DATA-EFFICIENT DESIGN AND ANALYSIS METHODOLOGIES FOR COMPUTER AND PHYSICAL EXPERIMENTS

A Dissertation

by

GECHENG CHEN

Submitted to the Graduate and Professional School of
Texas A&M University
in partial fulfillment of the requirements for the degree of

DOCTOR OF PHILOSOPHY

Chair of Committee,	Rui Tuo
Committee Members,	Raymond Wong
	Satish Bukkapatnam
	Xudong Zhang
Head of Department,	Lewis Ntaimo

May 2023

Major Subject: Industrial Engineering

ABSTRACT

Data science for experimentation, including the rapidly growing area of the design and analysis of computer experiments, aims to use statistical approaches to collect and analyze (physical or virtual) experimental responses and facilitate decision-making. The cost for each run of an experiment can be expensive. This dissertation proposes novel data-efficient methodologies to tackle three different challenges in this field. The first two are regarding computer experiments, and the third one is regarding physical experiments.

The first work aims to reconstruct the input-output relationship (surrogate model) given by the computer code via scattered evaluations with small sizes based on Gaussian process regression. Traditional isotropic Gaussian process models suffer from the curse of dimensionality when the input dimension is relatively high given limited data points. Gaussian process models with additive correlation functions are scalable to dimensionality, but they are more restrictive as they only work for additive functions. In the first work, we consider a projection pursuit model in which the nonparametric part is driven by an additive Gaussian process regression. We choose the dimension of the additive function higher than the original input dimension and call this strategy "dimension expansion". We show that dimension expansion can help approximate more complex functions. A gradient descent algorithm is proposed for model training based on the maximum likelihood estimation. Simulation studies show that the proposed method outperforms the traditional Gaussian process models.

The second work focuses on the designs of experiments (DoE) of multi-fidelity computer experiments with fixed budget. We consider the autoregressive Gaussian process model and the optimal nested design that maximizes the prediction accuracy subject to the budget constraint. An approximate solution is identified through the idea of multilevel approximation and recent error bounds of Gaussian process regression. The proposed (approximately) optimal designs admit a simple analytical form. We prove that, to achieve the same prediction accuracy, the proposed optimal multi-fidelity design requires much lower computational cost than any single-fidelity design

in the asymptotic sense.

The last work is proposed to model complex experiments when the distributions of training and testing input features are different (referred to as domain adaptation). In this work, we propose a novel transfer learning algorithm called Renewing Iterative Self-labeling Domain Adaptation (Re-ISDA) to tackle the domain adaptation problem. The learning problem is formulated as a dynamic programming model, and the latter is then solved by an efficient greedy algorithm by adding a renewing step to the original ISDA algorithm. This renewing step helps avoid a potential issue of the ISDA that the possible mis-labeled samples by a weak predictor in the initial stage of the iterative learning can cause serious harm to the subsequent learning process. Numerical studies show that the proposed method outperforms prevailing transfer learning methods. The proposed method also achieves high prediction accuracy for a cervical spine motion problem.

DEDICATION

To my mother, father, grandfather, and grandmother.

ACKNOWLEDGMENTS

First and foremost, I am extremely grateful to my supervisor, Prof. Rui Tuo, for his invaluable advice, continuous support, and patience during my Ph.D. study. His immense knowledge and plentiful experience have encouraged me in all the time of my academic research and daily life. I would also like to thank Dr. Xudong Zhang and Ph.D. student Yu Zhou for their technical support in my study for the spine motion prediction project. I would like to thank all the committee members. It is their kind help and support that have made my study and life in College Station a wonderful time. Finally, I would like to express my gratitude to my parents. Without their tremendous understanding and encouragement in the past few years, it would be impossible for me to complete my study.

CONTRIBUTORS AND FUNDING SOURCES

Contributors

This work was supported by a dissertation committee consisting of Prof. Rui Tuo [advisor] and Prof. Raymond Wong of the Department of Statistics, Prof. Satish Bukkapatnam of the Department of Industrial & Systems Engineering and Prof. Xudong Zhang of the Department of Industrial & Systems Engineering.

The spine motion data used in Chapter II was provided by Prof. Xudong Zhang and Ph.D student Yu Zhou.

All other work was completed by the student independently.

Funding Sources

Graduate study was supported by NSF grants DMS-1914636 and CCF-1934904. The work in Chapter 4 was supported by the Centers for Disease Control and Prevention/National Institute for Occupational Safety and Health under a research grant R010H010587.

NOMENCLATURE

DoE	Design of Experiment
ANOVA	Analysis of Variance
GPR	Gaussian Process Regression
PPR	Projection Pursuit Regression
PPGPR	Projection Pursuit Gaussian Process Regression
NLHD	Nested Latin Hypercube Design
RMSE	Root Mean Square Error
MAPE	Mean Absolute Percentage Error
ISDA	Iterative Self-labeling Domain Adaptation
NN	Neural Network
Re-ISDA	Renewing Iterative Self-labeling Domain Adaptation
MLE	Maximum Likelihood Estimation
FEM	Finite Element Method
MLGP	Multi-level Gaussian Process
PDE	Partial Differential Equation

TABLE OF CONTENTS

Page

LIST OF FIGURES

FIGURE Page

LIST OF TABLES

TABLE Page

1. INTRODUCTION

1.1 Background

Experimentation is one of the most common activities people implement to explore relationships between the input features and the response in applications [1]. It starts by setting the input variables purposely to the system (also referred to as designs of experiments (DoE)) and then models the input-output relationships via statistical tools based on the collected data. Modern experimental design and analysis date back to the pioneering work of R. A. Fisher in the 1930s in the development of blocking, randomization, replication, orthogonality, and the use of analysis of variance (ANOVA) and fractional factorial designs. G. E. P. Box and co-workers found that focusing more on process modeling and optimization than treatment comparison was more valuable, which led to the development of central composite development and optimal designs. After that, G. Taguchi advocated using robust parameter design to make the system less sensitive to variation.

More recently, new challenges have been posed to traditional experimentation methodologies. On the one hand, today's experiments can be very complex. Given these complexities, traditional methods, e.g., linear models, ANOVA, and factorial designs, are too naive to work well [2; 1]. For instance, making designs for experiments involving human motion can be very hard using traditional methods. On the other hand, in many cases, collecting enough input-response pairs from physical experiments is hard because each run of a physical experiment can be very costly. Given these two challenges, data-efficient methodologies with stronger modeling abilities are urgently needed.

Modern experimentation methodologies are now mainly developed in two directions to solve the aforementioned challenges. One direction is to implement more advanced methodologies, such as Gaussian process regression [3; 4] and neural networks [5; 6], to model complex computer experiments efficiently. The other direction is to use *computer experiments* to mimic complex physical systems. Compared with physical experiments, computer experiments are often much

less costly and more flexible to run. For example, computer models are implemented to simulate oil pressure at a well of a hydrocarbon reservoir [7]. Conceptually, each computer model is based on a mathematical model in terms of, for instance, a set of partial differential equations. The aim of a computer code run is to find a numerical solution to this mathematical model. However, each computer code run can still be expensive. For example, each run of a typical computational fluid dynamics model for aerospace engineering takes a few days or even weeks to run [8]. The design and analysis of computer experiments, an area drawing increasing attention recently, aim to use statistical approaches to collect and analyze computer simulation responses and facilitate decision-making. We refer to the books [2; 3] for an introduction to this field.

One of the most important tasks for computer experiments is to build fast and accurate *surrogate models* of computer experiments. A surrogate model is a statistical model constructed using the computer outputs over a selected set of input sites. A point evaluation of a surrogate model is much less costly than running the corresponding computer simulation code. Thus we can use the surrogate model to explore and analyze the underlying computer response function, which expedites the decision-making. The step towards an efficient surrogate model is to make efficient DoEs. As observed by many researchers, there is an essential distinction between designs for computer experiments when constructing the surrogate model [9; 2]. Therefore, another critical mission of computer experiments is to obtain efficient designs of experiments. In real applications, a limited budget is often allowed to make designs. Given that, we need to implement data-efficient design methods to collect information as much as possible.

This dissertation focuses on data-efficient methodologies for three different challenges, i.e., high-dimensional data modeling, multi-fidelity experimental designs, and domain adaptation in the design and analysis of experimentation. The first two are regarding computer experiments, and the third one focuses on physical experiments. In the rest of this chapter, the motivation and related works will be included for these three works, respectively.

1.2 Projection Pursuit Gaussian Process Regression

Gaussian process regression [10; 2] is one of the most popular surrogate models. Various modifications and extensions of the standard Gaussian process regression models have been proposed to address the specific needs in practical situations. An incomplete list of these methods include composite Gaussian processes [11], treed Gaussian processes [12], non-stationary models [13], transformed approximately additive Gaussian processes [14], etc.

Data analysis for computer simulations usually suffers from the "small data" issue, because the computer simulation runs can be highly costly. For example, each run of a typical computational fluid dynamics model for aerospace engineering takes a few days or even weeks to run [8]. Many computer simulations also pose the *curse of dimensionality* problem, in the sense that the input dimension is relatively high so that building an accurate surrogate model based on limited data points becomes more challenging. Classic approaches for dimension reduction in computer experiments include sensitivity analysis [15; 16; 17], ridge approximation [18; 19; 20]. Variable selection for Gaussian processes models is considered in [21], [22] and [23]. In Gaussian process regression, it is also known that some correlation structures perform better in high-dimensional scenarios [24]. Recently, additive Gaussian process models have received considerable attention [25; 26; 27; 28; 29; 30]. Although these models are more scalable to the input dimension, their capability of model fitting is lower because these models can only reconstruct additive functions precisely.

In this work, we propose a novel surrogate modeling technique based on the projection pursuit methodology [31] and additive Gaussian process models. Gaussian process regression can provide prediction variance as opposed to projection pursuit (neural networks). Additionally, unlike the conventional estimation approaches for projection pursuit [32; 33; 34], we suggest choosing a large number of intermediate nodes to introduce more model flexibility. Then we use the maximum likelihood estimation to identify the model parameters. A gradient descent algorithm is proposed to search the maximum of the likelihood function. In this work, we also find an error bound of the prediction error for Gaussian process regression with additive Matérn correlation functions. Our

theoretical results show that the prediction error of additive Gaussian process models is much lower than that given by isotropic Gaussian process models for high-dimensional problems, provided that a design with nice projection properties, such as a Latin hypercube design, is adopted.

1.3 Fixed-budget optimal designs for multi-fidelity computer experiments

Many computer simulation codes allow users to specify an accuracy level of the numerical responses in terms of the degree of discretization, the number of iterations, etc. Generally, there is a trade-off between the accuracy and the computational cost: a high-accuracy computer run is more costly than a low-accuracy run [35; 36]. Many statisticians have considered the problem of integrating the computer responses from multiple accuracy (also known as fidelity) levels to make predictions. The first model was suggested by [7], who proposed a Gaussian-process-based autoregressive model to integrate the output data from computer experiments with multiple fidelity levels. This model has been widely used in real-world applications, e.g., simulation oil pressure [7], designing of linear cellular alloy [36] and prediction of band gaps [37]. Alternative models multi-fidelity computer experiments include Bayesian hierarchical models [35], deep Gaussian process model [38], and nonstationary Gaussian process models [39]. In this article, we focus on the autoregressive models given the wide acceptance of these models.

Despite the widespread use of multi-fidelity computer experiments, there remain some unanswered questions. First, it is unclear why one would prefer a multi-fidelity computer experiment over a simple single-fidelity experiment. A conventional explanation says that the low-fidelity data can help explore the underlying response function quickly, and the high-fidelity data are then used to refine the model. This assertion is usually confirmed by numerical studies, but there is a lack of quantitative analysis of this relationship.

The second question is regarding the choice of the set of input sites for the computer simulation, known as a design of experiment. Studying the design of computer experiments is considered important because each computer run can be costly, and a good design can help collect more information under the same computational cost [2; 40]. The importance of design of experiments is even more dramatic for multi-fidelity computer experiments, because a poor design can even

make an adverse effect on the performance.

A number of designs for multi-fidelity computer experiments are proposed in the literature. [41] proposed the Nest Latin hypercube designs (NLHDs). The design points of an NLHD on one fidelity level forms a Latin hypercube, and those from different fidelity levels are nested. [42] proposed the nested orthogonal array-based Latin hypercube designs to achieve stratification in both bivariate and univariate margins. [9] constructed nested maximin Latin hypercube designs using a numerical search algorithm. See also [41; 43; 44; 45; 39; 36; 42; 46] for more related works. Despite the rich literature, none of these works addressed the aforementioned questions. Some existing works provided theoretical justifications in terms of numerical integration, e.g., [47], but it is still not clear why these designs are suitable for the autoregressive models.

There is another class of approaches from the area of applied mathematics for a related problem, known as multilevel Monte Carlo (MLMC) [48; 49; 50]. These methods are algorithms for computing expectations that arise in stochastic simulations with different levels of accuracy. reduces the computational cost of the traditional Monte Carlo method by taking most samples with low accuracy, and only very few samples taken at high accuracy. Multifidelity Monte Carlo (MFMC) [51], based on the control variable techniques for Monte Carlo simulation, resembles the MLMC method for more general engineering models. More related works can be found in [52; 53; 54; 55; 56]. Even though these multilevel based design are equipped with rigorous mathematical foundations, they are not based on Gaussian process models so that they cannot be used for posterior-based analysis and uncertainty quantification.

This work aims at underpinning the multi-fidelity computer experiments by analyzing their predictive error theoretically. Inspired by the idea of multilevel methods, we formulate a model framework for multi-fidelity computer experiments. We then consider the fixed precision optimal design defined as the design with the minimum total simulation cost subject to an accuracy guarantee. Our main theoretical contribution is to obtain the order of magnitude of the simulation cost as the required predictive error of the autoregressive model goes to zero. The analysis is based on the error bounds for Gaussian process regression provided by [57]. We prove that a well-designed

multi-fidelity experiment will require much lower simulation cost in the order of magnitude to reach the same accuracy level, compared with the best single-fidelity experiment. This result confirms the superiority of multi-fidelity computer experiments. Based on the theory, we propose a novel design scheme. Compared with the existing optimal design methods, the proposed method is easy to implement because it does not require a numerical search. Numerical studies confirm the superiority of the proposed method over existing alternatives.

