training dataset and suppose q is a measure on X, and $\boldsymbol{f}^\star$ a function in $\mathcal{F}_\rho$ where*

$$\mathcal{F}_\rho = \left\{ \boldsymbol{f}(\mathbf{x}) = \int_\Omega \boldsymbol{\alpha}(\boldsymbol{\omega}) \phi(\mathbf{x}; \boldsymbol{\omega}) \, d\boldsymbol{\omega} \;\middle|\; \|\boldsymbol{\alpha}(\boldsymbol{\omega})\|_\infty \leq C\rho(\boldsymbol{\omega}) \right\}.$$

If $[\boldsymbol{\omega}_j]_{j\in[N]}$ are drawn iid from ρ, then for $\delta > 0$, with probability at least $1-\delta$ over $[\boldsymbol{\omega}_j]_{j\in[N]}$, there exists a function $\boldsymbol{f}^\sharp \in \mathcal{F}_{\boldsymbol{\omega}}$ such that

$$\|\boldsymbol{f}^\sharp - \boldsymbol{f}^\star\|_2 \leq \frac{C\sqrt{d}}{\sqrt{N}} \left(1 + \sqrt{2 \log \frac{1}{\delta}} \right), \tag{5.24}$$

where $\mathcal{F}_{\boldsymbol{\omega}}$ is defined in Eq. (5.23).

Proof. We follow the proof technique described in [111]. As $\boldsymbol{f}^\star \in \mathcal{F}_\rho$, then $\boldsymbol{f}^\star(\mathbf{x}) = \int_{\boldsymbol{\Omega}} \boldsymbol{\alpha}(\boldsymbol{\omega})\phi(\mathbf{x};\boldsymbol{\omega})d\boldsymbol{\omega}$. Construct $\boldsymbol{f}_k = \boldsymbol{\beta}_k\phi(\cdot;\boldsymbol{\omega}_k)$, $k = 1,\cdots,N$ such that $\boldsymbol{\beta}_k = \frac{\boldsymbol{\alpha}(\boldsymbol{\omega}_k)}{\rho(\boldsymbol{\omega}_k)} = \frac{1}{\rho(\boldsymbol{\omega}_k)}\begin{bmatrix}\alpha_1(\boldsymbol{\omega}_k)\\ \vdots \\ \alpha_d(\boldsymbol{\omega}_k)\end{bmatrix}$.

Note that $\mathbb{E}_{\boldsymbol{\omega}}(\boldsymbol{f}_k) = \int_{\boldsymbol{\omega}} \frac{\boldsymbol{\alpha}(\boldsymbol{\omega}_k)}{\rho(\boldsymbol{\omega}_k)}\phi(\mathbf{x};\boldsymbol{\omega})\rho(\boldsymbol{\omega}_k)d\boldsymbol{\omega} = \boldsymbol{f}^\star$.

Define the sample average of these functions as $\boldsymbol{f}^\sharp(\mathbf{x}) = \sum_{k=1}^{N}\frac{\boldsymbol{\beta}_k}{N}\phi(\mathbf{x};\boldsymbol{\omega}_k)$.

As $\left\|\frac{\boldsymbol{\beta}_k}{N}\right\|_\infty \le \frac{C}{N}$, thus $\boldsymbol{f}^\sharp \in \mathcal{F}_{\boldsymbol{\omega}}$. Also note that $\|\boldsymbol{\beta}_k\phi(\cdot;\boldsymbol{\omega}_k)\|_2 \le \sqrt{d}\|\|\boldsymbol{\beta}_k\phi(\cdot;\boldsymbol{\omega}_k)\|_\infty \le \sqrt{d}C$.

In order to get the desired result, we use McDiarmid's inequality. Define a scaler function on $F = \{\boldsymbol{f}_1,\cdots,\boldsymbol{f}_N\}$ as $g(F) = \|\boldsymbol{f}^\sharp - \mathbb{E}_F\boldsymbol{f}^\sharp\|_2$. We claim that the function g is stable under perturbation of its ith argument.

Define $\tilde{F} = \{\boldsymbol{f}_1,\cdots,\tilde{\boldsymbol{f}}_i,\cdots,\boldsymbol{f}_N\}$ i.e., $\tilde{F}$ differs from F only in its ith element. Then

$$\left|g(F) - g(\tilde{F})\right| = \left|\left\|\boldsymbol{f}^\sharp - \mathbb{E}_F\boldsymbol{f}^\sharp\right\|_2 - \left\|\tilde{\boldsymbol{f}}^\sharp - \mathbb{E}_{\tilde{F}}\boldsymbol{f}^\sharp\right\|_2\right| \le \left\|\boldsymbol{f}^\sharp - \tilde{\boldsymbol{f}}^\sharp\right\|_2, \tag{5.25}$$

where the above inequality is obtained from triangle inequality. Further,

$$\left\|\boldsymbol{f}^\sharp - \tilde{\boldsymbol{f}}^\sharp\right\|_2 = \frac{1}{N}\left\|\boldsymbol{f}_i - \tilde{\boldsymbol{f}}_i\right\|_2 \le \left\|(\boldsymbol{\beta}_i - \tilde{\boldsymbol{\beta}}_i)\phi(\cdot;\boldsymbol{\omega})\right\|_2 \le \frac{\sqrt{d}C}{N}. \tag{5.26}$$

Thus $\mathbb{E}[g(F)^2] = \mathbb{E}\left[\left\|\boldsymbol{f}^\sharp - \mathbb{E}_F\boldsymbol{f}^\sharp\right\|_2^2\right] = \frac{1}{N}\left[\mathbb{E}\left[\left\|\sum_{k=1}^{N}\boldsymbol{f}_k\right\|_2^2\right] - \left\|\mathbb{E}\left[\sum_{k=1}^{N}\boldsymbol{f}_k\right]\right\|_2^2\right]$.

Since $\|\boldsymbol{f}_k\|_2 \le \sqrt{d}C$, using Jensen's inequality and above result we get

$$\mathbb{E}[g(F)] \le \sqrt{\mathbb{E}(g^2(F))} \le \frac{\sqrt{d}C}{\sqrt{N}}. \tag{5.27}$$

Finally, the required bounds can be obtained by combining the above result and McDiarmid's inequality. □

Using the above-stated Lemmas and results given in [35], we derive our main theorem. Specifically, we quantify the total variation between the distribution learned by our model and the true data distribution.

Theorem 5.2.3. *For a given probability density $q(\mathbf{x}_0)$ on $\mathbb{R}^d$ suppose the following conditions hold:*

1. *For all $t \geq 0$, the score $\nabla \log q(\mathbf{x}(t))$ is $L-$Lipschitz.*

2. *For some $\eta > 0$, $\mathbb{E}_{q(\mathbf{x}_0)}[\|.\|^{(2+\eta)}]$ is finite. Denote $m_2^2 = \mathbb{E}_{q(\mathbf{x}_0)}[\|.\|^2]$.*

Let p_θ denote learnt distribution corresponding to the DRFM trained parameterized model ϵ_θ. Let $p_\theta(\mathbf{x}_0)$ be the distribution of the samples generated by DRFM after K timesteps at terminal time T. Then for the SDE formulation of the DDPM algorithm, if the step size $h := T/K$ satisfies $h \lesssim 1/L$, where $L \geq 1$. Then,

$$TV(p_\theta(\mathbf{x}_0), q(\mathbf{x}_0)) \lesssim \sqrt{KL(q(\mathbf{x}_0)\|\gamma)}\exp(-T) + (L\sqrt{dh} + Lm_2 h)\sqrt{T} + \frac{C_2\sqrt{TKd}}{\sqrt{N}}\left(1 + \sqrt{2\log\frac{1}{\delta}}\right), \tag{5.28}$$

where $C_2 \geq \max_{1\leq i\leq N, 1\leq j\leq d}\left|\boldsymbol{\theta}_{ij}^{(2)}\right|$ and γ is the p.d.f. of the multivariate normal distribution with mean vector $\mathbf{0}$ and covariance matrix $\mathbf{I}_d$.

Proof. Suppose the error in score estimate is bounded in L^2, i.e.,

$$E_{q(\mathbf{x}(t))}[\|p_\theta(\mathbf{x}(t)) - \nabla \ln q(\mathbf{x}(t))\|^2] \leq \varepsilon_{score}^2.$$

Then from Theorem 1 in [35], we get

$$TV(p_\theta(\mathbf{x}_0), q(\mathbf{x}_0)) \lesssim \sqrt{KL(q(\mathbf{x}_0)\|\gamma)}\exp(-T) + (L\sqrt{dh} + Lm_2 h)\sqrt{T} + \sqrt{T}\varepsilon_{score}. \tag{5.29}$$

Using Lemma 2.3.7, we see that the class of functions approximated by DRFM is the same as that of RFMs. Thus using Lemma 5.2.2 and substituting $\varepsilon_{score} = \frac{C_2\sqrt{Kd}}{\sqrt{N}}\left(1 + \sqrt{2\log\frac{1}{\delta}}\right)$ such that $C_2 \geq \max_{1\leq i\leq N, 1\leq j\leq d}\left|\boldsymbol{\theta}_{ij}^{(2)}\right|$ we get the desired results.

□

The error bound given in Eq. (5.28) consists of three types of errors: (i) convergence error of the forward process; (ii) discretization error of the associated SDE with step size $h > 0$; and (iii) score estimation error, respectively. Note that the first two terms are independent of the model architecture. While for most models score estimation is difficult, our main contribution involves quantifying that error which gives us an estimate of the number of parameters needed for the third error to become negligible. We can combine the proof of Lemma 5.2.1, 5.2.2, and Theorem 1 from [35] to get the required bounds.