1.4 Renewing Iterative Self-labeling Domain Adaptation with Application to the Spine Motion Prediction

Advanced supervised learning approaches, e.g., neural networks, are widely used in modeling complex physical experiments. A primary task of supervised learning is to learn the input-output relationship within a training set. This learned relationship is described by a model; the model can then be used to make predictions for another set of interests, e.g., the testing set. Traditional supervised machine learning algorithms learn models based only on labeled training data. In practice, the marginal distribution of the training inputs often differs from that of the testing set. See [58] for an example in web-document classification. When this difference is large, traditional supervised machine learning algorithms may not give accurate predictions on the testing set, even if their models fitting on the training set are good. In such a circumstance, one needs to consider a transfer learning method, which builds machine learning models based on not only the training data but also the inputs of the testing set. We refer to [58; 59] for an introduction to this field.

According to [58], a domain is defined as $D = \{X, P(X)\}$, where X stands for a feature space and $P(X)$ denotes the marginal distribution of these features. Besides, a task T based on a specific domain D is defined as $T = \{Y, P(Y|X)\}$ where Y is the label space and $P(Y|X)$ denotes the conditional distribution. Given a source domain (training set) D_s with a learning task T_s and a target domain (test set) D_t with a learning task T_t, transfer learning aims to improve the learning of the predictive function on the target domain when $D_t \neq D_s$ or $T_s \neq T_t$. To be simple, transfer learning is proposed to enhance the prediction when the source and target domains are correlated but not the same. Based on this definition, the aforementioned difference between input features

can be summarised as $X_s = X_t$, $P_s(X) \neq P_t(X)$, and $T_s = T_t$, which is also referred to as *domain adaptation*.

The existing transfer learning methods for domain adaptation mainly fall into two general categories. The first category, instance-based transfer learning, proceeds by re-sampling the training data according to suitably chosen weights to approximate the distribution of testing samples. One of the most popular instance based methods is Kernel Mean Matching (KMM) [60], which estimates the weights for the source samples to make the input from two domains close. Dai proposes TrAdaboost in [61] to apply the Adaboost to find suitable weights. Chattopadhyay proposes Conditional Probability based Multi-source Domain adaptation (CP-MDA) and Two Stage Weighting Framework for Multi-source Domain adaptation (2SW-MDA) in [62] for multiple source transfer learning. The CP-MDA and 2SW-MDA decide the weights for different source domains based on their distance between the target domain. See [62; 63; 64] for related work. These methods are not suitable for problems with limited labeled samples because the information loss due to the re-sampling is more severe in problems with limited samples. The second category, feature-based transfer learning, proceeds by finding a transformation of the inputs so that the transformed input features from the two domains become closer. The most important work of this field is the Transfer Component Analysis (TCA) [65], which tries to find a suitable Reproducing Kernel Hilbert Space (RKHS) to minimize the Maximum Mean Discrepancy (MMD) between the input features in this space. Based on TCA, Duan proposes domain transfer multiple kernel learning in [64] to add the multiple kernel structure into TCA. Long puts forward a Joint Component Analysis (JCA) [66] to simultaneously correct for the marginal and conditional distribution differences. Pan gives a Spectral Feature Alignment (SFA) in [67] to discover a new feature representation by identifying the domain-specific and domain-independent features. See [68; 69; 70] for more related work. However, deciding a suitable transformation can be difficult and domain-dependent [58].

In this work, we adopt the general idea of the iterative self-labeling domain adaptation (ISDA) method [71] and propose a novel method, called the renewing self-labeling domain adaptation (Re-ISDA). The original ISDA method is a domain adaptation method for classification problems. In

this work, we propose a new framework, under which a version of the ISDA method is applicable for regression problems, by formulating the regression problem as a dynamic programming model with uniformly stable algorithms [72]. We also provide a greedy algorithm to solve this dynamic programming model efficiently, which proceeds by renewing all labels of testing samples labeled in former steps. This renewing step helps avoid a potential issue of the ISDA that the possible mis-labeled samples by a weak classifier in the initial stage of the iterative learning can cause serious harm to the subsequent learning process, which is also called mis-labeling issue [73; 74]. In our synthetic numerical examples, the proposed method outperforms three prevailing domain adaptation approaches. We also apply the proposed Re-ISDA method to a spine motion prediction problem. We find that the performance of Re-ISDA is superior to the three prevailing domain adaptation approaches.

2. PROJECTION PURSUIT GAUSSIAN PROCESS REGRESSION*

2.1 Introduction

In this chapter, we will introduce the proposed projection pursuit Gaussian process regression in detail. The remaining part of this chapter is organized as follows. In Section 2.2, we review the background of Gaussian process regression with isotropic and additive Matérn correlation functions. In Section 2.3, we introduce the proposed methodology, called the projection pursuit Gaussian process regression. An algorithm of the proposed method is given at the end of Section 2.3. In Section 2.4 and 2.5, we conduct simulation studies to demonstrate the use of the proposed method, and show that the proposed method outperforms some existing methods. In Section 2.6, we shows that the performance of the proposed method is satisfactory through a real-world application.

2.2 Review on Gaussian process regression

In this section, we review a simple version of the Gaussian process emulation [2]. Let Z be a stationary Gaussian process on $\mathbb{R}^d$ with mean zero, variance σ^2, and correlation function Φ. Given scattered evaluations $(x_1, Z(x_1)), \ldots, (x_n, Z(x_n))$, one can reconstruct Z using its conditional expectation

$$\hat{Z}(x) := \mathbb{E}(Z(x)|Z(x_1), \ldots, Z(x_n)) = r^T(x)K^{-1}Y, \tag{2.1}$$

for $x \in \mathbb{R}^d$, where $r(x) := (\Phi(x - x_1), \ldots, \Phi(x - x_n))^T, K = (\Phi(x_j - x_k))_{jk}$ and $Y = (Z(x_1), \ldots, Z(x_n))^T$.

2.2.1 Curse of dimensionality in Gaussian process regression with isotropic Matérn correlation

Curse of dimensionality is one of the fundamental challenges in various high-dimensional statistical and machine learning problems. In this section, we review how the curse of dimensionality

*This work is an Accepted Manuscript of an article published by Taylor & Francis in IISE Transactions on 30 Aug. 2022, available online: https://doi.org/10.1080/24725854.2022.2121882.

can affect the prediction performance of Gaussian process regression.

The prediction error of the Gaussian process regression is

$$Z(x) - \hat{Z}(x) = Z(x) - \mathbb{E}(Z(x)|Z(x_1), \ldots, Z(x_n)),$$

which is a function of x. In [57], the author studies the rate of convergence of the prediction error under different function norms, under the assumption that the Gaussian process has an isotropic Matérn correlation function [2], defined as

$$\Phi(x; \nu, \phi) = \frac{1}{\Gamma(\nu)2^{\nu-1}}(2\sqrt{\nu}\phi\|x\|)^{\nu} K_{\nu}(2\sqrt{\nu}\phi\|x\|), \tag{2.2}$$

where $\nu > 0$ is the *smoothness parameter*, K_{ν} is the modified Bessel function of the second kind, $\phi > 0$ is the scale parameter.

To explain the curse of dimensionality issue posed by the isotropic Matérn correlation functions, we refer to Theorem 3.3 of [57], which states a lower bound of the maximum of the prediction error of an isotropic Gaussian process. For simplicity, we consider the expected maximum prediction error. Suppose the input region of interest is Ω, and then the expected maximum prediction error is $\mathbb{E}\sup_{x\in\Omega}|Z(x) - \hat{Z}(x)|$. Here the expectation is taken over the randomness of the Gaussian process $Z(\cdot)$. Theorem 3.3 of [57] implies

$$\mathbb{E}\sup_{x\in\Omega}|Z(x) - \hat{Z}(x)| \geq C\sigma n^{-\nu/d}\sqrt{\log n}, \tag{2.3}$$

for a constant C independent of n, σ and the choice of the experimental design.

The lower bound in (2.3) shows that the uniform error of a Gaussian process regression predictor with an isotropic Matérn correlation is no less than a multiple of $n^{-\nu/d}\sqrt{\log n}$. This rate grows dramatically as d increases with a fixed ν. Therefore, when a Gaussian process model with an isotropic Matérn correlation is considered, its prediction suffers from the curse of dimensionality, in the sense that, for a high-dimensional problem, acquiring extra data points cannot improve the

prediction accuracy as effectively as in lower-dimensional problems.

In Gaussian process regression, the curse of dimensionality is inevitable if the underlying function is indeed a realization of a Gaussian process with isotropic Matérn correlation. The reason behind this is that the reproducing kernel Hilbert spaces generated by these correlation functions are too large in high-dimensional circumstances. Fortunately, in most real applications, we confront much "simpler" high-dimensional functions. These functions admit certain "sparse representation", and therefore, at least theoretically, can be recovered at a much higher rate of convergence. In Section 2.2.2, we examine a special and simple structure of this kind.

2.2.2 Additive models: accuracy and limitations

A scalable Gaussian process regression approach proceeds by equipping an additive correlation function. Denote $x = (x_{(1)}, \ldots, x_{(d)})$. We consider the following function:

$$\Phi(x) = \frac{1}{d} \sum_{j=1}^{d} \Phi_1(x_{(j)}), \tag{2.4}$$

where Φ_1 denotes a one-dimensional correlation function. It is easily seen that Φ is positive definite if Φ_1 is positive definite. Thus one can consider Gaussian process models with correlation (2.4). This approach is called the additive Gaussian process regression [25; 26; 27].

Compared to isotropic models, additive models are much more scalable to the dimensionality. It can be shown that the rate of convergence of the uniform error is independent of d. Specifically, if Φ_1 is a Matérn correlation function with smoothness ν, the uniform prediction error in (2.3) can have a rate of convergence $O(n^{-\nu}\sqrt{\log n})$; see our theoretical results in the Supplementary Materials.

Despite the above advantages, the limitations of additive models are also evident. Only additive functions, i.e., the functions that can be decomposed as the sum of functions such that each of them relies on only one entry of x, can be accurately reconstructed. This assumption is *not* true for most of the practical problems. Consider a two-dimensional input (x, y). A simple non-additive function is $f(x, y) = xy + x^2$. Figure 2.1 shows that the additive model cannot fit this function well, while

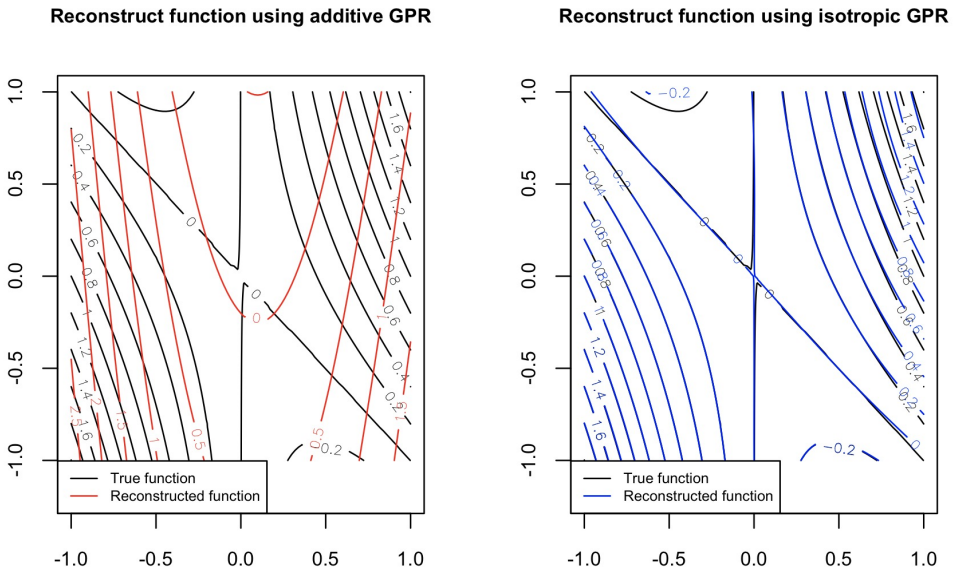


Figure 2.1: Contour plots of $f(x, y) = xy + x^2$ and the reconstructed functions by additive and isotropic Gaussian process regression (GPR) using a same 25-point random design between -1 and 1. It can be seen that the isotropic model has a much better prediction performance.

the isotropic model works in this case.

2.3 Projection pursuit Gaussian process regression

In this section, we propose a general approach to reconstruct multi-dimensional functions that admits more complicated sparse representations. To this end, we consider a model which is more flexible than additive Gaussian process models. Specifically, we employ the projection pursuit regression method [31] to model the underlying function as

$$y(x) = f(w_1^T x, w_2^T x, \ldots, w_M^T x), \tag{2.5}$$

where $w_1, \ldots, w_M$ are unknown vectors, M is a positive integer, and f is an additive function in the sense that f can be written as

$$f(w_1^T x, w_2^T x, \ldots, w_M^T x) = f_1(w_1^T x) + f_2(w_2^T x) + \cdots + f_M(w_M^T x), \tag{2.6}$$

with unknown univariate functions $f_1, \ldots, f_M$. In other words, this model first applies a linear transformation on the input space, and then use an additive function to fit the responses.