5.3 Experimental Results

In order to validate our findings, we train our proposed model on both image and audio data. We evaluate the validity of our model on two tasks: (i) generating data from noise and; (ii) denoising data corrupted with Gaussian noise. The experiments on images were done by taking one hundred images of the class "dress" and "shoes" separately from the fMNIST dataset. For audio data, we use two music samples corresponding to the flute and guitar. We compare our method with a fully connected version of DRFM (denoted by NN) where we train $[\mathbf{W}, \mathbf{b}, \boldsymbol{\theta}^{(1)}, \boldsymbol{\theta}^{(2)}]$ while preserving the number of trainable parameters, a U-Net model and a classical random feature approach with only $\boldsymbol{\theta}^{(2)}$ being trainable (denoted by RF). The details of the implementation of all the experiments and their results are described in the sections below.

5.3.1 The U-Net Architecture Used for Comparison

We use a simple U-Net architecture with the same number of trainable layers as that of DRFM. The representation is given below. Note that a positional time encoding layer is added before each downsampling step.

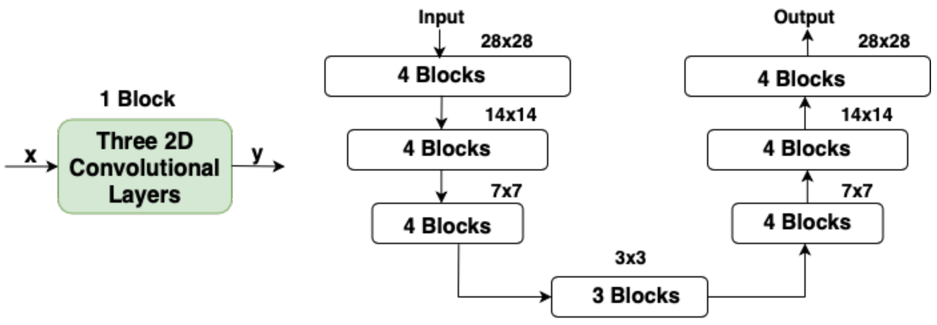


Figure 5.2: U-Net architecture used in the experiments. **Left:** One block consisting of three convolutional layers. **Right:** U-Net architecture with downsampling and upsampling layers.

5.3.2 Results on Fashion-MNIST (fMNIST) Data

We create the image dataset for our experiments by considering 100 images of size 28×28 taken from a particular class of fMNIST dataset. DRFM is trained with 80000 random features. For NN, U-Net, and RF we adjust the number of trainable parameters to match that of DRFM. We use 100 equally spaced timesteps for the forward diffusion process between 10^{-4} and 0.02 and train for 30000 epochs by minimizing the mean squared error (MSE) loss using ADAM optimizer with a learning rate of $\eta = 0.001$. For the task of generating images, we generated fifteen samples from randomly sampled noise. In our second task we give fifteen images from the same class (but not in the training set) corrupted with noise and test if our model can denoise the image.

Results from Figure 5.3 demonstrate that our method learns the overall features of the input distribution. Although the model is trained with a small size of training data (only one hundred images) and timesteps (one hundred timesteps), we can see that the samples generated from pure noise have already started to resemble the input distribution of dresses. The samples generated by the U-Net model also simply learn to generate the

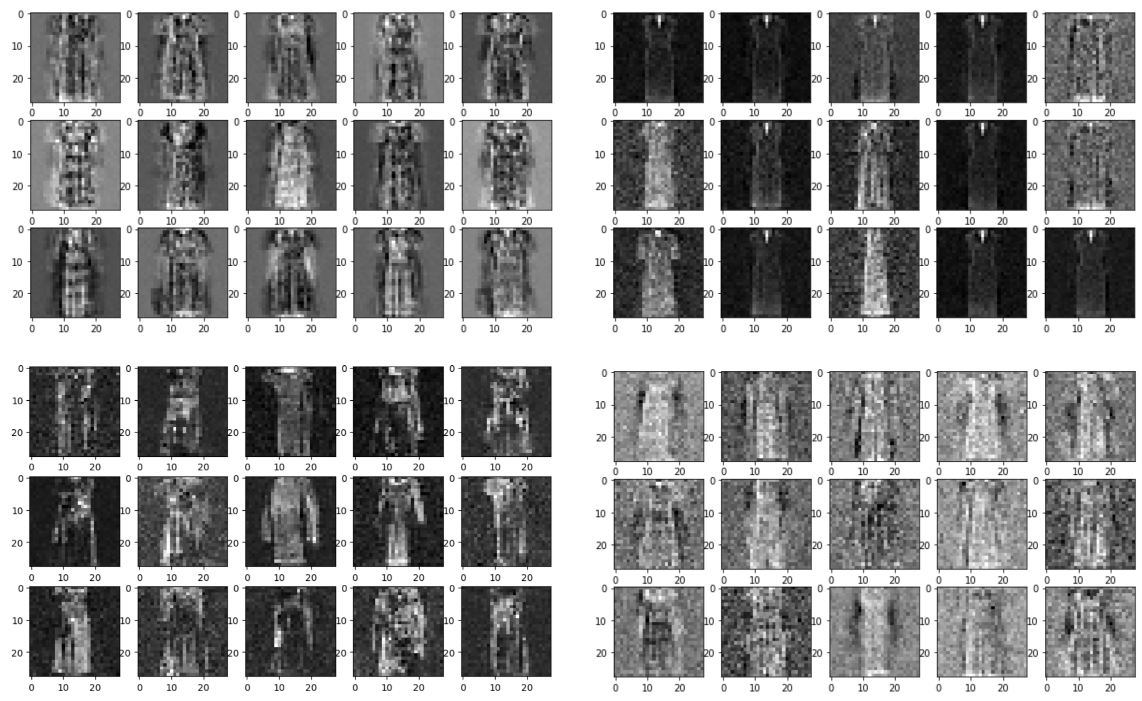


Figure 5.3: Figures generated from random noise when trained on 100 "dress" images with 100 timesteps linearly spaced between 10^{-4} and 0.02. The models were trained using 30000 epochs using ADAM optimizer with a learning rate $\eta = 0.001$ by minimizing the MSE loss. **Top left block:** DRFM. **Top right block:** NN. **Bottom left block:** U-Net. **Bottom right block:** RF.

overall features of the distribution. The overall sample quality is not significantly better than the ones generated by DRFM. For NN model we note that most of the generated samples are the same with a dark shadow while for RF model, the generated images are very noisy and barely recognizable.

We also test our model's ability to remove noise from images. We take fifteen random images of "dress" not in the training data and corrupt it with 20% noise. The proposed model is used for denoising. In Figure 5.4 we can see that the model can recover a denoised image which is in fact better than the results obtained when sampling from pure noise. The U-Net and NN models perform quite well for most of the images. However, in a few cases with the NN model, it fails to denoise anything and the final image remains noisy. RF model fails to denoise and the resultant images still contain noise.

Table 5.1 gives the Fréchet Inception Distance (FID) calculated using a batch of 50 images from the training dataset and 15 of the generated images by each of the models. Note that the scores values given are for the sake of comparing the four methods and more

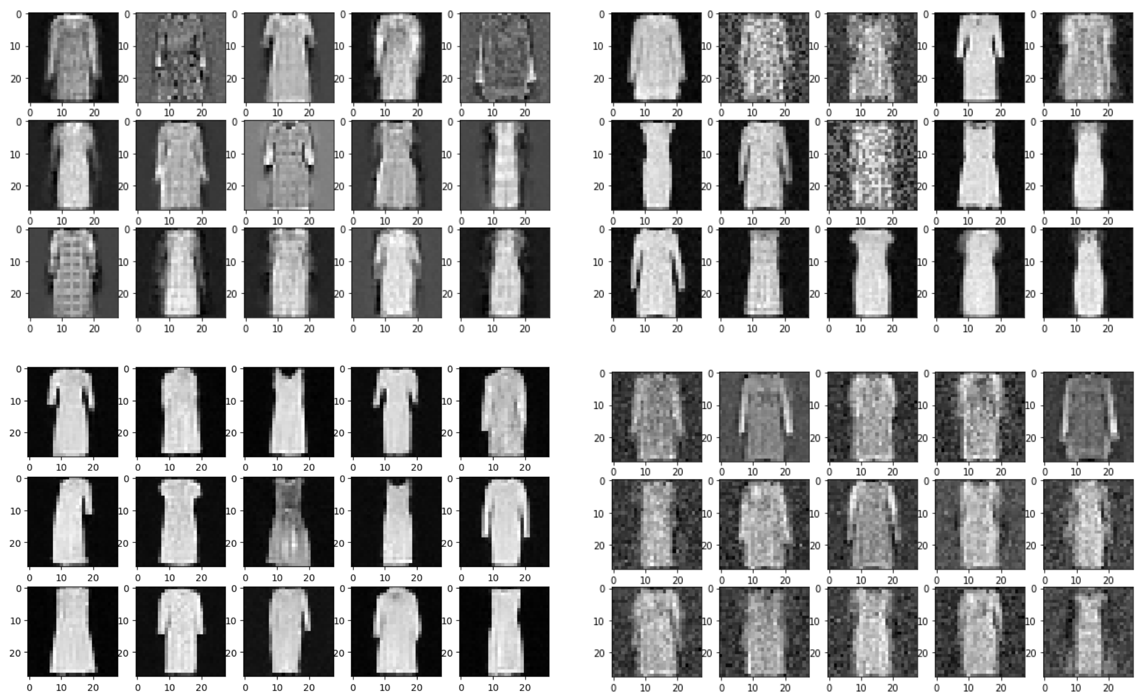


Figure 5.4: Figures denoised from images corrupted with 20 % noise.We use 100 timesteps linearly spaced between 10^{-4} and 0.02. The models were trained using 30000 epochs using ADAM optimizer with a learning rate $\eta = 0.001$ by minimizing the MSE loss. **Top left block:** DRFM. **Top right block:** NN. **Bottom left block:** U-Net. **Bottom right block:** RF.

samples may improve the FID score. We see that for the generative task, the proposed DRFM architecture gives the lowest scores. The more commonly used U-Net model is in the third position after the NN model. For the denoising task, we see that the NN gives the best results, (although some of the images are inconsistently noisy), followed by U-Net and then DRFM. Note that for U-Net, while most images are denoised perfectly, some images are incompletely formed which leads to a lower FID score. Moreover, we see that all the FID scores are also consistent with the qualitative assessment made in Figures 5.3 and 5.4.