A projection pursuit model can be represented by a four-layer network shown in Figure 2.2, which is similar to a neural network model. Neural networks have been widely used to enhance the precision of nonparametric regression [6; 75; 5; 76]; deep neural networks are employed in [77] and [78] to reduce the dimension of data; In [79], the author combines neural networks with Gaussian process regression method to tackle multi-task problems. The main difference between the projection pursuit method and neural networks lies in the activation functions. In neural networks, the activation functions are chosen as fixed function, such as rectified linear unit (ReLU) functions. In contrast, the projection pursuit method uses estimated activation functions. In this work, we call the two hidden layers the *transformation layers*.

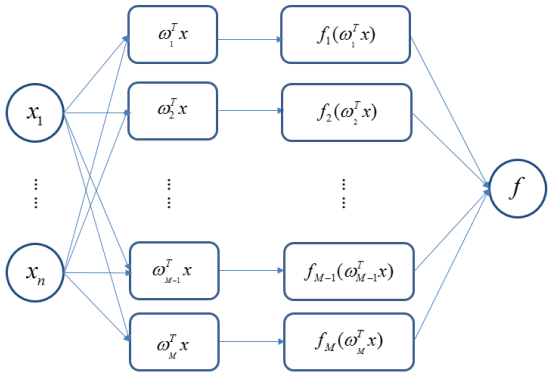


Figure 2.2: Network structure of PPGPR.

When $M = 1$, the projection pursuit model reduces to a single index model, which provides a parsimonious way to implement multivariate non-parametric regression. By imposing suitable priors on the parameters, [80], [81] and [82] use the Bayesian approach to estimate the parameters of the single index model. In [83], a dimension reduction method is applied to choose the number of nodes and then the link function is estimated using Gaussian process regression. In this work, we consider projection pursuit models with $M \gg 1$, which are much more flexible than single index models.

Given a sufficiently large M, it is known that the projection pursuit model can approximate any continuous function arbitrarily well [84]. For example, the non-additive function $f(x, y) = xy + x^2$ can be represented by projection pursuit as shown in Figure 2.3. Figure 2.3 also shows that the representation is not unique.

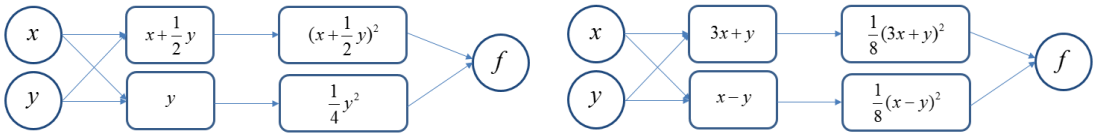


Figure 2.3: Two different representations of $f(x, y) = xy + x^2$ via projection pursuit.

The non-uniqueness of the projected pursuit representation suggests that each of the "directions" w_i may not be essential. In contrast, these vectors exhibit a "synergistic effect", so that they need to be estimated jointly. Consider the example shown in Figure 2.3 (a). Taking the direction $x + y/2$ along is not helpful in obtaining the underlying function $xy + x^2$; this direction makes sense only when it is paired by the direction y. This phenomenon differs from the classical results in linear models, in which the significant directions (usually defined by the principal components) are fixed, and their importance is ordered by the corresponding eigenvalues.

Understanding this difference between the linear and nonlinear models helps build a better projection pursuit regression model. Traditionally, the projection pursuit method is usually regarded

as a dimension reduction approach [32; 33], and greedy algorithms are usually applied to identify w_i's [33; 85; 86]. These strategies have the following deficiencies: 1) it is often hard to accurately approximate the underlying functions through dimension reduction ($M \ll d$). For example, the function $f(x, y) = xy + x^2$ cannot be recovered through a one-dimensional factor. 2) Greedy algorithms, which proceed by picking the current "most significant" direction in each step, cannot perform well when there is no order of importance in the directions, as in the example shown in Figure 2.3. In this work, we propose a method, which conducts a dimension expansion ($M \geq d$) to improve the approximation power substantially.

When $M \geq d$, the projection pursuit model is in general non-identifiable; see Figure 2.3 for an example. The learning outcome on w_i's are meaningless, and we only focus on the prediction of the underlying response at untried input points. Our numerical experience shows that as long as M is large enough, the prediction performance of the proposed method is not heavily dependent on the specific value of M. We recommend choosing M close to, but slightly less than the sample size n.

In this work, we propose a novel approach, called the projection pursuit Gaussian process regression (PPGPR). To reconstruct the underlying function, we need to: 1) estimate the weight parameters $w = (w_1, w_2, ..., w_M)$; 2) reconstruct the combination function f given w using Gaussian process regression [2; 10]. Recall that the design matrix is denoted as $X = (x_1, x_2, ..., x_n)^T$, $x_i \in \mathbb{R}^d$ for $i = 1, 2, ..., n$, and the response as $Y = (f(x_1), f(x_2), ..., f(x_n))^T$. Now we employ the idea of Gaussian process regression to assume that f is a realization of a Gaussian process. Specifically, we assume that the Gaussian process has mean zero and an additive correlation function (2.4). We believe that the mean zero assumption is not too restrictive because the model is already non-identifiable.

The training of the proposed method proceeds by an iterative approach. First, we choose an initial weight parameter w. Then we compute the initial correlation matrix $K_\omega = \left(\frac{1}{M}\sum_{k=1}^{M} \Phi(w_k^T(x_i - x_j))\right)_{ij}$

based on the initial ω. Next, we invoke (3.1) to reconstruct the underlying function f as

$$\hat{f}(x) = r^T(w^T x)(K_\omega + \delta I)^{-1}Y, \tag{2.7}$$

where δ is a nugget term to enhance the numerical stability.

Our goal is to seek for w^* which maximizes the log-likelihood function of Gaussian Process Regression [2], that is,

$$\min_w(l(w)) = \min_w(Y^T(K_w + \delta I)^{-1}Y + \log\det(K_w + \delta I)). \tag{2.8}$$

We refer $l(w)$ to the *model loss*. The gradient of $l(w)$ with respect to w_k is

$$\frac{\partial l(w)}{\partial w_k} = -\frac{1}{M}\sum_{i=1}^{n}\sum_{j=1}^{n}(Y^T K_w^{-1}\frac{\partial K_w}{\partial w_k}K_w^{-1}Y + Tr(K_w^{-1}))(x_i - x_j)^T, \tag{2.9}$$

for $k = 1, 2, ..., M$. The derivative of the matrix K_w can be computed using the following facts. The derivative of the Matérn correlation function is [87]

$$\frac{\partial}{\partial x}\Phi(x;\nu,\phi) = -\frac{2\nu x}{\phi^2(\nu-1)}\Phi\left(\sqrt{\frac{\nu}{\nu-1}}x;\nu-1,\phi\right),$$

and the derivative of the Gaussian correlation function is

$$\frac{\partial}{\partial x}\Phi(x;\phi) = -\frac{|x|}{\phi^2}\exp\left(-\frac{x^2}{2\phi^2}\right).$$

Then the gradient decent method can be applied here to find the minimizer via iteratively updating

$$w_k \leftarrow w_k - \eta\frac{\partial l(w)}{\partial w_k},$$

where η is the step length for the gradient descent algorithm, and is referred to the *learning rate* in the rest of this article.

When the algorithm converges or a stopping criterion is met, one can again reconstruct the underlying function using (2.7). Algorithm 1 lists the detailed steps of the proposed training method, each iteration (epoch) includes calculating the gradient for all weights and renewing the weights. To avoid overfitting, early-stopping criterion [88] should be implemented when choosing P (the number of epochs).

Besides P, there are other hyper-parameters in the proposed methodology, including M, η and the hyper-parameters of the covariance function. We refer the activity of adjusting these parameters to the *tuning process*. Below is a list of our general recommendations for tuning.

- The proposed method does not use the ML estimators [2] to estimate the hyper-parameters of the GP covariance because the ML estimators are likely to overfit with relatively small sample sizes [2].
- Determining a proper learning rate η through cross-validation such that it maintains a stable training process (i.e., the model loss decreases neither too sharply nor too slowly);
- Increasing the size of representation nodes M until the performance on the testing points starts to deteriorate. In practice, we recommend considering M in the range $[4d, 8d]$ in a d-dimensional problem;
- Adopting early stopping policies [88] in the training process when choosing P to avoid overfitting;
- Using cross-validation to choose the hyper-parameters of the covariance function.

More discussion regarding the tuning process is provided through a numerical study in Section 2.4.1.

2.4 Simulation Studies

In this section, we examine the performance of the proposed method via simulation studies. Based on four numerical experiments, we will provide some guidelines for parameter tuning for

Algorithm 1 Training steps for transformation weight w

Input: design matrix $X = (x_1, x_2, ..., x_n)$, response $Y = (y_1, y_2, ..., y_n)$, initialized weight $w = (w_1, w_2, ..., w_M)$, correlation function Φ, learning rate η, number of iterations P

Output: transformation weight w

for p in $1 : P$ **do**
  $X' \leftarrow w^T X$
  $K_w \leftarrow \Phi(X', X')$
  for k in $1 : M$ **do**
    $grad_k \leftarrow -\frac{1}{M} \sum_{i=1}^{n} \sum_{j=1}^{n} (Y^T K_w^{-1} \frac{\partial K_w}{\partial w_k} K_w^{-1} Y + Tr(K_w^{-1}))(x_i - x_j)^T$
    $w_k \leftarrow w_k - \eta \cdot grad_k$
  end for
end for

PPGPR in Section 2.4.1. In Section 2.4.2, we compare the proposed method with some other prevailing algorithms and show the advantages of the proposed method.

2.4.1 Choice of tuning parameters

In this section, we study how the choice of the hyper-parameters of PPGPR can affect its prediction performance. Recall that the hyper-parameters include the learning rate η, the size of nodes M in the transformation layers, the number of epochs (iterations) P, the choice of the correlation function (Matérn or Gaussian) and smoothness parameter ν if a Matérn correlation is used.

In the rest of this subsection, we will use the Borehole function [89] as the test function to study the performance of the proposed PPGPR under different choices of hyper-parameters. The Borehole function is defined as

$$y = \frac{2\pi T_u (H_u - H_l))}{\log(\frac{r}{r_w})[1 + \frac{T_u}{T_l} + \frac{2LT_u}{\log(\frac{r}{r_w}) r_w^2 K_w}]},$$

with the ranges for the eight variables given by $r_w \in (0.05, 0.15)$, $r \in (100, 50000)$, $T_u \in (63070, 115600)$, $H_u \in (900, 1110)$, $T_l \in (63.1, 116)$, $H_l \in (700, 820)$, $L \in (1120, 1680)$ and $K_w \in (9855, 12045)$. A Halton sequence[†] [90] with 40 samples are used as the training set inputs

[†]Halton sequence are deterministic low discrepancy sequences used to generate points in space for numerical

and 500 random samples are used as the testing set inputs. We consider different choices of the tuning parameters and compare the corresponding prediction performance in terms of the mean absolute percentage error (MAPE) [91]:

$$MAPE = \frac{1}{n}\sum_{i=1}^{n}\left|\frac{\hat{y}_i - y_i}{y_i}\right|, \tag{2.10}$$

where $n = 500$ is the size of testing samples; $\hat{y}$ and y denote the predictive value and true value of a testing sample, respectively.

The details of the numerical experiments are described in Sections 2.4.1.1-2.4.1.3. We choose 10^{-6} as the nugget term of (2.8) in this section to avoid some numerical instability, see [92] for more guideline for choosing nugget terms.

2.4.1.1 Learning rate η and number of representation nodes M

In this experiment, a Matérn correlation function with $\nu = 2.5$ is used and training epochs $P = 150$. We examine the performance of PPGPR under different learning rates and different node sizes in the transformation layers.

Figure 2.4 shows the MAPE of PPGPR under different learning rate with respect to the size of representation nodes. It can be seen that when $\eta = 10^{-10}$, the MAPE is much higher than those in the other three situations. For $\eta = 10^{-8}$, the model reaches its best performance when $M = 28$. The models with $M = 35$ have lower MAPE when $\eta = 10^{-7}$ and $\eta = 10^{-9}$. In general, the models with $\eta = 10^{-9}$ perform slightly better and more stably.

According to [84], the PPGPR model can approximate any continuous functions as $M \to \infty$. This explains why the performance of PPGPR grows as M increases when M is small. However, when M is above 35, the MAPE becomes worse for most of the curves in Figure 2.4, which may be due to overfitting because there are too many hidden nodes. In practice, we suggest employing cross-validation to select the optimal M.

Figure 2.5 shows four curves generated with a common initial w and different learning rates

experiments. The Halton sequences can be generated efficiently by an R package "SDraw".

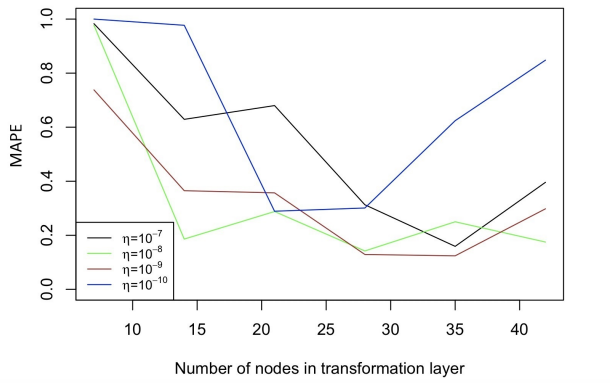


Figure 2.4: MAPE under different learning rates and size of nodes in transformation layers

when $M = 35$. Each of them shows the relationship between the model loss defined in (2.10) and the number of iteration. From Figure 2.5, we find that, 10^{-10} is too low as a learning rate, because the model loss is still high (about 5×10^5) even after 100 iterations. This observation is also confirmed by the MAPE results in Figure 2.4, in which the MAPE for $M = 35$ corresponding to $\eta = 10^{-10}$ is much higher than those in the other ones. The model loss curves for the other three learning rates are similar. We believe that the choice of $\eta = 10^{-9}$ gives a slightly better result than those given by $\eta = 10^{-8}$ or $\eta = 10^{-7}$, because the model loss curve under $\eta = 10^{-9}$ decreases more smoothly than the other two, which implies a more stable learning process [93]. According to [94], a flat minima might have higher generalization than sharp minima. Besides, a too small model loss after training might result in overfitting which will be shown in Section 2.4.1.3. Figure 2.4 also implies that $\eta = 10^{-9}$ gives the best MAPE when $M = 35$. In practice, the optimal learning rate relies on the underlying function. Therefore, we recommend tuning η via cross-validation.