Model	Timesteps	FID Score
DRFM	100	**453.87**
U-Net	100	463.28
NN	100	457.21
RF	100	470.12

Model	Timesteps	FID Score
DRFM	100	394.99
U-Net	100	388.38
NN	100	**378.18**
RF	100	450.23

Table 5.1: FID scores for all the models when trained with 100 equally spaced timesteps between 10^{-4} and 0.02. **Right:** FID scores for generative task. **Left:** FID scores for denoising task.

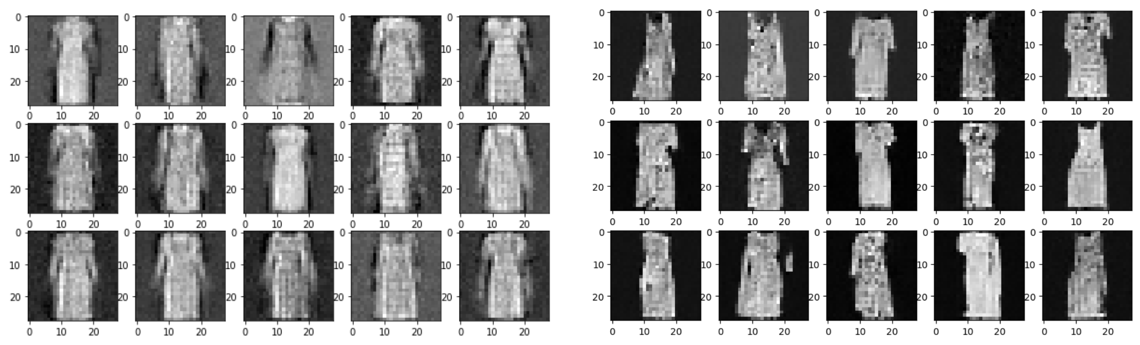


Figure 5.5: Generated images trained on 100 “dress” images with 1000 timesteps. **Left block:** DRFM. **Right block:** U-Net.

In order to check the effect of the number of timesteps on the sampling power of DRFM, we also run our model using 1000 timesteps between 10^{-4} and 0.02. The images generated are given in Figure 5.5. Samples generated from noise seem to improve with the increase in the number of timesteps for both DRFM and U-Net. However, for the denoising task, the results are similar to those with 100 timesteps as observed in Figure 5.6. The improvement in sample quality with an increased number of timesteps for generating data is expected as more reverse steps would be required to generate a point in the input distribution.

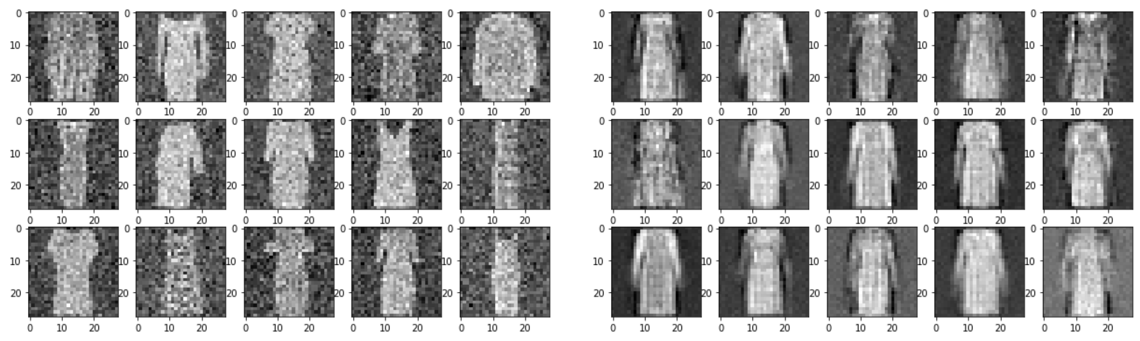


Figure 5.6: Denoised images with DRFM trained on 100 “dress” images with 1000 timesteps. **Left block:** Noisy sample. **Right block:** Denoised image by DRFM.

We also conducted an experiment with a different class of data with the same experimental setup as above with 100 timesteps. This time we select the class “shoes” and test our model’s performance. The conclusions drawn support the claims we made with the previous class of data where DRFM can denoise images well while only learning to generate basic features of the model class when generating from pure noise. The plots for the above are depicted in Figure 5.7.

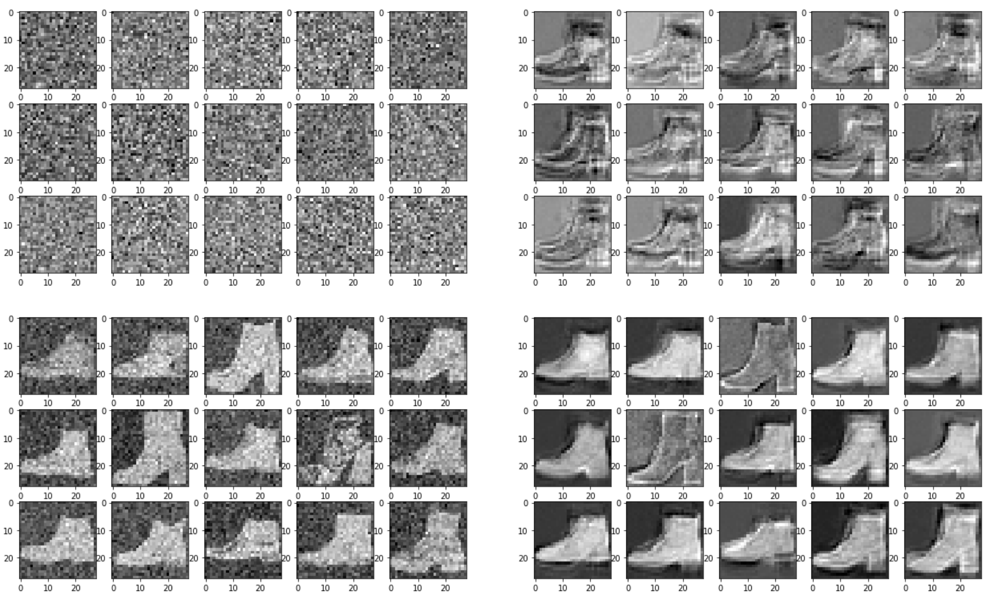
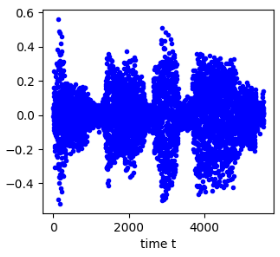


Figure 5.7: Figures generated from random noise when trained on 100 "shoes" images with 100 timesteps linearly spaced between 10^{-4} and 0.02. The models were trained using 30000 epochs using ADAM optimizer with a learning rate $\eta = 0.001$ by minimizing the MSE loss. **Top left block:** Gaussian noise. **Top right block:** DRFM. **Bottom left block:** Noisy samples. **Bottom right block:** Denoised image by DRFM.

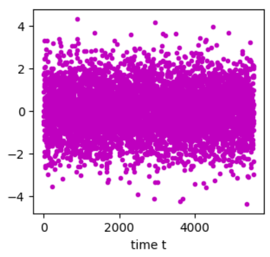
5.3.3 Results on Audio Data

Our second experiment involves learning to generate and denoise audio signals. We use one data sample each taken from two different instruments, namely guitar and flute. There are a total of 5560 points for each instrument piece. We train our model using 15000 random features with 100 timesteps taken between 10^{-4} and 0.02 for 30000 epochs. The samples are generated from pure noise using the trained model to remove noise for each reverse diffusion step. We also test if our model is capable of denoising a signal when it is not a part of the training set explicitly (but similar to it). For that, we use a validation data point containing samples from a music piece when both guitar and flute are played simultaneously. We plot the samples generated from pure Gaussian noise in Figure 5.8. The plots in Figure 5.8 demonstrate the potential of DRFM to generate signals not a part of the original training data. Figure 5.8(b) shows that there is no advantage of using a NN model since the results are much worse. The network does not learn anything and the signal generated is just another noise. On the other hand, our proposed model DRFM generates signals that are similar to the original two signals used to train the data.

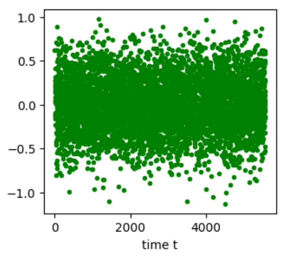
Figure 5.9 shows that when our trained model is applied to a validation data point, it can successfully recover and denoise the signal. This is more evident when the signal is played as an audio file. There are however some extra elements that get added while recovering due to the presence of noise which is a common effect of using diffusion models



(a) DRFM



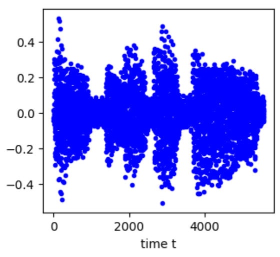
(b) NN



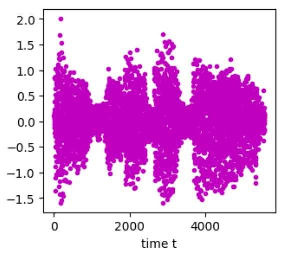
(c) RF

Figure 5.8: Generated samples using different models.

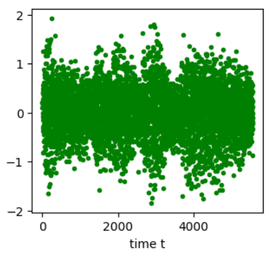
for denoising.



(a) DRFM



(b) NN



(c) RF

Figure 5.9: Denoised signals using different models.

5.4 Summary

We propose a model based on diffusion models and random feature methods. We show that our method of formulating the model helps us to derive bounds on the sampled data. Our experiments indicate that the proposed model can learn to generate as well as denoise data from a small training dataset.