2.4.1.2 Effects of correlation function type and parameters

In this experiment we examine the performance of PPGPR under different correlation functions and smoothness parameters with $\eta = 10^{-9}$ and $P = 150$.

Figure 2.6 shows the MAPE for PPGPR with the Matérn correlation functions under different

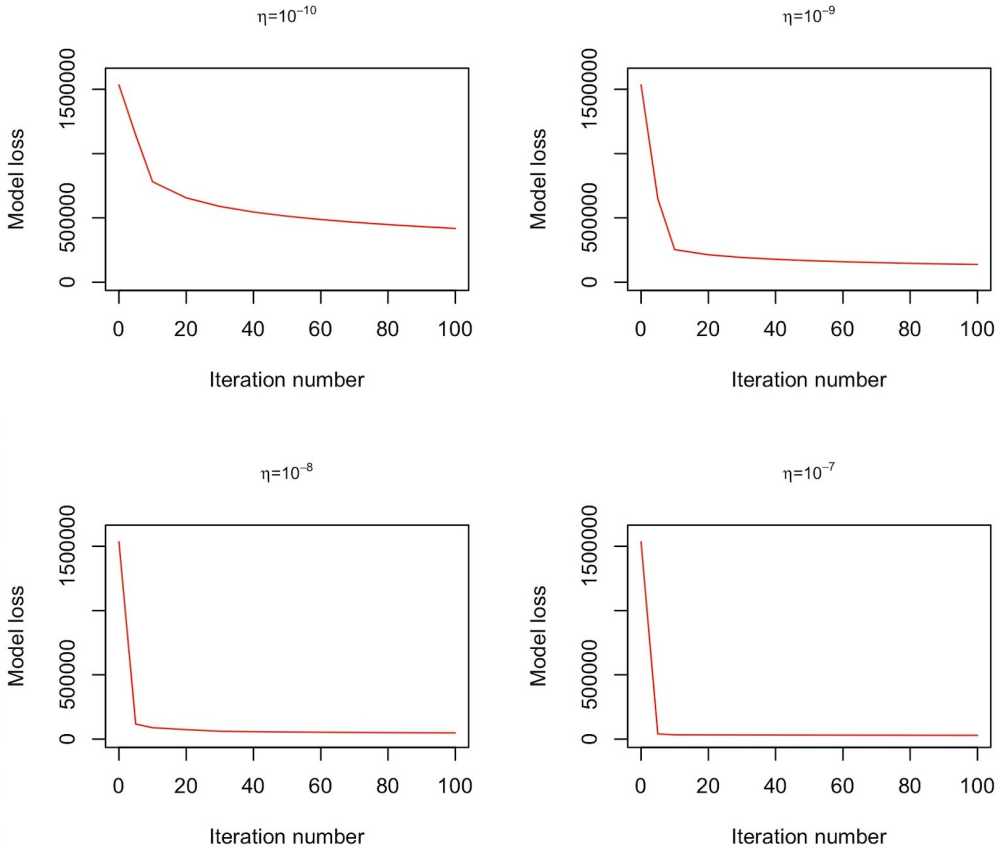


Figure 2.5: Model loss with different learning rate

M and ν with $\phi = 1$. It can be seen that when $\nu = 2.5$ (green line), the model performs better than other choices. Under $\nu = 2.5$, the best prediction performance is achieved when $M = 35$. Generally, with a larger ν, the reconstructed function would be smoother, which may lead to overfitting; with a smaller ν, the reconstructed function would be less smooth, which may result in instability or underfitting. Figure 2.7 shows the MAPE for PPGPR with Gaussian correlation functions under different M and ϕ. We can see that, when $M = 35$, the green line ($\phi = 0.5$) reaches its lowest MAPE, which is slightly better than the MAPE under other M and ϕ in this experiment. This experiment shows that Matérn correlation functions with $\nu = 2.5$ seem to be an appropriate choice of the correlation functions. We also recommend using cross-validation to determine the optimal correlation function if computational resource permits. Table 4.1 shows the numerical values of the lowest MAPE of PPGPR under the above Matérn and Gaussian correlation

functions.

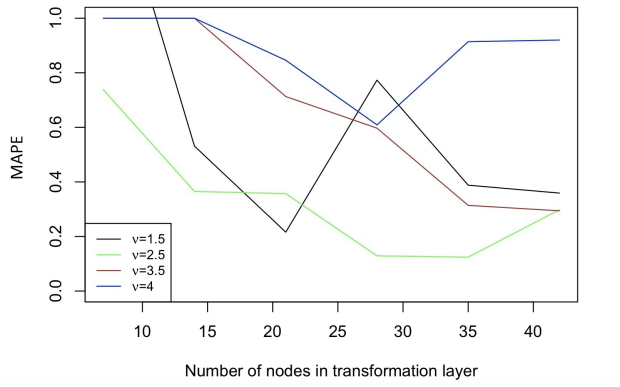


Figure 2.6: MAPE under different ν and M for Matérn correlation functions

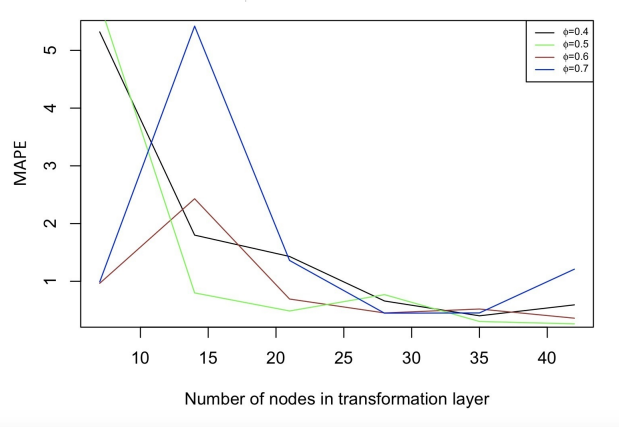


Figure 2.7: MAPE under different ϕ and M for Gaussian correlation functions

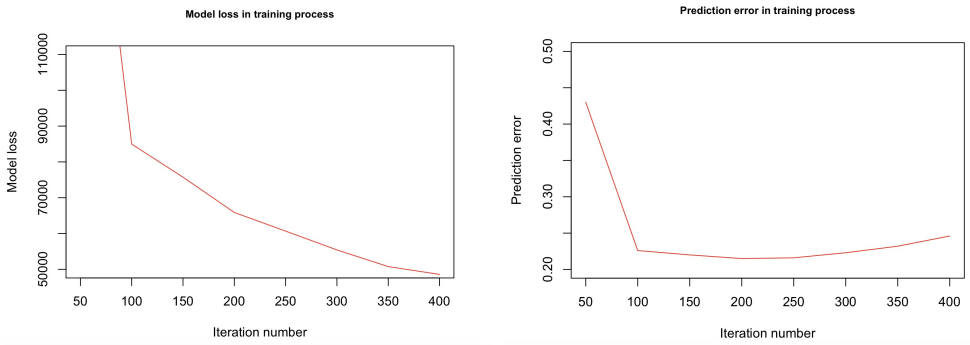
2.4.1.3 Training epochs P

In this experiment the model loss and the prediction error of PPGPR during the training process are monitored. Here we use a Matérn correlation function with $\nu = 2.5$ and $\eta = 10^{-8}$, $M = 21$.

Table 2.1: Best MAPE for PPGPR with Gaussian and Matérn correlation functions

	M	ϕ	ν	MAPE
Matérn	35	1	2.5	0.124
Gaussian	42	0.5	-	0.263

Figures 2.8a and 2.8b plot the model loss and prediction error against the training epochs, respectively. We can see from Figure 2.8a that the model loss is monotonically decreasing as M increase. This implies that the proposed gradient descent algorithm works in a desired way. However, Figure 2.8b shows that the prediction error is *not* a monotonic function in the model loss. The model achieves its best performance when $P = 220$, and as P further increases, the prediction error increases. This phenomenon has been observed in other network structures such as neural networks. In a typical neural network training process, a slower early-stopping criterion with 4% (i.e., stopping the training process when the relative generalization improvement is less than 4%) could be used to avoid overfitting caused by an overshot training process [88]. We suggest adopting a similar approach in training the proposed PPGPR model.



(a) Model loss in training process (b) Precision during the training process

Figure 2.8: Model loss and precision during the training process

2.4.2 Numerical comparisons

In this section we compare PPGPR with GPR, Neural Network (NN), SVR (Supporting Vector Regression) and GBDT (Gradient Boosting Decision Trees) using three test functions: OTL circuit function [95], Borehole function [89] and Wingweight function [96]. The training set is chosen as Halton series [90] with length $p = 5 \times d$, where d is the dimension of input space, and the size of testing set is 500. The implementation details of five methods for these experiments are shown below:

- SVR: Matérn correlation with $\nu = 2.5$;
- GBDT: Gaussian distribution and 100 trees;
- NN (deep learning): for OTL circuit function, it has structure $(6, 12, 24, 12, 1)$ (meaning the node size of input layer is 6, the second layer has 12 nodes and so on) with learning rate 0.01 and 150 epochs. For Borehole function it has structure $(8, 16, 32, 1)$ with learning rate 0.01 and 150 epochs. For Wingweight function it has structure $(10, 20, 30, 20, 1)$ with learning rate 0.1 and 200 epochs;
- GPR (with isotropic and product correlation functions): We use the Dicekriging package [97] with isotropic and product Matérn correlation and smoothness $\nu = 2.5$ to compute the predictive results. The product correlation is defined as $K(x) = \prod_{i=1}^{n} \Phi_1(x_{(j)})$, where $\Phi_1(x_{(j)})$ is the same as in (2.4);
- PPGPR: for OTL circuit function, Matérn correlation with $\nu = 2.5$, $M = 42$, $\eta = 10^{-9}$, $P = 150$, for Borehole function, Matérn correlation with $\nu = 2.5$, $M = 35$, $\eta = 10^{-9}$, $P = 150$, for Wingweight function, Matérn correlation with $\nu = 2.5$, $M = 35$, $\eta = 10^{-10}$, $P = 150$.

The MAPE of each method above is given in Table 4.2. One can easily find that the performances of SVR and GBDT are inferior, which can be explained because these approaches may require more training data [98; 99]. We try our best to tune the parameters of the NN, in order

to obtain the best results we can achieve. It is worth noting that the parameter tuning for NN is time-consuming. In contrast, the tuning process of PPGPR is much easier because it has only one hidden layer. Also, PPGPR outperforms NN in all three experiments. Moreover, PPGPR can beat GPR with isotropic and product correlation functions because the curse of dimensionality has less impact on PPGPR. Note that GPR with isotropic kernels performs worse than GPR with product kernels. This is not surprising in view of the slow rate of convergence for isotropic kernels shown in Section 2.1. The rate of convergence for product kernels under a general condition is not well-established, but they are known to outperform the isotropic kernels in high-dimensional circumstances [100].

Table 2.2: MAPEs of Supporting Vector Regression (SVR), Gradient Boosting Decision Trees (GBDT), Neural Network (NN), Gaussian Process Regression (GPR) with isotropic and product correlations and Projection Pursuit Gaussian Process Regression (PPGPR) for three functions. PPGPR outperforms all other methods.

	OTL circuit($d = 6$)	Borehole($d = 8$)	Wingweight($d = 10$)
SVR	0.121	0.792	0.127
GBDT	0.130	0.407	0.142
NN	0.0334	0.222	0.240
GPR(iso)	0.0182	0.204	0.0224
GPR(pro)	0.0162	0.134	0.0199
PPGPR	**0.0139**	**0.124**	**0.0184**

Additionally, we compare the performance of PPGPR and GPR with product kernel when the size of training set changes. Figure 2.9 shows the MAPEs of the proposed PPGPR and GPR with product kernels for OTL function, when the number of training set varies. It can be seen that when the size of training samples is less than 48 ($8d$) the PPGPR works much better than GPR. When the size of training set increases the MAPEs of both methods decrease and the MAPE of

GPR decreases faster than PPGPR. The results in Figure 2.9 can prove that the proposed PPGPR is highly suitable for a sparse learning environment but when enough traning samples are available the PPGPR is not recommended.

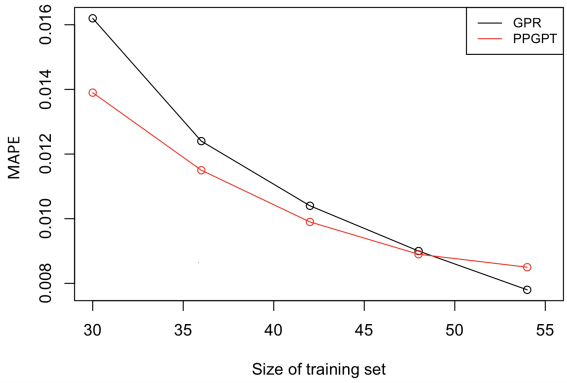


Figure 2.9: MAPEs of GPR and PPGPR with different size of training set for OTL function

2.5 More numerical studies

We conduct more numerical studies. The results show the proposed PPGPR method outperforms the alternatives under different design and test functions. Due to the space limitation, we place these results in the Appendix A.