Chapter 6

Conclusions

6.1 Summary and Conclusions

In this thesis, we explored the theory and various applications of random feature models which are beyond their conventional applications in the literature. Applications of RFMs in data science are still limited. We start our research work with a fundamental application of surrogate modeling i.e., function approximation. The Pareto principle states that a small number of low-complexity interactions dominates most real-world systems. Thus, for a given dataset we can find a sparse representation from a large set of features. In particular, our goal was to build a model for high dimensional sparse additive functions. We proposed a new high-dimensional sparse additive model called HARFE utilizing the random feature method with two sparsity priors. First, we assumed that the number of terms needed in the model is small which leads to function approximations with low model complexity. Second, we enforce a random and sparse connectivity pattern between the hidden layer and the input layer which helps to extract input variable dependencies. Based on the numerical experiments on high-dimensional synthetic examples, the Friedman functions, and real data, the HARFE algorithm was shown to produce robust results that have the added benefit of extracting interpretable variable information. The analysis of the HARFE algorithm utilizes techniques from compressive sensing and it was shown that the method converges and has a reasonable error bound depending on the number of features, the number of samples, the ridge parameter, the sparsity, the noise level, and dimensional parameters.

Following the success of our proposed model for high-dimensional sparse additive function approximation, we extended the idea of function approximation to learning dynamical systems from epidemiology for short-term predictions using incomplete data. We particularly looked into short-term forecasts of an epidemic-based trajectory as they serve as a communication channel between the scientific community and decision-makers. It helps in making informed short-term decisions such as the allocation of medical supplies, health-care staffing needs, lockdowns, and closures. We proposed a delay embedding-based model called SPADE4 which forecasts the trajectory (daily active cases or cumulative cases) in a one-week-forecast horizon. We used the given incomplete data and embedded it into a higher dimensional space by using delay time embedding. Using Takens' theorem, we assumed that the new representation was diffeomorphic to the original unknown system and used it to learn the trajectory of the rate of change of infected people using a random feature basis. We compared SPADE4 with benchmark methods with SEIR, SEIR$_{\beta(t)}$, and SμEIR fitting. Our experiments on simulated and real datasets show that SPADE4 can provide a reliable and comparatively accurate forecast that exceeds the performance of the benchmark method. While some knowledge of the target model and initial values are required by the benchmark model, SPADE4 has no such prerequisites for its performance. Although the proposed algorithm requires choosing appropriate hyperparameters for its claimed performance, using existing theoretical results on delay embedding and sparse random feature models, we show that the choice of hyperparameters made is reasonable for its performance. Additionally, we also provided results to show that the predictions given by SPADE4 are stable with respect to the basis chosen as well as our choice of the embedding dimension p. For a fair comparison, we also consider alternative function approximation models such as a (orthogonal) polynomial fitting and a fully connected shallow neural network approximation. We conclude from our results that SPADE4 can perform comparably to the best method for each dataset with the lowest validation errors. Some possible limitations of the method were also discussed. For example, the performance of SPADE4 is partially dependent on an accurate estimation of the derivative and so if the data is extremely noisy, the model predictions may be inaccurate. For such situations, one may need to preprocess the data using appropriate denoising techniques before applying SPADE4.

Working with dynamical systems motivated us to work with random features that are dependent on time. This way of formulating the model would be the starting point for building interpretable random features for learning distributions. Thus we proposed the Diffusion Random Feature Model (DRFM). Optimization of the coefficients in DRFM happens corresponding to the output layer as well as the time parameter. Thus for each timestep, there is a weighted random feature model getting optimized with the weights representing the intermediate diffusion timesteps. Since the architecture is interpretable from existing results, it is possible to find theoretical upper bounds on the samples generated by DRFM with respect to the input distribution. We validated our findings using numerical experiments on audio and a small subset of the Fashion-MNIST dataset. Our findings indicated the power of our method to learn the process of generating data from as few as one hundred training samples and one hundred timesteps. Comparisons with a fully connected network (when all layers are trainable) and random features method (all but the last layer is fixed i.e., only $\boldsymbol{\theta}_2$ is trainable) highlighted the advantages of our model which performed better than both.

6.2 Further Directions

The research conducted in the thesis opens up a wide options worthy of further exploration. In our work on predicting epidemics, our focus was on short-term prediction of diseases using either data from daily infected cases or cumulative cases. For the input data coming from daily infected cases, we analyzed the disease one wave at a time. An interesting future work would be to study seasonal diseases such as chicken pox which has oscillating data with short periods. We would expect a successful method to be able to predict the oscillating nature of the disease with accurate predictions for the peak for a few seasons. It would also be a worthy exploration to understand the high-dimensional diffeomorphic system that is assumed to be recovered from embeddings of the input data along with a guided method to choose the embedding dimension and time delay parameter.

Since the research work on DRFM was based on time-dependent random features, we think it would be interesting to apply the proposed architecture for learning dynamical systems. Since our proposed model preserves the input and output dimensions, it can be used to learn the full state information of high-dimensional systems. DRFM might also be a good model to use for learning stochastic differential equations as the states are not deterministic and thus allocating an RFM for each timestep might be beneficial.

Letters of Copyright Permission

SPRINGER NATURE LICENSE TERMS AND CONDITIONS

Dec 07, 2023

This Agreement between Ms. Esha Saha ("You") and Springer Nature ("Springer Nature") consists of your license details and the terms and conditions provided by Springer Nature and Copyright Clearance Center.

License Number	5641380552030
License date	Oct 03, 2023
Licensed Content Publisher	Springer Nature
Licensed Content Publication	Sampling Theory, Signal Processing, & Data Analysis
Licensed Content Title	HARFE: hard-ridge random feature expansion
Licensed Content Author	Esha Saha et al
Licensed Content Date	Aug 9, 2023
Type of Use	Thesis/Dissertation
Requestor type	academic/university or research institute
Format	print and electronic
Portion	full article/chapter
Will you be translating?	no
Circulation/distribution	2000 - 4999
Author of this Springer Nature content	yes
Title	Expanding the Scope of Random Feature Models: Theory and Applications
Institution name	University of Waterloo
Expected presentation date	Oct 2023
Requestor Location	Ms. Esha Saha 216 Residence Road Waterloo, ON N2L6P6 Canada Attn: Ms. Esha Saha
Total	**0.00 CAD**

Terms and Conditions

4. 1. An alternative scope of license may apply to signatories of the STM Permissions Guidelines ("STM PG") as amended from time to time and made available at https://www.stm-assoc.org/intellectual-property/permissions/permissions-guidelines/.

4. 2. For content reuse requests that qualify for permission under the STM PG, and which may be updated from time to time, the STM PG supersede the terms and conditions contained in this License.

4. 3. If a License has been granted under the STM PG, but the STM PG no longer apply at the time of publication, further permission must be sought from the Rightsholder. Contact journalpermissions@springernature.com or bookpermissions@springernature.com for these rights.

5. Duration of License

5. 1. Unless otherwise indicated on your License, a License is valid from the date of purchase ("License Date") until the end of the relevant period in the below table:

Reuse in a medical communications project	Reuse up to distribution or time period indicated in License
Reuse in a dissertation/thesis	Lifetime of thesis
Reuse in a journal/magazine	Lifetime of journal/magazine
Reuse in a book/textbook	Lifetime of edition
Reuse on a website	1 year unless otherwise specified in the License
Reuse in a presentation/slide kit/poster	Lifetime of presentation/slide kit/poster. Note: publication whether electronic or in print of presentation/slide kit/poster may require further permission.
Reuse in conference proceedings	Lifetime of conference proceedings
Reuse in an annual report	Lifetime of annual report
Reuse in training/CME materials	Reuse up to distribution or time period indicated in License
Reuse in newsmedia	Lifetime of newsmedia
Reuse in coursepack/classroom materials	Reuse up to distribution and/or time period indicated in license

6. Acknowledgement

6. 1. The Licensor's permission must be acknowledged next to the Licensed Material in print. In electronic form, this acknowledgement must be visible at the same time as the figures/tables/illustrations or abstract and must be hyperlinked to the journal/book's homepage.

6. 2. Acknowledgement may be provided according to any standard referencing system and at a minimum should include "Author, Article/Book Title, Journal name/Book imprint, volume, page number, year, Springer Nature".

7. Reuse in a dissertation or thesis

SPRINGER NATURE LICENSE TERMS AND CONDITIONS

Dec 07, 2023

This Agreement between Ms. Esha Saha ("You") and Springer Nature ("Springer Nature") consists of your license details and the terms and conditions provided by Springer Nature and Copyright Clearance Center.

License Number	5641380357799
License date	Oct 03, 2023
Licensed Content Publisher	Springer Nature
Licensed Content Publication	Bulletin of Mathematical Biology
Licensed Content Title	SPADE4: Sparsity and Delay Embedding Based Forecasting of Epidemics
Licensed Content Author	Esha Saha et al
Licensed Content Date	Jun 19, 2023
Type of Use	Thesis/Dissertation
Requestor type	academic/university or research institute
Format	print and electronic
Portion	full article/chapter
Will you be translating?	no
Circulation/distribution	2000 - 4999
Author of this Springer Nature content	yes
Title	Expanding the Scope of Random Feature Models: Theory and Applications
Institution name	University of Waterloo
Expected presentation date	Oct 2023
Requestor Location	Ms. Esha Saha 216 Residence Road Waterloo, ON N2L6P6 Canada Attn: Ms. Esha Saha
Total	**0.00 CAD**
Terms and Conditions	

1. 1. The Licensor grants you a personal, non-exclusive, non-transferable, non-sublicensable, revocable, world-wide License to reproduce, distribute, communicate to the public, make available, broadcast, electronically transmit or create derivative works using the Licensed Material for the purpose(s) specified in your RightsLink Licence Details only. Licenses are granted for the specific use requested in the order and for no other use, subject to these Terms and Conditions. You acknowledge and agree that the rights granted to you under this License do not include the right to modify, edit, translate, include in collective works, or create derivative works of the Licensed Material in whole or in part unless expressly stated in your RightsLink Licence Details. You may use the Licensed Material only as permitted under this Agreement and will not reproduce, distribute, display, perform, or otherwise use or exploit any Licensed Material in any way, in whole or in part, except as expressly permitted by this License.

1. 2. You may only use the Licensed Content in the manner and to the extent permitted by these Terms and Conditions, by your RightsLink Licence Details and by any applicable laws.

1. 3. A separate license may be required for any additional use of the Licensed Material, e.g. where a license has been purchased for print use only, separate permission must be obtained for electronic re-use. Similarly, a License is only valid in the language selected and does not apply for editions in other languages unless additional translation rights have been granted separately in the License.

1. 4. Any content within the Licensed Material that is owned by third parties is expressly excluded from the License.

1. 5. Rights for additional reuses such as custom editions, computer/mobile applications, film or TV reuses and/or any other derivative rights requests require additional permission and may be subject to an additional fee. Please apply to journalpermissions@springernature.com or bookpermissions@springernature.com for these rights.

2. Reservation of Rights

Licensor reserves all rights not expressly granted to you under this License. You acknowledge and agree that nothing in this License limits or restricts Licensor's rights in or use of the Licensed Material in any way. Neither this License, nor any act, omission, or statement by Licensor or you, conveys any ownership right to you in any Licensed Material, or to any element or portion thereof. As between Licensor and you, Licensor owns and retains all right, title, and interest in and to the Licensed Material subject to the license granted in Section 1.1. Your permission to use the Licensed Material is expressly conditioned on you not impairing Licensor's or the applicable copyright owner's rights in the Licensed Material in any way.

3. Restrictions on use

3. 1. Minor editing privileges are allowed for adaptations for stylistic purposes or formatting purposes provided such alterations do not alter the original meaning or intention of the Licensed Material and the new figure(s) are still accurate and representative of the Licensed Material. Any other changes including but not limited to, cropping, adapting, and/or omitting material that affect the meaning, intention or moral rights of the author(s) are strictly prohibited.

3. 2. You must not use any Licensed Material as part of any design or trademark.

3. 3. Licensed Material may be used in Open Access Publications (OAP), but any such reuse must include a clear acknowledgment of this permission visible at the same time as the figures/tables/illustration or abstract and which must indicate that the Licensed Material is not part of the governing OA license but has been reproduced with permission. This may be indicated according to any standard referencing system but must include at a minimum 'Book/Journal title, Author, Journal Name (if applicable), Volume (if applicable), Publisher, Year, reproduced with permission from SNCSC'.

4. STM Permission Guidelines

4. 1. An alternative scope of license may apply to signatories of the STM Permissions Guidelines ("STM PG") as amended from time to time and made available at https://www.stm-assoc.org/intellectual-property/permissions/permissions-guidelines/.

4. 2. For content reuse requests that qualify for permission under the STM PG, and which may be updated from time to time, the STM PG supersede the terms and conditions contained in this License.

4. 3. If a License has been granted under the STM PG, but the STM PG no longer apply at the time of publication, further permission must be sought from the Rightsholder. Contact journalpermissions@springernature.com or bookpermissions@springernature.com for these rights.

5. Duration of License

5. 1. Unless otherwise indicated on your License, a License is valid from the date of purchase ("License Date") until the end of the relevant period in the below table:

Reuse in a medical communications project	Reuse up to distribution or time period indicated in License
Reuse in a dissertation/thesis	Lifetime of thesis
Reuse in a journal/magazine	Lifetime of journal/magazine
Reuse in a book/textbook	Lifetime of edition
Reuse on a website	1 year unless otherwise specified in the License
Reuse in a presentation/slide kit/poster	Lifetime of presentation/slide kit/poster. Note: publication whether electronic or in print of presentation/slide kit/poster may require further permission.
Reuse in conference proceedings	Lifetime of conference proceedings
Reuse in an annual report	Lifetime of annual report
Reuse in training/CME materials	Reuse up to distribution or time period indicated in License
Reuse in newsmedia	Lifetime of newsmedia
Reuse in coursepack/classroom materials	Reuse up to distribution and/or time period indicated in license

6. Acknowledgement

6. 1. The Licensor's permission must be acknowledged next to the Licensed Material in print. In electronic form, this acknowledgement must be visible at the same time as the figures/tables/illustrations or abstract and must be hyperlinked to the journal/book's homepage.

6. 2. Acknowledgement may be provided according to any standard referencing system and at a minimum should include "Author, Article/Book Title, Journal name/Book imprint, volume, page number, year, Springer Nature".

7. Reuse in a dissertation or thesis

References

[1] Ben Adcock, Simone Brugiapaglia, and Clayton G Webster. Compressed sensing approaches for polynomial approximation of high-dimensional functions. In *Compressed Sensing and Its Applications: Second International MATHEON Conference 2015*, pages 93–124. Springer, 2017.

[2] Ben Adcock, Anders C Hansen, and Bogdan Roman. A note on compressed sensing of structured sparse wavelet coefficients from subsampled Fourier measurements. *IEEE signal processing letters*, 23(5):732–736, 2016.

[3] Amir Ali Ahmadi and Bachir El Khadir. Learning dynamical systems with side information. In *Learning for Dynamics and Control*, pages 718–727. PMLR, 2020.

[4] Christian L Althaus. Estimating the reproduction number of Ebola virus (EBOV) during the 2014 outbreak in West Africa. *PLoS currents*, 6, 2014.

[5] J Arino and P Van Den Driessche. Time delays in epidemic models. *Delay Differential Equations and Applications, NATO Science Series*, 205:539–578, 2006.

[6] Ibrahim Ayed, Emmanuel de Bézenac, Arthur Pajot, Julien Brajard, and Patrick Gallinari. Learning dynamical systems from partial observations. *arXiv preprint arXiv:1902.11136*, 2019.

[7] Francis Bach. On the equivalence between kernel quadrature rules and random feature expansions. *The Journal of Machine Learning Research*, 18(1):714–751, 2017.

[8] Peter L Bartlett, Philip M Long, Gábor Lugosi, and Alexander Tsigler. Benign overfitting in linear regression. *Proceedings of the National Academy of Sciences*, 117(48):30063–30070, 2020.

[9] George Bebis and Michael Georgiopoulos. Feed-forward neural networks. *IEEE Potentials*, 13(4):27–31, 1994.

[10] Mikhail Belkin, Daniel Hsu, Siyuan Ma, and Soumik Mandal. Reconciling modern machine-learning practice and the classical bias–variance trade-off. *Proceedings of the National Academy of Sciences*, 116(32):15849–15854, 2019.

[11] Mikhail Belkin, Alexander Rakhlin, and Alexandre B Tsybakov. Does data interpolation contradict statistical optimality? In *The 22nd International Conference on Artificial Intelligence and Statistics*, pages 1611–1619. PMLR, 2019.

[12] Joe Benton, Valentin De Bortoli, Arnaud Doucet, and George Deligiannidis. Linear convergence bounds for diffusion models via stochastic localization. *arXiv preprint arXiv:2308.03686*, 2023.

[13] Soufiane Bentout, Abdennasser Chekroun, and Toshikazu Kuniya. Parameter estimation and prediction for coronavirus disease outbreak 2019 (COVID-19) in Algeria. *AIMS Public Health*, 7(2):306, 2020.

[14] Jens Berg and Kaj Nyström. Data-driven discovery of PDEs in complex datasets. *Journal of Computational Physics*, 384:239–252, 2019.

[15] Gregory Beylkin, Jochen Garcke, and Martin J Mohlenkamp. Multivariate regression and machine learning with sums of separable functions. *SIAM Journal on Scientific Computing*, 31(3):1840–1857, 2009.

[16] Kush Bhatia, Prateek Jain, and Purushottam Kar. Robust regression via hard thresholding. *Advances in Neural Information Processing Systems*, 28, 2015.

[17] Peter Binev, Wolfgang Dahmen, and Philipp Lamby. Fast high-dimensional approximation with sparse occupancy trees. *Journal of Computational and Applied Mathematics*, 235(8):2063–2076, 2011.

[18] Adam Block, Youssef Mroueh, and Alexander Rakhlin. Generative modeling with denoising auto-encoders and Langevin sampling. *arXiv preprint arXiv:2002.00107*, 2020.

[19] Michael GB Blum and Viet Chi Tran. HIV with contact tracing: a case study in approximate Bayesian computation. *Biostatistics*, 11(4):644–660, 2010.

[20] Vivek S Borkar, Vikranth R Dwaracherla, and Neeraja Sahasrabudhe. Gradient estimation with simultaneous perturbation and compressive sensing. *The Journal of Machine Learning Research*, 18(1):5910–5936, 2017.

[21] Rasmus Bro and Age K Smilde. Principal component analysis. *Analytical methods*, 6(9):2812–2831, 2014.

[22] David S Broomhead and Gregory P King. Extracting qualitative dynamics from experimental data. *Physica D: Nonlinear Phenomena*, 20(2-3):217–236, 1986.

[23] Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, et al. Language models are few-shot learners. *Advances in Neural Information Processing Systems*, 33:1877–1901, 2020.

[24] Simone Brugiapaglia, Sjoerd Dirksen, Hans Christian Jung, and Holger Rauhut. Sparse recovery in bounded Riesz systems with applications to numerical methods for PDEs. *Applied and Computational Harmonic Analysis*, 53:231–269, 2021.

[25] Simone Brugiapaglia, Stefano Micheletti, Fabio Nobile, and Simona Perotto. Wavelet–Fourier corsing techniques for multidimensional advection–diffusion–reaction equations. *IMA Journal of Numerical Analysis*, 41(4):2744–2781, 2021.

[26] Steven L Brunton, Bingni W Brunton, Joshua L Proctor, Eurika Kaiser, and J Nathan Kutz. Chaos as an intermittently forced linear system. *Nature Communications*, 8(1):1–9, 2017.

[27] Steven L Brunton and J Nathan Kutz. *Data-driven science and engineering: Machine learning, dynamical systems, and control.* Cambridge University Press, 2022.