2.6 Approximated heat exchanger case study

In this section, we apply the proposed method PPGPR on a heat exchanger (HE) application introduced by [45]. The HE data in [45] have two fidelities, known as detailed data (high fidelity) and approximated data (low fidelity). Because this work considers only the surrogate modeling for single-fidelity datasets, we use only the approximated data to implement the proposed method. The main objective of this application is to explore the impact of four factors, including the mass flow rate of entry air m, the temperature of entry air T_{in}, the temperature of the heat source T_{wall} and the solid material thermal conductivity M, on the total rate of steady state heat transfer y_a achieved by a heat exchanger. All design points live in a hypercube whose upper and lower bounds

are shown in Table 2.3. We follow the treatment in [45] to partition the dataset into a training set of 64 samples and a testing set of 14 samples.

Table 2.3: Assumed design range for HE case

	$m(kg/s)$	$T_{in}(K)$	$k(W/mK)$	$T_{wall}(K)$
Lower Bound	0.00055	270.00	202.4	330
Upper Bound	0.001	303.15	360.0	400

In this section, we compare the performance of GPR with isotropic and product correlations, Transformed Approximately Additive Gaussian Process Regression (TAAG) proposed in [14], and the proposed PPGPR. In [14] the performance was assessed in terms of the root mean square error (RMSE), defined as

$$RMSE = \sqrt{\frac{1}{n}\sum_{n}^{i=1}(\hat{y} - y)^2}, \tag{2.11}$$

where $\hat{y}$ is the predicted value and y means the true value for every sample, n stands for the size of testing set. Therefore, we consider RMSE of all the candidate methods. The implementation details of the these methods are as follows:

- GPR (with isotropic and product correlation functions): we use the Dicekriging package [97] with isotropic and product Matérn correlation and smoothness $\nu = 2.5$ to compute the predictive results;
- TAAG: the result in [14] is refered here;
- PPGPR: we use the Matérn correlation with smoothness $\nu = 2.5$ and $M = 28$, $\eta = 10^{-9}$.

The results of these three methods are shown in Table 3.5. It can be seen that the proposed method has a lower RMSE than other methods.

Table 2.4: RMSEs of Gaussian Process Regression (GPR) with isotropic and product correlations, Transformed Approximately Additive Gaussian Process Regression (TAAG) and the Projection Pursuit Gaussian Process Regression (PPGPR)

	GPR(iso)	GPR(pro)	TAAG	PPGPR
RMSE	4.20	4.26	2.08	1.82

In the Appendix A, we present an upper bound of the uniform prediction error of Gaussian process regression with an additive correlation function, which implies a promising rate of convergence of additive Gaussian process models. Also, more numerical studies are included in Appendix A.

3. FIXED-BUDGET OPTIMAL DESIGNS FOR MULTI-FIDELITY COMPUTER EXPERIMENTS

3.1 Introduction

In this chapter, we will introduce the proposed multi-level Gaussian process optimal design method and analysis the corresponding cost theoretically. This chapter is organized as follows. Section 3.2 reviews the autoregressive models. In Section 3.4, we introduce the formulation of the modified autoregressive model and propose the MLGP design method. In Section 3.5, we give the detailed steps for implementing the MLGP method in applications. In Section 3.6, some numerical studies are conducted to show the superiority of the proposed MLGP.

3.2 Review on Gaussian process regression and autoregressive models

In this section, we review the Gaussian process regression and the autoregressive models for multi-fidelity computer experiments.

3.2.1 Gaussian process regression

Let Z be a Gaussian process on $\mathbb{R}^d$ with mean zero and covariance function $r(x_1, x_2)$. We call Z stationary [2] if r can be expressed as a function of $x_1 - x_2$, i.e., $r(x_1, x_2) = \sigma^2\Phi(x_1 - x_2)$, where σ^2 is the variance, and Φ is a correlation function with $\Phi(0) = 1$; otherwise, we call it nonstationary. Given scattered evaluations $(x_1, Z(x_1)), \ldots, (x_n, Z(x_n))$, the Gaussian process regression method (also known as kriging) estimates $Z(x)$ for any $x \in \mathbb{R}^d$ using the conditional expectation

$$\hat{Z}(x) := E(Z(x)|Z(x_1), \ldots, Z(x_n)) = a^T(x)R^{-1}Y, \tag{3.1}$$

where $a(x) := (r(x - x_1), \ldots, r(x - x_n))^T$, $R = (r(x_j - x_k))_{jk}$ and $Y = (Z(x_1), \ldots, Z(x_n))^T$. In the area of computer experiments, (3.1) is used to reconstruct the response function $Z(x)$, where $\{x_1, \ldots, x_n\}$ denotes the set of designs points, $\{Z(x_1), \ldots, Z(x_n)\}$ the corresponding response, and x an untried point.

3.2.2 Autoregressive model

[7] proposes a model to fit multi-fidelity computer experiments using Gaussian process regression. The main objective of this model is to incorporate the computer outputs with different fidelity levels and predict the response value of the computer code at an untried point of the highest fidelity level.

Suppose we have K levels of computer codes, with increasing accuracies, denoted as $y_1, \ldots, y_K$. [7] considers the following autoregressive model to link each pair of consecutive y_i's:

$$y_i(x) = \rho_{i-1} y_{i-1}(x) + \delta_i(x), \quad for \quad i = 2, \ldots, K, \tag{3.2}$$

where ρ_i's are known or unknown autoregressive coefficients, and y_1 and δ_t's as mutually independent Gaussian processes.

This autoregressive model contains some hyper-parameters. The parameter estimation is usually done using maximum likelihood [101] or Bayesian methods [7]. Both approaches involve certain iterative algorithms and thus are expensive in computation. Here we skip the details of the hyper-parameters estimation because they are irrelevant to our main topic. We only consider the prediction for the computer outputs at the highest fidelity given the hyper-parameters. The autoregressive model has been widely used in real-world applications for multi-fidelity computer experiments e.g., simulation oil pressure [7], designing of linear cellular alloy [36] and prediction of bandgaps [37].

3.3 Autoregressive models for computer code with infinite fidelity levels

In this work, we focus on computer experiments with infinite fidelity levels. There are many examples of this kind. For example, in finite element analysis, the accuracy is governed by the mesh density, which, at least conceptually, has infinite levels. Other examples include the step length in a finite difference algorithm, the number of iteration in an iterative algorithm, etc. Such a parameter is referred to as a "tuning parameter" in [39].

Now let us use the finite element method (FEM) as an example to introduce our intuition. FEM

generally uses piecewise linear functions over prespecified local patches to approximate general functions, such as the partial differential equation (PDE) solutions. The number of local patches are governed by the chosen mesh size. In multi-fidelity computer experiments, one may consider a sequence of consecutively refined meshes. Figure 3.1 shows the finite elements corresponding to the first three mesh size in the set of refining meshes $\{1/2, 1/4, 1/8, ..., 1/2^n, ...\}$ in a unit square.

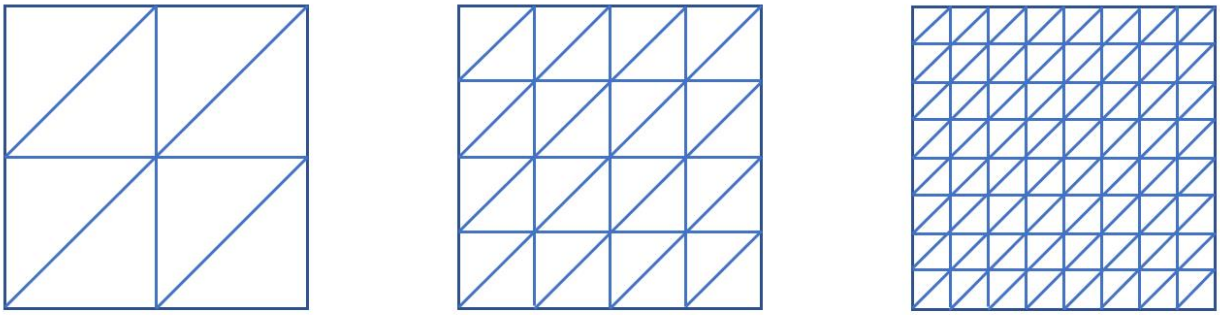


Figure 3.1: Consecutively refined meshes. The number of the triangle local patches quadruples as the mesh size is halved.

The question here is: *what is a suitable configuration of autoregressive models for the above FEM example which can potentially have infinite fidelity levels?* Clearly, we need to make some modifications to (3.2). The most obvious one follows from the fact that in the above example, there are infinite fidelity levels. That is, conceptually, we should have computer response functions y_i for $i = 1, \ldots, K, \ldots$. Next, we presume that the FEM solution converges to the exact solution of the PDE as the mesh size goes to zero, which is usually the case. Then the underlying quantity of interest is the function $y_\infty := \lim_{i\to\infty} y_i$. To guarantee the existence of this limit, more constraints must be imposed in the autoregressive model (3.2). First, we must have $\lim_{i\to\infty} \rho_i = 1$ as the existence of y_∞ requires $\lim_{i\to\infty}(y_i - y_{i-1}) = 0$. In this work, we assume $\rho_i = 1$ for all i for simplicity. Then we have the expression

$$y_\infty = y_1 + \sum_{i=2}^{\infty} \delta_i.$$

Note that var(y_∞)=var(y_1)+$\sum_{i=2}^{\infty}$ var (δ_i). To ensure that y_∞ has a finite variance, δ_i must have decreasing variances. Here we model $\delta_i \sim GP(0, \lambda^{2i}\sigma^2\Phi_i)$ for some correlation function Φ_i, variance σ^2 and $0 < \lambda < 1$. The exponential decay in the variance can be justified because the mesh size in Figure 3.1 also decays exponentially as the refinement continues. It is also fair to assume that all the $\delta_i' s$ share the same correlation function Φ because we we have not seen the data yet.

We now define the modified version of the autoregressive model as,

$$y_i(x) = y_{i-1}(x) + \delta_i(x), \quad for \quad i = 0, \ldots, K, \ldots, \tag{3.3}$$

where $y_{-1} = 0$, and $\delta_i \sim GP(0, \lambda^{2i}\sigma^2\Phi)$ are mutually independent.

3.4 Multiple fidelity designs based on multi-level Gaussian processes

Many statisticians have considered the problem of integrating the computer responses from multiple accuracy (also known as fidelity) levels to make predictions. To achieve this goal, the first step is to choose a pre-determined model to model the specific multi-fidelity computer experiment. Next step is to construct designs of experiments for the chosen model. In this work, we propose the multi-level Gaussian processes (MLGP) method to construct optimal designs for the autoregressive model.

At the same time, the number of finite elements increases exponentially as the mesh refines iteratively so that the resulted cost for each untried point also increases exponentially. In view of this, we assume the cost for each sample in the i-th level as $C_i \leq C_a a^i$, where C_a is a constant, $a > 1$ is the cost ratio. Additionally, the difference of the approximation values given by successive mesh sizes decreases as mesh size gets finer.

3.4.1 A fixed precision optimal design for multi-fidelity models

In this section, we introduce the problem setting for constructing optimal designs for the modified autoregressive model with a fixed precision. As discussed in the above section, we use y_∞ to denote the underlying function. (3.3) allows us to reduce the multi-fidelity problem to a sequence of single-fidelity problems. If we can reconstruct δ_i (i.e., $y_i - y_{i-1}$) for each i, we are able to reconstruct y_∞ using the summation of all these constructed functions. To reconstruct δ_i, a simple idea is to evaluate the function δ_i over a set of selected points, denoted as $\mathcal{X}_i$ correspondingly, via the Gaussian process regression. We assume the underlying function in the K-th layer, i.e., $y_K = \sum_{i=0}^{K} \delta_i$, is sufficiently accurate. One can reconstruct the y_K by $\hat{y}_K$ defined as (3.4) to approximate y_∞.

$$\hat{y}_K(x) = E\left[\sum_{i=0}^{K} \delta_i(x) \Big| y_K(\mathcal{X}_K), ..., y_0(\mathcal{X}_0)\right] \tag{3.4}$$

Denote the desired accuracy as ϵ ($\epsilon < 1$). Then we can formulate the general objective function for constructing optimal designs for the modified autoregressive model as

$$\min_{n_i, K} \quad \sum_{i=0}^{K} n_i C_i \tag{3.5}$$

$$\text{s.t.} \quad E\left\|y_\infty - \hat{y}_K\right\|_{L^2}^2 \leq \epsilon^2, \tag{3.6}$$

where $n_i \in \mathbb{N}$ denotes the sample size of $\mathcal{X}_i$, and $\left\|y_\infty - \hat{y}_K\right\|_{L^2}^2$ is the expected squared predictive error of $\hat{y}_K$. This objective function can be interpreted as: we want to minimize the cost caused by the designs $\{n_i, K\}$ which satisfies the accuracy requirement.

3.4.2 MLGP method

The main difficulties for solving (3.5) include: 1) y_∞ is unknown; 2) $\hat{y}_K$ shown in (3.4) does not have a closed form under an arbitrary design; 3) the integer optimization problem is NP-hard. In this section, we make some approximations and restrictions to make (3.5) solvable. In detail,

to make the expected squared error in (3.6) tractable, we only consider nested designs [102; 41] to simplify $\hat{y}_K$. Additionally, we replace the left hand side of (3.6) by its upper bound to get an approximated solution. We also relax n_i's and K to be real numbers at first to search for the optimal solution, and then round down them as the final optimal solution.

The widely used nested designs can refine the low-accuracy experiments with the high-accuracy experiments to obtain a better model because the response is also available in low-accuracy experiments at the same design points [41]. Denote $\mathcal{X}_0, \mathcal{X}_1, \ldots, \mathcal{X}_K$ as a sequence of nested sets of points, then the correlation between them can be formulated as $\mathcal{X}_K \subset \cdots \subset \mathcal{X}_1 \subset \mathcal{X}_0$. A good property of the nested design given in Theorem 3.4.1 helps simplify the approximation (3.4).