[28] Steven L Brunton, Joshua L Proctor, and J Nathan Kutz. Discovering governing equations from data by sparse identification of nonlinear dynamical systems. *Proceedings of the National Academy of Sciences*, 113(15):3932–3937, 2016.

[29] Shengze Cai, Zhicheng Wang, Sifan Wang, Paris Perdikaris, and George Em Karniadakis. Physics-informed neural networks for heat transfer problems. *Journal of Heat Transfer*, 143(6):060801, 2021.

[30] Colin Campbell. Kernel methods: a survey of current techniques. *Neurocomputing*, 48(1-4):63–84, 2002.

[31] Alex Capaldi, Samuel Behrend, Benjamin Berman, Jason Smith, Justin Wright, and Alun L Lloyd. Parameter estimation and uncertainty quantication for an epidemic model. *Mathematical Biosciences and Engineering*, page 553, 2012.

[32] Simon Cauchemez and Neil M Ferguson. Likelihood-based estimation of continuous-time epidemic models from time-series data: application to measles transmission in London. *Journal of the Royal Society Interface*, 5(25):885–897, 2008.

[33] Jair Cervantes, Farid Garcia-Lamont, Lisbeth Rodríguez-Mazahua, and Asdrubal Lopez. A comprehensive survey on support vector machine classification: Applications, challenges and trends. *Neurocomputing*, 408:189–215, 2020.

[34] Kathleen P Champion, Steven L Brunton, and J Nathan Kutz. Discovery of nonlinear multiscale systems: Sampling strategies and embeddings. *SIAM Journal on Applied Dynamical Systems*, 18(1):312–333, 2019.

[35] Sitan Chen, Sinho Chewi, Jerry Li, Yuanzhi Li, Adil Salim, and Anru Zhang. Sampling is as easy as learning the score: theory for diffusion models with minimal data assumptions. In *The Eleventh International Conference on Learning Representations, ICLR 2023, Kigali, Rwanda, May 1-5, 2023.* OpenReview.net, 2023.

[36] Zhijun Chen and Hayden Schaeffer. Conditioning of random feature matrices: Double descent and generalization error. *arXiv preprint arXiv:2110.11477*, 2021.

[37] Krzysztof Marcin Choromanski, Valerii Likhosherstov, David Dohan, Xingyou Song, Andreea Gane, Tamás Sarlós, Peter Hawkins, Jared Quincy Davis, Afroz Mohiuddin, Lukasz Kaiser, David Benjamin Belanger, Lucy J. Colwell, and Adrian Weller. Rethinking attention with performers. In *9th International Conference on Learning Representations, ICLR 2021, Virtual Event, Austria, May 3-7, 2021*. OpenReview.net, 2021.

[38] Krzysztof Marcin Choromanski, Han Lin, Haoxian Chen, Arijit Sehanobish, Yuanzhe Ma, Deepali Jain, Jake Varley, Andy Zeng, Michael S. Ryoo, Valerii Likhosherstov, Dmitry Kalashnikov, Vikas Sindhwani, and Adrian Weller. Hybrid random features. In *The Tenth International Conference on Learning Representations, ICLR 2022, Virtual Event, April 25-29, 2022*. OpenReview.net, 2022.

[39] Aakanksha Chowdhery, Sharan Narang, Jacob Devlin, Maarten Bosma, Gaurav Mishra, Adam Roberts, Paul Barham, Hyung Won Chung, Charles Sutton, Sebastian Gehrmann, Parker Schuh, Kensen Shi, Sasha Tsvyashchenko, Joshua Maynez, Abhishek Rao, Parker Barnes, Yi Tay, Noam Shazeer, Vinodkumar Prabhakaran, Emily Reif, Nan Du, Ben Hutchinson, Reiner Pope, James Bradbury, Jacob Austin, Michael Isard, Guy Gur-Ari, Pengcheng Yin, Toju Duke, Anselm Levskaya, Sanjay Ghemawat, Sunipa Dev, Henryk Michalewski, Xavier Garcia, Vedant Misra, Kevin Robinson, Liam Fedus, Denny Zhou, Daphne Ippolito, David Luan, Hyeontaek Lim, Barret Zoph, Alexander Spiridonov, Ryan Sepassi, David Dohan, Shivani Agrawal, Mark Omernick, Andrew M. Dai, Thanumalayan Sankaranarayana Pillai, Marie Pellat, Aitor Lewkowycz, Erica Moreira, Rewon Child, Oleksandr Polozov, Katherine Lee, Zongwei Zhou, Xuezhi Wang, Brennan Saeta, Mark Diaz, Orhan Firat, Michele Catasta, Jason Wei, Kathy Meier-Hellstern, Douglas Eck, Jeff Dean, Slav Petrov, and Noah Fiedel. Palm: Scaling language modeling with pathways. *J. Mach. Learn. Res.*, 24:240:1–240:113, 2023.

[40] Il Yong Chun, Ben Adcock, and Thomas M Talavage. Efficient compressed sensing SENSE pMRI reconstruction with joint sparsity promotion. *IEEE transactions on Medical Imaging*, 35(1):354–368, 2015.

[41] Corinna Cortes and Vladimir Vapnik. Support-vector networks. *Machine Learning*, 20:273–297, 1995.

[42] Estee Y Cramer, Evan L Ray, Velma K Lopez, Johannes Bracher, Andrea Brennen, Alvaro J Castro Rivadeneira, Aaron Gerding, Tilmann Gneiting, Katie H House, Yuxin Huang, et al. Evaluation of individual and ensemble probabilistic forecasts of COVID-19 mortality in the United States. *Proceedings of the National Academy of Sciences*, 119(15):e2113561119, 2022.

[43] Miles Cranmer, Sam Greydanus, Stephan Hoyer, Peter Battaglia, David Spergel, and Shirley Ho. Lagrangian neural networks. *arXiv preprint arXiv:2003.04630*, 2020.

[44] R Devipriya, S Dhamodharavadhani, and S Selvi. SEIR model for COVID-19 epidemic using delay differential equation. In *Journal of Physics: Conference Series*, volume 1767, page 012005. IOP Publishing, 2021.

[45] Ronald DeVore, Guergana Petrova, and Przemyslaw Wojtaszczyk. Approximation of functions of few variables in high dimensions. *Constructive Approximation*, 33(1):125–143, 2011.

[46] Laurent Dinh, Jascha Sohl-Dickstein, and Samy Bengio. Density estimation using real NVP. In *5th International Conference on Learning Representations, ICLR 2017, Toulon, France, April 24-26, 2017, Conference Track Proceedings*. OpenReview.net, 2017.

[47] Dheeru Dua and Casey Graff. UCI machine learning repository, 2017.

[48] Vanja Dukic, Hedibert F Lopes, and Nicholas G Polson. Tracking epidemics with Google flu trends data and a state-space SEIR model. *Journal of the American Statistical Association*, 107(500):1410–1426, 2012.

[49] Weinan E, Chao Ma, Stephan Wojtowytsch, and Lei Wu. Towards a mathematical understanding of neural network-based machine learning: what we know and what we don't. *arXiv preprint arXiv:2009.10713*, 2020.

[50] Oliver G Ernst, Antje Mugler, Hans-Jörg Starkloff, and Elisabeth Ullmann. On the convergence of generalized polynomial chaos expansions. *ESAIM: Mathematical Modelling and Numerical Analysis*, 46(2):317–339, 2012.

[51] William Feller. Retracted chapter: On the theory of stochastic processes, with particular reference to applications. In *Selected Papers I*, pages 769–798. Springer, 2015.

[52] Simon Foucart. Hard thresholding pursuit: an algorithm for compressive sensing. *SIAM Journal on Numerical Analysis*, 49(6):2543–2563, 2011.

[53] Simon Foucart and Holger Rauhut. *A Mathematical Introduction to Compressive Sensing*. Birkhäuser Basel, 2013.

[54] Jonathan Frankle and Michael Carbin. The lottery ticket hypothesis: Finding sparse, trainable neural networks. In *7th International Conference on Learning Representations, ICLR 2019, New Orleans, LA, USA, May 6-9, 2019*. OpenReview.net, 2019.

[55] Jerome H Friedman, Forest Baskett, and Leonard J Shustek. An algorithm for finding nearest neighbors. *IEEE Transactions on computers*, 100(10):1000–1006, 1975.

[56] Kunihiko Fukushima. Artificial vision by Deep CNN neocognitron. *IEEE Transactions on Systems, Man, and Cybernetics: Systems*, 51(1):76–90, 2021.

[57] Amin Ghadami and Bogdan I Epureanu. Data-driven prediction in dynamical systems: recent developments. *Philosophical Transactions of the Royal Society A*, 380(2229):20210213, 2022.

[58] Paolo Girardi and Carlo Gaetan. An SEIR model with time-varying coefficients for analyzing the SARS-CoV-2 Epidemic. *Risk Analysis*, 43(1):144–155, 2023.

[59] Raul González-García, Ramiro Rico-Martìnez, and Ioannis G Kevrekidis. Identification of distributed parameter systems: A neural net based approach. *Computers & Chemical Engineering*, 22:S965–S968, 1998.

[60] Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative adversarial nets. *Advances in Neural Information Processing Systems*, 27, 2014.

[61] Nina M Gottschling, Vegard Antun, Ben Adcock, and Anders C Hansen. The troublesome kernel: why deep learning for inverse problems is typically unstable. *arXiv preprint arXiv:2001.01258*, 2020.

[62] Pawan Goyal and Peter Benner. Discovery of nonlinear dynamical systems using a Runge–Kutta inspired dictionary-based sparse regression approach. *Proceedings of the Royal Society A*, 478(2262):20210883, 2022.

[63] Samuel Greydanus, Misko Dzamba, and Jason Yosinski. Hamiltonian neural networks. *Advances in Neural Information Processing Systems*, 32, 2019.

[64] Jiuxiang Gu, Zhenhua Wang, Jason Kuen, Lianyang Ma, Amir Shahroudy, Bing Shuai, Ting Liu, Xingxing Wang, Gang Wang, Jianfei Cai, et al. Recent advances in convolutional neural networks. *Pattern recognition*, 77:354–377, 2018.

[65] Nicola Guglielmi, Elisa Iacomini, and Alex Viguerie. Delay differential equations for the spatially resolved simulation of epidemics with specific application to covid-19. *Mathematical Methods in the Applied Sciences*, 45(8):4752–4771, 2022.

[66] Abolfazl Hashemi, Hayden Schaeffer, Robert Shi, Ufuk Topcu, Giang Tran, and Rachel Ward. Generalization bounds for sparse random feature expansions. *Applied and Computational Harmonic Analysis*, 62:310–330, 2023.

[67] Trevor Hastie, Andrea Montanari, Saharon Rosset, and Ryan J Tibshirani. Surprises in high-dimensional ridgeless least squares interpolation. *arXiv preprint arXiv:1903.08560*, 2019.

[68] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 770–778, 2016.

[69] Marti A. Hearst, Susan T Dumais, Edgar Osuna, John Platt, and Bernhard Scholkopf. Support vector machines. *IEEE Intelligent Systems and their Applications*, 13(4):18–28, 1998.

[70] Jonathan Ho, Ajay Jain, and Pieter Abbeel. Denoising diffusion probabilistic models. *Advances in Neural Information Processing Systems*, 33:6840–6851, 2020.

[71] Lam Si Tung Ho, Forrest W Crawford, and Marc A Suchard. Direct likelihood-based inference for discretely observed stochastic compartmental models of infectious disease. *The Annals of Applied Statistics*, 12(3):1993–2021, 2018.

[72] Lam Si Tung Ho, Jason Xu, Forrest W Crawford, Vladimir N Minin, and Marc A Suchard. Birth/birth-death processes and their computable transition probabilities with biological applications. *Journal of Mathematical Biology*, 76(4):911–944, 2018.

[73] Thomas Hofmann, Bernhard Schölkopf, and Alexander J Smola. Kernel methods in machine learning. 2008.

[74] Jer-Nan Juang and Richard S Pappa. An eigensystem realization algorithm for modal parameter identification and model reduction. *Journal of Guidance, Control, and Dynamics*, 8(5):620–627, 1985.

[75] Eurika Kaiser, J Nathan Kutz, and Steven L Brunton. Sparse identification of nonlinear dynamics for model predictive control in the low-data limit. *Proceedings of the Royal Society A*, 474(2219):20180335, 2018.

[76] Mason Kamb, Eurika Kaiser, Steven L Brunton, and J Nathan Kutz. Time-delay observables for Koopman: Theory and applications. *SIAM Journal on Applied Dynamical Systems*, 19(2):886–917, 2020.

[77] Kirthevasan Kandasamy and Yaoliang Yu. Additive approximations in high dimensional nonparametric regression via the SALSA. In *International Conference on Machine Learning*, pages 69–78. PMLR, 2016.

[78] Kenji Kawaguchi, Bo Xie, and Le Song. Deep semi-random features for nonlinear function approximation. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 32, 2018.

[79] Diederik P. Kingma and Max Welling. Auto-encoding variational bayes. In Yoshua Bengio and Yann LeCun, editors, *2nd International Conference on Learning Representations, ICLR 2014, Banff, AB, Canada, April 14-16, 2014, Conference Track Proceedings*, 2014.

[80] Carl Kingsford and Steven L Salzberg. What are decision trees? *Nature Biotechnology*, 26(9):1011–1013, 2008.

[81] Daphne Koller and Nir Friedman. *Probabilistic graphical models: principles and techniques*. MIT press, 2009.

[82] Sotiris B Kotsiantis. Decision trees: a recent overview. *Artificial Intelligence Review*, 39:261–283, 2013.

[83] Ellen Kuhl and Ellen Kuhl. Introduction to mathematical epidemiology. *Computational Epidemiology: Data-Driven Modeling of COVID-19*, pages 3–31, 2021.

[84] F Kuo, I Sloan, Grzegorz Wasilkowski, and Henryk Woźniakowski. On decompositions of multivariate functions. *Mathematics of Computation*, 79(270):953–966, 2010.

[85] Isaac E Lagaris, Aristidis Likas, and Dimitrios I Fotiadis. Artificial neural networks for solving ordinary and partial differential equations. *IEEE Transactions on Neural Networks*, 9(5):987–1000, 1998.

[86] Jeannette Lawrence. *Introduction to neural networks*. California Scientific Software, 1993.

[87] Soledad Le Clainche and José M Vega. Higher order dynamic mode decomposition. *SIAM Journal on Applied Dynamical Systems*, 16(2):882–925, 2017.

[88] Zewen Li, Fan Liu, Wenjie Yang, Shouheng Peng, and Jun Zhou. A survey of convolutional neural networks: analysis, applications, and prospects. *IEEE Transactions on Neural Networks and Learning Systems*, 2021.

[89] Zhu Li, Jean-Francois Ton, Dino Oglic, and Dino Sejdinovic. Towards a unified analysis of random Fourier features. In *International Conference on Machine Learning*, pages 3905–3914. PMLR, 2019.

[90] Tengyuan Liang and Alexander Rakhlin. Just interpolate: Kernel "ridgeless" regression can generalize. *The Annals of Statistics*, 48(3):1329–1347, 2020.

[91] Alex Tong Lin, Daniel Eckhardt, Robert Martin, Stanley Osher, and Adrian S Wong. Parameter inference of time series by delay embeddings and learning differentiable operators. *arXiv preprint arXiv:2203.06269*, 2022.

[92] Hongzhou Lin and Stefanie Jegelka. ResNet with one-neuron hidden layers is a universal approximator. *Advances in Neural Information Processing Systems*, 31, 2018.

[93] Guodong Liu, Hong Chen, and Heng Huang. Sparse shrunk additive models. In *International Conference on Machine Learning*, pages 6194–6204. PMLR, 2020.

[94] Cheng Lu, Yuhao Zhou, Fan Bao, Jianfei Chen, Chongxuan Li, and Jun Zhu. DPM-solver: A fast ODE solver for diffusion probabilistic model sampling in around 10 steps. *Advances in Neural Information Processing Systems*, 35:5775–5787, 2022.

[95] Lu Lu, Xuhui Meng, Zhiping Mao, and George Em Karniadakis. DeepXDE: A deep learning library for solving differential equations. *SIAM Review*, 63(1):208–228, 2021.

[96] Bethany Lusch, J Nathan Kutz, and Steven L Brunton. Deep learning for universal linear embeddings of nonlinear dynamics. *Nature Communications*, 9(1):1–10, 2018.

[97] Niall M Mangan, J Nathan Kutz, Steven L Brunton, and Joshua L Proctor. Model selection for dynamical systems via sparse regression and information criteria. *Proceedings of the Royal Society A: Mathematical, Physical and Engineering Sciences*, 473(2204):20170009, 2017.

[98] Maia Martcheva. *An introduction to mathematical epidemiology*, volume 61. Springer, 2015.

[99] Song Mei, Theodor Misiakiewicz, and Andrea Montanari. Generalization error of random feature and kernel methods: Hypercontractivity and kernel matrix concentration. *Applied and Computational Harmonic Analysis*, 59:3–84, 2022.

[100] Song Mei and Andrea Montanari. The generalization error of random features regression: Precise asymptotics and the double descent curve. *Communications on Pure and Applied Mathematics*, 2019.

[101] Jacob Menick and Nal Kalchbrenner. Generating high fidelity images with subscale pixel networks and multidimensional upscaling. In *7th International Conference on Learning Representations, ICLR 2019, New Orleans, LA, USA, May 6-9, 2019*. OpenReview.net, 2019.

[102] David Meyer, Friedrich Leisch, and Kurt Hornik. The support vector machine under test. *Neurocomputing*, 55(1-2):169–186, 2003.

[103] Taketomo Mitsui and Guang-Da Hu. *Numerical Analysis of Ordinary and Delay Differential Equations*, volume 145. Springer Nature, 2023.

[104] Kumpati S Narendra and Kannan Parthasarathy. Neural networks and dynamical systems. *International Journal of Approximate Reasoning*, 6(2):109–131, 1992.

[105] Nicholas H Nelsen and Andrew M Stuart. The random feature model for input-output maps between Banach spaces. *SIAM Journal on Scientific Computing*, 43(5):A3212–A3243, 2021.

[106] Himanshu Pandotra, Eeshan Malhotra, Ajit Rajwade, and Karthik S Gurumoorthy. Signal recovery in perturbed Fourier compressed sensing. In *2018 IEEE Global Conference on Signal and Information Processing (GlobalSIP)*, pages 326–330. IEEE, 2018.

[107] Daniel Potts and Michael Schmischke. Interpretable approximation of high-dimensional data. *SIAM Journal on Mathematics of Data Science*, 3(4):1301–1323, 2021.

[108] Tong Qin, Kailiang Wu, and Dongbin Xiu. Data driven governing equations approximation using deep neural networks. *Journal of Computational Physics*, 395:620–635, 2019.

[109] Ali Rahimi and Benjamin Recht. Random features for large-scale kernel machines. In *NIPS*, volume 3, page 5. Citeseer, 2007.

[110] Ali Rahimi and Benjamin Recht. Uniform approximation of functions with random bases. In *2008 46th Annual Allerton Conference on Communication, Control, and Computing*, pages 555–561. IEEE, 2008.

[111] Ali Rahimi and Benjamin Recht. Weighted sums of random kitchen sinks: replacing minimization with randomization in learning. In *NIPS*, pages 1313–1320. Citeseer, 2008.

[112] Maziar Raissi, Paris Perdikaris, and George E Karniadakis. Physics-informed neural networks: A deep learning framework for solving forward and inverse problems involving nonlinear partial differential equations. *Journal of Computational Physics*, 378:686–707, 2019.