Theorem 3.4.1. *Suppose $\delta_i \overset{i.i.d}{\sim} GP(0, \sigma^2\Phi)$, $i = 0, 1, \ldots, K$, where σ^2 is the variance and Φ is the correlation function. Let $\mathcal{X}_K \subset \cdots \subset \mathcal{X}_1 \subset \mathcal{X}_0$ be a sequence of nested sets of points. Then for each x, the estimator $\hat{y}_K(x)$ defined in (3.4) admits the expression*

$$\hat{y}_K(x) = \sum_{i=0}^{K} E\left[(y_i - y_{i-1})(x) \middle| (y_i - y_{i-1})(\mathcal{X}_i)\right]. \tag{3.7}$$

Theorem 3.4.1 shows that if a nested design is used, Gaussian process regression for an autoregressive model can be done by solving a set of single-fidelity Gaussian process regression problems. Theorem 3.4.1 inspires us to use a nested design for the multi-fidelity computer experiment. Let $\mathcal{X}_K \subset \cdots \subset \mathcal{X}_1 \subset \mathcal{X}_0$ be a sequence of nested sets of points, where X_i consists of n_i points. Therefore the multilevel interpolation approximation (3.4) can be rewritten as (3.7).

The next step is to construct the nest designs for the (3.5). We can get a upper bound for the expected predictive error in (3.6) by:

$$E\left\|y_\infty - \hat{y}_K\right\|_{L^2}^2 \leq E\left\|y_\infty - y_K\right\|_{L^2}^2 + E\left\|y_K - \hat{y}_K\right\|_{L^2}^2. \tag{3.8}$$

To achieve the desired accuracy, we assume it is sufficient to bound both error contributions in

right hand side of (3.8) by $\epsilon^2/2$, i.e., we convert the constrain (3.6) into

$$E \left\| y_\infty - y_K \right\|_{L^2}^2 < \epsilon^2/2, \tag{3.9}$$

and

$$E \left\| y_K - \hat{y}_K \right\|_{L^2}^2 < \epsilon^2/2. \tag{3.10}$$

where (3.9) bounds the error caused by using y_K to approximate y_∞ and (3.10) bounds the error caused by the Gaussian process interpolation. Next we will handle these two parts separately.

First, we focus on (3.9). Given the modified autoregressive model with shrinking variance, it is easy to computer the upper bound of the error of the left hand side of (3.9).

Theorem 3.4.2. *Assume an infinite series $\delta_i \overset{i.i.d}{\sim} GP(0, \lambda^{2i}\sigma^2\Phi)$, $i = 0, 1, \ldots, K, \ldots$, $0 < \lambda^2 < 1$ and σ^2 is the variance. Denote y_∞ and y_K as the summation of the whole series and the summation of the first $(k+1)$ terms, respectively, then we have*

$$E \left\| y_\infty - y_K \right\|_{L^2}^2 \leq C_l^2 \lambda^{2K} \sigma^2, \tag{3.11}$$

where $C_l = \sqrt{C_s}/(1-\lambda)$ is a constant, C_s is constant meaning the area of the region of x.

Based on Theorem 3.4.2, it is sufficient to bound the upper bound of the left hand side of (3.11) by $\epsilon^2/2$ to make (3.9) hold, that is,

$$E \left\| y_\infty - y_K \right\|_{L^2}^2 \leq C_l^2 \lambda^{2K} \sigma^2 \leq \epsilon^2/2.$$

Then we can choose

$$K = \left\lceil \log_{\lambda^2} \frac{\epsilon^2}{2C_l^2\sigma^2} \right\rceil, \tag{3.12}$$

as the total number of levels K, where $\lceil a \rceil$ denotes the smallest integer which is larger or equal to a.

Next, we focus on (3.10) which is about the error caused by Gaussian process interpolation. Let's first focus on the error caused by original Gaussian process interpolation. Given scattered evaluations $(x_1, Z(x_1)), \ldots, (x_n, Z(x_n))$ where the design is quasi-uniform, if we use the Matérn kernel to reconstruct the underlying function $Z(x)$, then there exist two positive constants p and q such that [57]:

$$E \left\| Z(x) - \hat{Z}(x) \right\|_{L^2}^2 < p n^{-2\nu_0/d}, \tag{3.13}$$

and

$$E \left\| Z(x) - \hat{Z}(x) \right\|_{L^2}^2 > q n^{-2\nu_0/d}, \tag{3.14}$$

where $\hat{Z}(x)$ is the reconstructed function, ν is the smoothness parameter of the Matern kernel and d is the dimension of the design. Here we make the similar assumption as the [57] to our nested designs. Before introduce the assumption, we make some definitions. Assume a set of design points $X = \{x_1, x_2, ..., x_n\} \subset \Omega$, define the fill distance as

$$h_{X,\Omega} = \sup_{x \in \Omega} \inf_{x_j \in X} ||x - x_j||,$$

and the separation radius as

$$q_X = \min_{1 \leq j \neq k \leq n} ||x_j - x_k||/2.$$

$\mathcal{X}_K \subset ... \subset \mathcal{X}_1 \subset \mathcal{X}_0$ is a sequence of nested sets of points, denote the corresponding fill distance and separation radius as $h_{X_i,\Omega}$ and q_{X_i}. There exists a constant A such that $h_{X_i,\Omega}/q_{X_i} \leq A$ holds for all i.

Based on the error bound of the original Gaussian process regression, we propose the upper bound of the interpolation error in the modified autoregressive model in Theorem 3.4.3.

Theorem 3.4.3. *Assume a infinite series* $\delta_i \overset{i.i.d}{\sim} GP(0, \lambda^{2i}\sigma^2\Phi)$. *Denote* y_K *as the summation of*

the first $(K+1)$ *terms, and* $\hat{y}_K$ *defined in (3.7) as the emulation of* y_K *based on the set of points* $\mathcal{X}_i$ *and Matern kernel, then under Assumption 3.4.2 we have*

$$E\left\|y_K - \hat{y}_K\right\|_{L^2}^2 \leq \sum_{i=0}^{K} p\lambda^{2i}\sigma^2 n_i^{-2\nu/d}, \tag{3.15}$$

where n_i *is the number of points of* X_i*,* d *is the number of dimension,* p *is a constant,* ν *is the smoothness parameter of* Φ*.*

Based on Theorem 3.4.3, it is sufficient to bound the upper bound of the left hand side of (3.15) by $\epsilon^2/2$ to make (3.10) hold. Then we can approximate the solution of objective function (3.5) with (3.6) by solving a relaxed version, i.e.,

$$\begin{aligned} \min_{n_i} \quad & \sum_{i=0}^{K} n_i C_i \\ \text{s.t.} \quad & \sum_{i=0}^{K} p\lambda^{2i}\sigma^2 n_i^{-2\nu/d} \leq \epsilon^2/2 \end{aligned} \tag{3.16}$$

where $K = \lceil 2\log_{\lambda^2} \frac{\epsilon}{2C_l\sigma} \rceil$. One can see that (3.16) is an integer programming problem which is NP-hard. In view of this, we consider a relaxed version of it. First we assume n_i's can choose any real number, then (3.16) becomes a convex optimization problem whose optimal solution is

$$n_i = \left(\frac{\epsilon^2}{2p\sigma^2 S}\right)^{-\frac{d}{2\nu}} \left(\frac{a^i}{\lambda^{2i}}\right)^{-\frac{d}{d+2\nu}}, \quad i = 0, ..., K, \tag{3.17}$$

where $S = \sum_{i=0}^{K} (a^i)^{\frac{\nu}{d+2\nu}} (\lambda^{2i})^{\frac{d}{d+2\nu}}$, p is a constant. The proof of (3.17) is given in the Supplementary.

Note that $\lceil n_i \rceil$'s also locate in the feasible region of (3.16). We regard the $\lceil n_i \rceil$'s shown in (3.18) as a nearly optimal solution to (3.16).

$$n_i = \left\lceil \left(\frac{\epsilon^2}{2p\sigma^2 S}\right)^{-\frac{d}{2\nu}} \left(\frac{a^i}{\lambda^{2i}}\right)^{-\frac{d}{d+2\nu}} \right\rceil, \quad i = 0, ..., K, \tag{3.18}$$

In summary, we consider the designs generated based on (3.18) and (3.12) as the optimal designs for the modified autoregressive model. We will discuss the implement details in Section 3.5.

3.4.3 Comparison with single-fidelity experiments

In this section, we will show that the MLGP needs less cost than any single-level method to achieve a given accuracy provided that the accuracy level is small enough. Let us denote $a = \exp(\alpha)$ and $\lambda^2 = \exp(-\beta)$ for convenience ($\alpha, \beta > 0$). In this section, we use the symbol "$\lesssim$" to represent that the left hand side of this symbol is less than the right hand side multiplied by a constant depending only on Φ. First, we give the upper bound for the total cost if we use the MLGP method to reach a fixed accuracy ϵ.

Theorem 3.4.4. *Assume $y_i(x) = y_{i-1}(x) + \delta_i(x)$, for $i = 0, \ldots, K, \ldots$, where $\delta_i \overset{i.i.d}{\sim} GP(0, \lambda^{2i}\sigma^2\Phi)$, $\lambda^2 < 1$ and $y_{-1} = 0$. If we use the design (3.12) and (3.18) to reach a fixed accuracy ϵ, then the upper bound for the total cost is summarized as*

$$C_\epsilon \lesssim \begin{cases} \epsilon^{-d/\nu}, & 2\alpha\nu < d\beta \\ \epsilon^{-d/\nu} |\ln \epsilon^2|^{\frac{d+2\nu}{2\nu}}, & 2\alpha\nu = d\beta \\ \epsilon^{-2\alpha/\beta}, & 2\alpha\nu > d\beta \end{cases} \tag{3.19}$$

Next, we discuss the lower bound for the total cost if we implement the single-fidelity method to reach the accuracy ϵ.

Theorem 3.4.5. *Assume $y_i(x) = y_{i-1}(x) + \delta_i(x)$, for $i = 0, \ldots, K, \ldots$, where $\delta_i \overset{i.i.d}{\sim} GP(0, \lambda^{2i}\sigma^2\Phi)$, $\lambda^2 < 1$ and $y_{-1} = 0$. If we only spread design samples in one layer to reach a fixed accuracy ϵ, then the lower bound for the total cost is*

$$C_\epsilon^{SL} \gtrsim \epsilon^{-(\frac{d}{\nu} + \frac{2\alpha}{\beta})}. \tag{3.20}$$

The proofs of Theorem 3.4.4 and Theorem 3.4.5 are given in the Supplementary.

3.5 Fixed budget designs

For many real applications, a more realistic assumption is that the total budget, rather than the accuracy, is fixed. In this section, we will introduce how to implement the MLGP design method given a fix budget.

First, assume we know the values for the hyper-parameters λ^2, correlation function, and ν. In practice, λ^2 can be decided based on expert knowledge. For example, we can choose the decreasing ratio of the mesh sizes as λ^2 in the FEM case. The correlation function and the corresponding parameter(s) need to be decided based on the smoothness of the underlying function, see [103; 104] for more information.

In order to obtain the designs based on (3.12) and (3.18) under a certain budget B, we need to have the exact values for the parameters p, C_l, σ and the desired accuracy ϵ. These parameters are often hard to decide in real-world applications. However, the number of available layers is often determined in specific application. Based on this, we just assume K is determined. Please note that this assumption does not mean we need to use all the layers in our designs because n_i can be very small for a certain layer if the MLGP method thinks that the corresponding layer is not needed.

The next thing is to calculate n_i. Denote $\tau := p\sigma^2/\epsilon^2$ and $\eta := p/C_l^2$, then we rewrite (3.18) as

$$n_i = \left\lceil (2\tau S)^{\frac{d}{2\nu}} (\frac{a^i}{\lambda^{2i}})^{-\frac{d}{d+2\nu}} \right\rceil, \quad i = 0, 1, ..., K \tag{3.21}$$

where $\tau = \frac{\eta}{2\lambda^{2K}}$.

In order to avoid the estimation of those constants (p, C_l, σ and ϵ), one can tune η to get the corresponding optimal design under the fixed budget B. By choosing a value for η, we can get a value for τ which leads to corresponding $n_i, i = 0, ..., K$ according to (3.21). The resulted design that triggers the highest cost and meets the budget requirement can be chosen as the optimal design n_i^*'s. In applications, the user can try different values for η in a pre-specified set (e.g., the integers from 1 to 100) to search for the optimal design n_i^*'s.

Note that the design n_i^*'s do not use all the budget in most cases because we need to choose an integer for n_i^*'s when implementing (3.21). Denote the unallocated budget as

$$B_K = B - \sum_{i=0}^{K} n_i^* C_i. \tag{3.22}$$

The final step is to re-allocate B_K to modify the design $n_i^*, i = 0, ..., K$. We implement a greedy algorithm to solve the following objective function (3.23) to re-allocate the budget to further reduce the error in the left hand side of the constrain of 3.16.

$$\min_{n_i^{**}} \quad \sum_{i=0}^{K} \lambda^{2i}\sigma^2((n_i^{**})^{-2\nu/d} - (n_i^*)^{-2\nu/d}) \tag{3.23}$$

$$\text{s.t.} \quad \sum_{i=0}^{K}(n_i^{**} - n_i^*)C_i \leq B_K \tag{3.24}$$

Then we get the final optimal designs $n_i^{**}, i = 0, ..., K$.

Denote a K level design as a vector $N = [n_1, ..., n_K]$, and a function $G(N^{**}) = \sum_{i=0}^{K} \lambda^{2i}\sigma^2((n_i^{**})^{-2\nu/d} - (n_i^*)^{-2\nu/d})$. Also denote a vector with the i-th element equal to 1 and other elements equal to 0 (i.e., $1^i = [0, ..., 1, ...0]$) as one sample in the i-th layer. Now the detailed steps of the re-allocating starting from the design N^* are as follow:

- Search i exhaustively in the range $[0, K]$ to minimize $G(N^* + 1^i)$ on the condition that $C_i \leq B_K$;
- Add one sample to the i-th layer of N^*, that is, $N^* = N^* + 1^i$;
- Calculate the unallocated budget $B_K = B_K - C_i$;
- Repeat the above steps until $B_K = 0$ or B_K cannot afford any additional sample.