[113] Pradeep Ravikumar, Han Liu, John D Lafferty, and Larry A Wasserman. SpAM: Sparse additive models. In *NIPS*, pages 1201–1208, 2007.

[114] Waseem Rawat and Zenghui Wang. Deep convolutional neural networks for image classification: A comprehensive review. *Neural Computation*, 29(9):2352–2449, 2017.

[115] Nicholas Richardson, Hayden Schaeffer, and Giang Tran. SRMD: Sparse random mode decomposition. *Communications on Applied Mathematics and Computation*, pages 1–28, 2023.

[116] Vijay K Rohatgi and AK Md Ehsanes Saleh. *An introduction to probability and statistics*. John Wiley & Sons, 2015.

[117] Diana Patricia Rojas, Natalie E Dean, Yang Yang, Eben Kenah, Juliana Quintero, Simon Tomasi, Erika Lorena Ramirez, Yendi Kelly, Carolina Castro, Gabriel Carrasquilla, et al. The epidemiology and transmissibility of Zika virus in Girardot and San Andres island, Colombia, September 2015 to January 2016. *Eurosurveillance*, 21(28):30283, 2016.

[118] O Ronneberger, P Fischer, and T Brox. Convolutional networks for biomedical image segmentation. In *Medical Image Computing and Computer-Assisted Intervention–MICCAI 2015 Conference Proceedings*, 2022.

[119] Olaf Ronneberger, Philipp Fischer, and Thomas Brox. U-net: Convolutional networks for biomedical image segmentation. In *Medical Image Computing and Computer-Assisted Intervention–MICCAI 2015: 18th International Conference, Munich, Germany, October 5-9, 2015, Proceedings, Part III 18*, pages 234–241. Springer, 2015.

[120] Alessandro Rudi and Lorenzo Rosasco. Generalization properties of learning with random features. In *NIPS*, pages 3215–3225, 2017.

[121] Samuel H Rudy, Steven L Brunton, Joshua L Proctor, and J Nathan Kutz. Data-driven discovery of partial differential equations. *Science Advances*, 3(4):e1602614, 2017.

[122] Esha Saha, Lam Si Tung Ho, and Giang Tran. SPADE4: Sparsity and delay embedding based forecasting of epidemics. *Bulletin of Mathematical Biology*, 85(8):71, 2023.

[123] Esha Saha, Hayden Schaeffer, and Giang Tran. HARFE: Hard-ridge random feature expansion. *Sampling Theory, Signal Processing, and Data Analysis*, 21(2):1–24, 2023.

[124] Esha Saha and Giang Tran. Diffusion random feature model. 2023.

[125] Hayden Schaeffer. Learning partial differential equations via data discovery and sparse optimization. *Proceedings of the Royal Society A: Mathematical, Physical and Engineering Sciences*, 473(2197):20160446, 2017.

[126] Hayden Schaeffer, Giang Tran, and Rachel Ward. Extracting sparse high-dimensional dynamics from limited data. *SIAM Journal on Applied Mathematics*, 78(6):3279–3295, 2018.

[127] Jürgen Schmidhuber. Deep learning in neural networks: An overview. *Neural Networks*, 61:85–117, 2015.

[128] John Shawe-Taylor and Nello Cristianini. *Kernel methods for pattern analysis*. Cambridge university press, 2004.

[129] Vincent Sitzmann, Julien Martel, Alexander Bergman, David Lindell, and Gordon Wetzstein. Implicit neural representations with periodic activation functions. *Advances in Neural Information Processing Systems*, 33:7462–7473, 2020.

[130] Alexandra Smirnova, Linda deCamp, and Gerardo Chowell. Forecasting epidemics through nonparametric estimation of time-dependent transmission rates using the SEIR model. *Bulletin of Mathematical Biology*, 81:4343–4365, 2019.

[131] Alex J Smola and Bernhard Schölkopf. *Learning with kernels*, volume 4. Citeseer, 1998.

[132] Jascha Sohl-Dickstein, Eric Weiss, Niru Maheswaranathan, and Surya Ganguli. Deep unsupervised learning using nonequilibrium thermodynamics. In *International Conference on Machine Learning*, pages 2256–2265. PMLR, 2015.

[133] Yang Song and Stefano Ermon. Generative modeling by estimating gradients of the data distribution. *Advances in Neural Information Processing Systems*, 32, 2019.

[134] Yang Song and Stefano Ermon. Improved techniques for training score-based generative models. *Advances in Neural Information Processing Systems*, 33:12438–12448, 2020.

[135] Yang Song, Liyue Shen, Lei Xing, and Stefano Ermon. Solving inverse problems in medical imaging with score-based generative models. In *The Tenth International Conference on Learning Representations, ICLR 2022, Virtual Event, April 25-29, 2022*. OpenReview.net, 2022.

[136] Wei-Hung Su, Ching-Shan Chou, and Dongbin Xiu. Deep learning of biological models from data: applications to ODE models. *Bulletin of Mathematical Biology*, 83(3):1–19, 2021.

[137] Daniel Svozil, Vladimir Kvasnicka, and Jiri Pospichal. Introduction to multi-layer feed-forward neural networks. *Chemometrics and Intelligent Laboratory Systems*, 39(1):43–62, 1997.

[138] Floris Takens. Detecting strange attractors in turbulence. In *Dynamical Systems and Turbulence, Warwick 1980: proceedings of a symposium held at the University of Warwick 1979/80*, pages 366–381. Springer, 2006.

[139] Kashvi Taunk, Sanjukta De, Srishti Verma, and Aleena Swetapadma. A brief review of nearest neighbor algorithm for learning and classification. In *2019 International Conference on Intelligent Computing and Control Systems (ICCS)*, pages 1255–1260. IEEE, 2019.

[140] Hugo Touvron, Thibaut Lavril, Gautier Izacard, Xavier Martinet, Marie-Anne Lachaux, Timothée Lacroix, Baptiste Rozière, Naman Goyal, Eric Hambro, Faisal Azhar, et al. Llama: Open and efficient foundation language models. *arXiv preprint arXiv:2302.13971*, 2023.

[141] Giang Tran and Rachel Ward. Exact recovery of chaotic systems from highly corrupted data. *Multiscale Modeling & Simulation*, 15(3):1108–1129, 2017.

[142] Alexander Tsigler and Peter L Bartlett. Benign overfitting in ridge regression. *J. Mach. Learn. Res.*, 24:123–1, 2023.

[143] Shreshth Tuli, Shikhar Tuli, Rakesh Tuli, and Sukhpal Singh Gill. Predicting the growth and trend of COVID-19 pandemic using machine learning and cloud computing. *Internet of Things*, 11:100222, 2020.

[144] Gonzalo Uribarri and Gabriel B Mindlin. Dynamical time series embeddings in recurrent neural networks. *Chaos, Solitons & Fractals*, 154:111612, 2022.

[145] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. *Advances in Neural Information Processing Systems*, 30, 2017.

[146] René Vidal, Yi Ma, S Shankar Sastry, René Vidal, Yi Ma, and S Shankar Sastry. Principal component analysis. *Generalized Principal Component Analysis*, pages 25–62, 2016.

[147] Pascal Vincent. A connection between score matching and denoising autoencoders. *Neural Computation*, 23(7):1661–1674, 2011.

[148] Pantelis R Vlachas, Wonmin Byeon, Zhong Y Wan, Themistoklis P Sapsis, and Petros Koumoutsakos. Data-driven forecasting of high-dimensional chaotic systems with long short-term memory networks. *Proceedings of the Royal Society A: Mathematical, Physical and Engineering Sciences*, 474(2213):20170844, 2018.

[149] E Weinan. A proposal on machine learning via dynamical systems. *Communications in Mathematics and Statistics*, 1(5):1–11, 2017.

[150] Zifeng Wu, Chunhua Shen, and Anton Van Den Hengel. Wider or deeper: Revisiting the ResNet model for visual recognition. *Pattern Recognition*, 90:119–133, 2019.

[151] Yuege Xie, Robert Shi, Hayden Schaeffer, and Rachel Ward. Shrimp: Sparser random feature models via iterative magnitude pruning. In *Mathematical and Scientific Machine Learning*, pages 303–318. PMLR, 2022.

[152] Ling Yang, Zhilong Zhang, Yang Song, Shenda Hong, Runsheng Xu, Yue Zhao, Yingxia Shao, Wentao Zhang, Bin Cui, and Ming-Hsuan Yang. Diffusion models: A comprehensive survey of methods and applications. *arXiv preprint arXiv:2209.00796*, 2022.

[153] Ian En-Hsu Yen, Ting-Wei Lin, Shou-De Lin, Pradeep K Ravikumar, and Inderjit S Dhillon. Sparse random feature algorithm as coordinate descent in Hilbert space. In *Advances in Neural Information Processing Systems*, pages 2456–2464, 2014.

[154] Linan Zhang and Hayden Schaeffer. On the convergence of the SINDy algorithm. *Multiscale Modeling & Simulation*, 17(3):948–972, 2019.

[155] Qinsheng Zhang and Yongxin Chen. Diffusion normalizing flow. *Advances in Neural Information Processing Systems*, 34:16280–16291, 2021.

[156] Qinsheng Zhang, Molei Tao, and Yongxin Chen. gddim: Generalized denoising diffusion implicit models. In *The Eleventh International Conference on Learning Representations, ICLR 2023, Kigali, Rwanda, May 1-5, 2023*. OpenReview.net, 2023.

[157] Tong Zhang. Learning bounds for kernel regression using effective data dimensionality. *Neural Computation*, 17(9):2077–2098, 2005.

[158] Hattie Zhou, Janice Lan, Rosanne Liu, and Jason Yosinski. Deconstructing lottery tickets: Zeros, signs, and the supermask. *Advances in Neural Information Processing Systems*, 32, 2019.

[159] Difan Zou, Lingxiao Wang, Pan Xu, Jinghui Chen, Weitong Zhang, and Quanquan Gu. Epidemic model guided machine learning for COVID-19 forecasts in the United States. *MedRxiv*, 2020.