Remark 1. In this section, we regard some hyper-parameters as known (i.e., λ and smoothness parameter ν) to derive the optimal designs, but we may not know the true values for them in real-world applications. However, we will prove that the designs generated by the proposed MLGP can

also beat others even with the misspecified values of these hyperparameters. See Section 3.6.2 for more details.

Given the detailed steps for calculating the n_i^{**}'s for each layer, we still need to decide how to spread these samples in each layer. Please recall that the error bound in Theorem 3.4.3 only holds when Assumption 3.4.2 holds. But it is very hard to maintain this assumption in applications rigorously because it involves a lot of optimization steps. we consider approximately space-filling sequences, such as Halton sequences [90] which is a type of low-discrepancy sequences. Halton sequences enjoy the following two advantages. First, it can give designs of any sample sizes. Second, the sequences have analytical expressions which are easy to compute.

3.5.1 Examples

In this section, we visualize the designs generated by the MLGP under different budgets or different λ^2 in $[0, 1]^2$. We choose $a = 1$ and $C_a = 1$ for the experiments in this section.

In the first experiment, we use $B = 192$, $K = 2$ and Matérn correlation function with $\nu = 1.25$ to generate designs under four different λ^2 using the MLGP. Tabel 3.1 and Figure 3.2 show the design structures when λ^2 changes from $1/8$ to $3/4$. Note that the column "Design structure" shows the number of samples from the lowest to the highest layer. For example, the design $20, 7, 3$ means a nested design with 20 samples (empty circles in 3.2) in the 0-th layer, 7 samples (filled circles in 3.2) in the 1st layer and 3 samples (filled triangles in 3.2) in the 2nd layer. The subplot a, b, c and d are corresponding to $\lambda^2 = 1/8$, $\lambda^2 = 1/4$, $\lambda^2 = 1/2$ and $\lambda^2 = 3/4$, respectively. It can be seen that the MLGP allocates more budget on the higher layers with the λ^2 increases. This phenomenon exactly matches the changing of the true function. This is because the variance of δ_i's in 3.7 in the high layers take less proportion in the underlying function y_K if λ^2 is small. In other words, the information of the higher layers plays a less important role in the overall accuracy. As a result, if we want to get a low error as much as possible under a fixed budget, we should pay more budget on the samples in the lower layers and vice versa. This exclusive flexibility of the MLGP can help the users a lot to construct suitable designs when facing different λ^2 while the multi-level and single-level methods can only build designs with fixed proportions.

In the second experiment, we use $K = 2$, $\lambda^2 = 1/2$, Matérn correlation function with $\nu = 1.25$ to generate designs under budgets from 96 to 240 using the MLGP. Table 3.2 and Figure 3.3 show the design structures generated by the MLGP under different B.

In the third experiment, we test our method under large ν's for the Matérn correlation function. We use $K = 3$, $\lambda = 1/2$ and budget $B = 4760$ to generate designs under different ν's from 5 to 50. Table 3.3 shows the designs in this experiment, we can see that our method still works well under large ν's.

Table 3.1: Designs under different λ^2

λ^2	Design structure
$1/8$	48, 16, 5
$1/4$	32, 16, 6
$1/2$	24, 14, 7
$3/4$	16, 12, 8

Table 3.2: Designs under different budgets B

B	Design structure
96	20, 7, 3
144	24, 10, 5
192	24, 14, 7
240	28, 17, 9

Table 3.3: Designs under different budgets B

ν	Design structure
5	24,16,16,7
10	16, 9, 17, 7
25	16, 9, 9, 8
50	16, 9, 9, 8

3.6 Numerical study

In this section, some numerical studies will be conducted to show the superiority of the proposed MLGP. In Section 3.6.1, we will first compare the performance of the proposed MLGP, classical multi-fidelity designs, and single-level designs in 1-dimensional cases when we know all hyper-parameters (λ^2, correlation function, and ν). Then we will test the performance of these three methods if one of the hyper-parameters is misspecified in Section 3.6.2. In Section 3.6.3, we will compare the proposed MLGP with Nested Latin Hypercube designs and single-level designs in a 2-dimensional case. Emulators given by the proposed MLGP yield more accurate results than other methods in all cases.

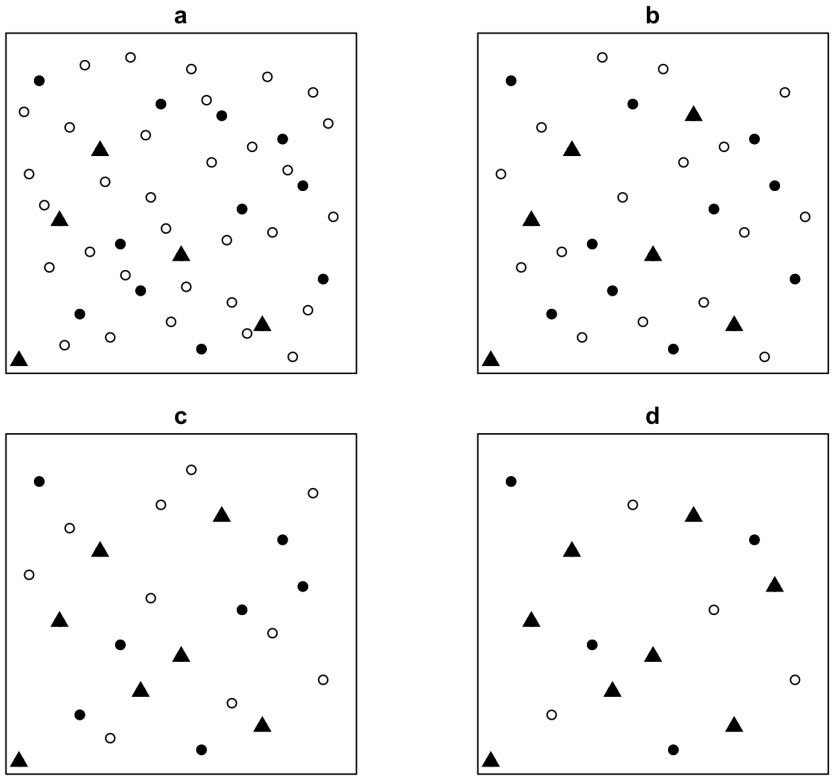


Figure 3.2: Designs generated by MLGP under different λ^2

3.6.1 Numerical study in one-dimensional space

In the section, we prepare a four-level function $f(x)$ (i.e., $K = 3$) defined in $[0, 15]$ as the simulation function. Each level is assumed as an i.i.d Gaussian process with zero mean and Matérn correlation function. We assume $\lambda^2 = 1/3$, the correlation matrix is Matérn correlation function with $\nu = 1.25$. In this experiment, we choose the cost ratio $a = 8$ and the cost for each sample in the 0-th layer to be 1, i.e., $C_a = 1$.

In this experiment, we assume all the true values hyper-parameters are known. We calculate three groups of designs based on the proposed MLGP method, the classical multi-fidelity design method, and the single-level design method under different budgets. After calculating the number of samples in each layer, we generate the corresponding (nested) design samples based on the Halton sequences [90] for all three methods. Then we train a Gaussian process regression (GPR) emulation based on each design to test the prediction accuracy on a testing set with 200 samples.

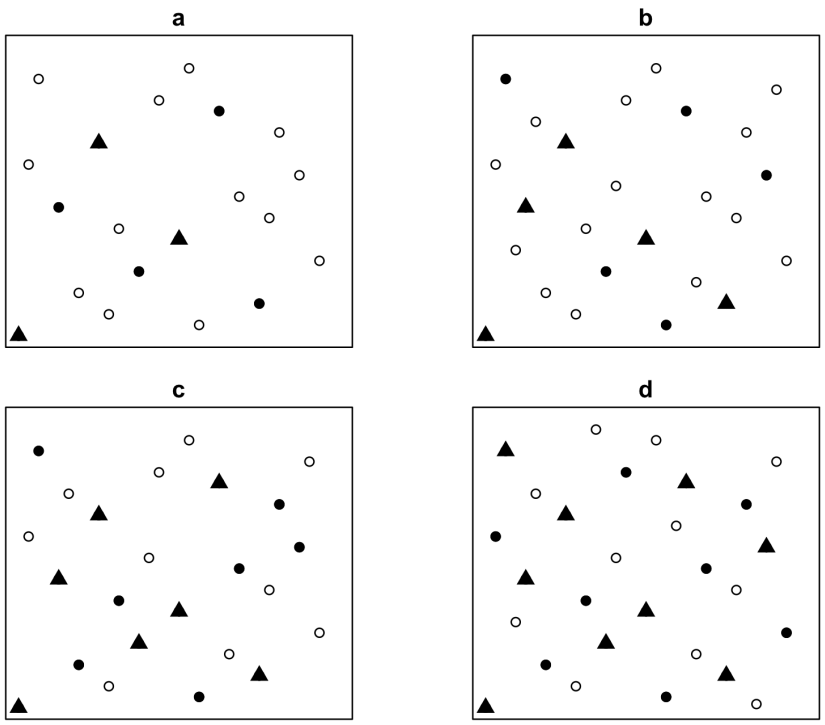


Figure 3.3: Designs generated by MLGP under different budgets

Note that we do not use maximum likelihood estimation (MLE) to estimate the variance and the length parameter. The accuracy of GPR emulations is measured by the root mean squared error (RMSE) defined as $RMSE = \sqrt{N^{-1}\sum_{i=1}^{N}(y_i - \hat{y}_i)^2}$, where $N = 200$, y_i, and $\hat{y}_i$ are the true label and the predictive label for the i-th testing sample, respectively.

In order to make the results more convincing and show the robustness of the proposed MLGP algorithm, we repeat this experiment 30 times starting from different realizations of $f(x)$ and compare the mean RMSE of these methods.

Figure 3.4 shows the average RMSEs of the three methods under different budgets. The blue line shows the performance of the single-level method, and it almost keeps the same even with different budgets. This is because in all cases, the best accuracy of the single-level method is achieved by only sampling at the 2-th layer. The red line and the black line represent the performance of the multi-level method and the MLGP, respectively. One can easily see that the designs given by the MLGP lead to the emulations with the lowest RMSEs compared with other methods, which proves

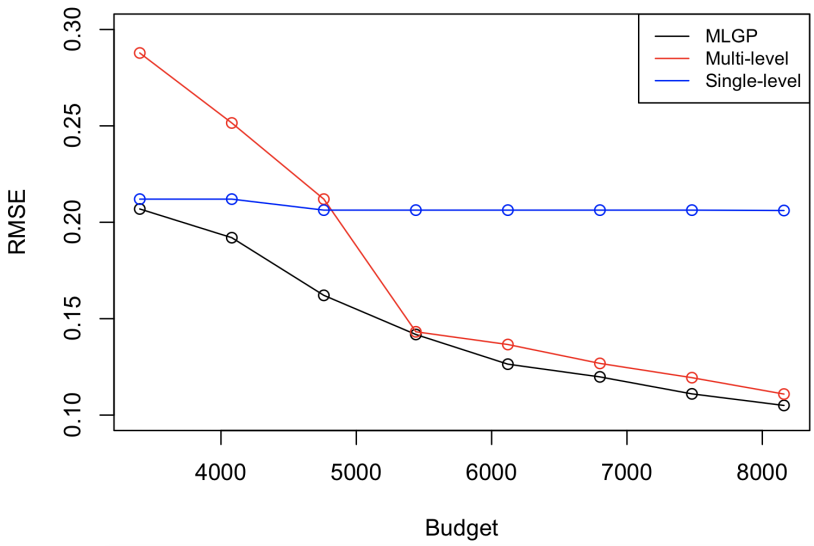


Figure 3.4: RMSEs of the MLGP, classical multi-fidelity method and single-level method for the 1-dimensional case under different budgets.

the superiority of the proposed MLGP.

3.6.2 Numberical study for the situations with misspecified hyper-parameters

We may not get the exact values for all the hyper-parameters in some cases. To show the robustness of the MLGP, we also test the performance of the proposed MLGP when we incorporate a misspecified hyper-parameter. First, we use the same parameters as the first experiment to generate the four-level function $f(x)$, i.e., $K = 3$, $\lambda^2 = 1/3$, Matérn correlation function with $\nu = 1.25$. Also, we choose the cost ratio $a = 8$ and $C_a = 1$. We generate designs based on the proposed MLGP using all the same parameters but a misspecified $\lambda^2 = 1/2$. Then we test the GPR emulation trained by this design and compare the accuracy of the testing set with other methods. Next, we use the hyper-parameters $K = 3$, $\lambda^2 = 1/3$, Matérn correlation function with $\nu = 1.25$ to generate the four-level function $f(x)$ and generate designs based the proposed MLGP using all the same parameters but a misspecified $\nu = 1.5$. GPR emulation is also used to compare the performance of the MLGP and the other two methods.

Table 4.2 shows the results for these two experiments. One can see that the MLGP still gets the lowest error even though the misspecified hyper-parameters are incorporated.

Table 3.4: Mean RMSEs for the proposed MLGP method, classical multi-fidelity design method and single-level design method with misspecified λ^2 or ν

Situation	Method	Average RMSE	Designs structure
misspecified λ^2	MLGP	0.2831	88, 32, 21, 6
	Multi-fidelity	0.3441	56, 28, 14, 7
	Single-level	0.3789	0, 0, 74, 0
misspecified ν	MLGP	0.1652	88, 40, 20, 6
	Multi-fidelity	0.2127	56, 28, 14, 7
	Single level 2	0.2064	0, 0, 74, 0

3.6.3 Case study in two-dimensional space

In this section, we will compare the performance of the proposed MLGP, nested Latin hypercube design (NLHD) and single level design in a two-dimensional case. We prepare a three-level function $f(x)$ (i.e., $K = 2$) defined in $[0, 1]^2$ as the simulation function. Each level is assumed as an i.i,d Gaussian process (GP) with zero mean and Matérn correlation function with $\nu = 1.25$. We assume $\lambda^2 = 1/2$. In this experiment, we choose the cost ratio $a = 4$ and the cost for each sample in the 0-th layer to be 1, i.e., $C_a = 1$.

In this experiment, we assume all the true parameters are known, i.e., $\lambda^2 = 1/2$, the correlation function is Matérn correlation function with $\nu = 1.25$. For the proposed MLGP and single-level methods, we first calculate the design structure and then generate the corresponding (nested) samples using the Halton sequences [90]. As for the NLHD, we use a design given in 3.6. Then we train a GPR emulation based on each design to test the prediction accuracy on a testing set with 500 samples based on the root mean squared error (RMSE). In this experiment, 30 different realizations of $f(x)$ are generated to test the average RMSE of the GPR emulations. Table 3.5 shows the average RMSEs and the design structures of these three methods. We can see that the MLGP yields the lowest RMSE, which proves the superiority of the MLGP in two-dimensional cases.

In Appendix B, we present proofs regarding the theorems in this chapter.

Table 3.5: Results for the proposed MLGP method, Nested Latin hypercube design (NLHD) and single level design method

Method	Average RMSE	Designs structure
MLGP	0.1510	20, 7, 3
Multi-fidelity	0.1735	32, 8, 2
Single-level	0.1658	0, 0, 6

Table 3.6: Nested Latin hypercube designs in two-dimensional space

Level 3	(0.166670,0.863640), (0.712120,0.318180),
Level 2	(0.166670,0.863640), (0.712120,0.318180), (0.590910,0.712120), (0.984850,0.590910), (0.863640,0.984850), (0.045455,0.166670), (0.439390,0.045455), (0.287880,0.469700),
Level 3	(0.166670,0.863640), (0.712120,0.318180), (0.590910,0.712120), (0.984850,0.590910), (0.863640,0.984850), (0.045455,0.166670), (0.439390,0.045455), (0.287880,0.469700), (0.954550,0.409090), (0.833330,0.681820), (0.196970,0.136360), (0.348480,0.924240), (0.681820,0.954550), (0.651520,0.530300), (0.136360,0.378790), (0.530300,0.439390), (0.893940,0.833330), (0.621210,0.075758), (0.318180,0.772730), (0.257580,0.287880), (0.772730,0.106060), (0.075758,0.742420), (0.378790,0.196970), (0.106060,0.560610), (0.560610,0.227270), (0.803030,0.500000), (0.469700,0.621210), (0.227270,0.651520), (0.924240,0.257580), (0.409090,0.348480), (0.500000,0.893940), (0.742420,0.803030),

4. RENEWING ITERATIVE SELF-LABELING DOMAIN ADAPTATION WITH APPLICATION TO THE SPINE MOTION PREDICTION

4.1 Introduction

In this chapter, we will introduce the renewing iterative self-labeling domain adaptation method and apply it to a spine motion prediction case. The remainder of this chapter is organized as follows. We first formulate the problem mathematically in Section 4.2 and review the iterative self-labeling domain adaptation briefly in Section 4.3. Then we establish a new theoretical foundation for ISDA and propose the Re-ISDA in Section 4.4. A numerical example is investigated in Section 4.5. In Section 4.6, we apply the proposed method to a cervical spine motion prediction problem, for which we are supposed to construct an efficient domain adaptation model based on limited labeled samples.

4.2 Problem formulation

We start with a general formulation of domain adaptation, as discussed in [58]. A domain is defined as $D = \{X, P(X)\}$, where X stands for a feature space and $P(X)$ denotes the marginal distribution of these features. Besides, a task T based on a specific domain D is defined as $T = \{Y, P(Y|X)\}$ where Y is the label space and $P(Y|X)$ denotes the conditional distribution. Then a domain adaptation task can be described as [105; 58]: Given a source domain (training set) D_s and learning task T_s based on D_s, domain adaptation aims to help improve the learning of the target predictive function on target domain (testing set) D_t where $X_s = X_t$, $P_s(X) \neq P_t(X)$ and $T_s = T_t$.

In this work, we focus on regression problems in domain adaptation. Assume that the source domain $D_s = \{(x_s^1, y_s^1), (x_s^2, y_s^2), ..., (x_s^q, y_s^q)\}$ and the target domain as $D_t = \{x_t^0, x_t^1, ..., x_t^{p-1}\}$. Besides, choose a labeled sample as a calibration point (x_c, y_c). In Section 4.4.2 we will discuss how to choose this calibration point in detail. Let (X_s, Y_s) and (X_t, Y_t) be the joint distributions of inputs and outputs on the source and the target domains. We suppose that $P(X_s) \neq P(X_t)$, i.e.,

the two data sets have different input distributions. We also assume that the relationship between the input and output, characterized by the conditional distributions, are the same, i.e., $P(Y_s|X_s) = P(Y_t|X_t)$. Let (x_s, y_s) and (x_t, y_t) be data from the source and the target distributions, respectively. Suppose we are given a loss function $L(h; x, y)$, such as the quadratic loss $L(h; x, y) = (y-h(x))^2$. The observed data are independent copies of (x_s, y_s) and independent copies of x_t. The problem of interest is to estimate

$$h_T := \text{argmin}_h \, \mathbb{E}L(h; x_t, y_t).$$

4.3 Review to the iterative self-labeling domain adaptation

[71] propose the ISDA based on the domain adaptation SVM (DASVM) [106] and gives a theoretical foundation for this domain adaptation classification algorithm. The procedure of ISDA is illustrated in Figure 4.1: assume we have some blue points as the training set (source domain) and our mission is to predict the labels of red points (target domain) as shown in the first plot of Figure 4.1. In order to predict the labels for target samples more precisely, ISDA mainly repeats these three steps in its iteration process:

- First, learn a model f_n from the current labeled and pseudo labeled sample set S_n. As shown in the first plot of Figure 4.1, it learns a classifier through the labeled samples (blue points);
- Second, pseudo-label some target samples based on f_n. If we assume it pseudo-labels one target sample each step, one of the target samples will be labeled as Y_1^1 as shown in the second plot of Figure 4.1;
- Then incorporate the newly labeled target sample(s) into the source samples to build up S_{n+1} to progressively modify the current classifier. As shown in the third plot of Figure 4.1, the ISDA will train a new classifier based on the source samples and the newly labeled target sample, and then label a new target sample using the new classifier as Y_2^2.

Note that in original ISDA, k source samples will be eliminated from the source domain at each step based on their distance from the classification boundary. In this work we assume $k = 0$.

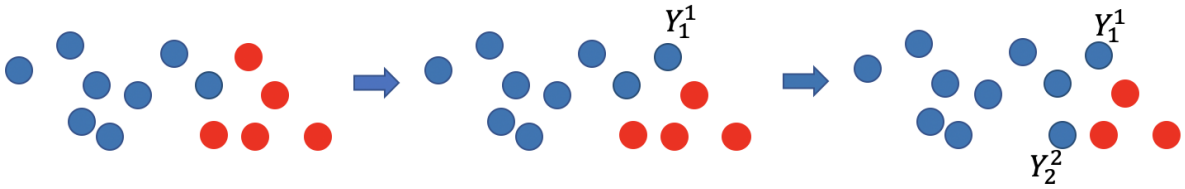


Figure 4.1: Iteration process of ISDA.

Habrard proves that the final classifier will work better for domain adaptation problem if all the classifiers during the iteration work better than random guessing and at least one of these classifier is good enough. However, the possible mis-labeled samples by a weak classifier in the initial stage of the iterative learning can cause serious harm to the subsequent learning process, which is also called mis-labeling issue [73; 74].

4.4 Renewing Iterative Self-labeling Domain adaptation

The proposed Renewing Iterative Self-labeling Domain Adaptation (Re-ISDA) method also follows the general steps of ISDA but it renews all the pseudo labels of target samples at each step to modify their possible error. The proposed Re-ISDA proceeds by iteratively repeating the following three steps:

1. Learn a model f_n from the current labeled and pseudo labeled sample set S_n;

2. Pseudo-label some new target samples and update all the pseudo labels given in previous steps based on f_n;

3. Combine the newly labeled target sample(s) and the target samples with updated labels with the source samples to build up S_{n+1}.

The main steps of Re-ISDA are illustrated in Figure 4.2. First, we learn a predictor through the source samples (blue points). In Figure 4.2, we pseudo-label only one target sample per step. Then the Re-ISDA labels one of the target samples as Y_1^1 which is shown in the second plot of Figure 4.2. Next, as shown in the third plot of Figure 4.2, the ISDA will train a new predictor using the

source samples and the newly labeled target sample with its label Y_1^1. Based on this new predictor, it will label a new target sample using the new predictor as Y_2^2 and update the pseudo-label Y_1^1 by Y_2^1 which means the label for the first target sample in the second iteration. By adding this updating strategy, the Re-ISDA has the ability to modify the possible mis-labels in the previous steps and prevent the mis-labeling issue. It is worth noting that in practice, we may pseudo-label multiple target samples in one step; see Section 4.4.2 for more discussions.

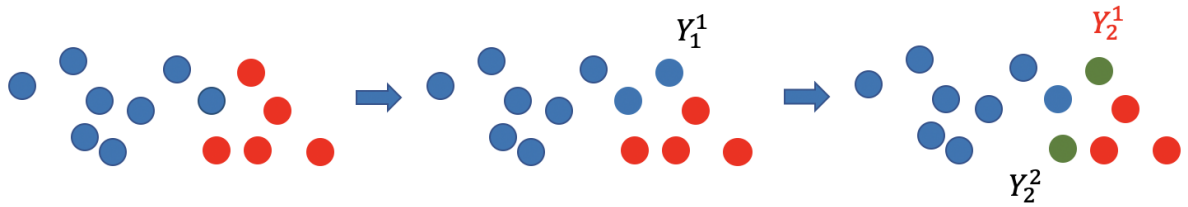


Figure 4.2: Iteration process of Re-ISDA.

In Section 4.4.1, we introduce a theoretical foundation of the Re-ISDA by constructing a dynamic programming framework for the domain adaptation regression problems. Some implementation guidelines for the proposed method are provided in Section 4.4.2.

4.4.1 The proposed methodology

The theoretical foundation of ISDA given by [71] is not applicable in the current context, because it focuses on classification problems but we are interested in regression problems. Before introducing our dynamic programming framework, let us introduce some basic concepts. [72] define the *uniform stable algorithms* as follows:

Definition 1. *An algorithm A has a uniform stability β with respect to the loss function l if* $\forall S \in \{X \times Y\}^m, \forall i \in 1, 2, ..., m$

$$\sup_{(x,y)\in S} |l(h_S(x), y) - l(h_{S^{\setminus i}}(x), y)| < \beta, \tag{4.1}$$

where hypothesis h_S is learned from S and $S^{\setminus i}$ is obtained from S with deleting the i-th sample.

It is also shown that β decreases at the order of magnitude $O(1/m)$ if the algorithm is stable, where m is the size of S. So we can denote it as β_S [72]. According to this definition, we can get an equivalent proposition:

Assume that there exists a function f with uniform stability β_S which can fit a big enough labeled set $S = \{(x_1, y_1), (x_2, y_2), ..., (x_q, y_q)\}$, then for each point $(x, y) \in S$,

$$\sum_{f \in \mathcal{F}} |f(x) - y| < n\beta_0, \tag{4.2}$$

where $\mathcal{F} = \{f = T(s) | s \subseteq S \setminus (x, y)\}$, $f = T(s)$ means f is trained based on the set s, the size of every s is big enough, n is the number of such f, $\beta_0 = max\{\beta_s | s \subseteq S \setminus (x, y)\}$.

We only consider the big enough subsets of S in Proposition 4.4.1 because when the size of s is too small, the corresponding β could be arbitrarily large, which makes the left hand side of (4.2) intractable. If we define the left hand of (4.2) as the *point loss* of a fixed point in S, Proposition 4.4.1 can be interpreted as: if a big enough labeled set can be fitted by a uniformly stable function, then the point loss is small for each fixed point in this set.

In view of Proposition 4.4.1, we have an equivalent statement of the original domain adaptation problem. Let D_{tl} be the labeled target domain. The equivalent problem is to find labels Y_t for D_t such that $D_{tl} \bigcup D_s$ can be fitted by a uniformly stable function, i.e., find labels Y_t to minimize the point loss of a calibration point (x_c, y_c), that is,

$$\min_{Y_t} \sum_{f \in \mathcal{F}} |f(x_c) - y_c|, \tag{4.3}$$

where $\mathcal{F} = \{f = T(s) | s \subseteq D_s \bigcup D_{tl}\}$, D_{tl} is the labeled target domain by Y_t, the size of s is big enough. But the size of $\mathcal{F}$ in (4.4) may be too large, which makes this problem intractable. To address this challenge, we propose an approximation scheme. We define a proper subset of $\mathcal{F}$,

denoted as F. Then instead of solving (4.3), we consider the approximated problem

$$\min_{Y_t} \sum_{f \in F} |f(x_c) - y_c|. \tag{4.4}$$

Next, we fix the order of the target samples and model the problem (4.4) as a dynamic programming problem. First, define state $s_n = \{u_1^n, u_2^n\}$, where u_1 is the set of labeled or pseudo-labeled samples, and u_2 is the set of η newly unlabeled target samples in each state. Also we define action function as a_n, y_2^n determined by a_n is the set of pseudo labels for u_2^n. The loss at each state caused by the action a_n is denoted as

$$r(s_n, a_n) = |f_n(x_c) - y_c|, \tag{4.5}$$

where $f_n = T(u_1^n \bigcup (u_2^n, y_2^n))$. Figure 4.3 shows the first two states of the whole system, where $X_n = \{x_{n\eta}, x_{n\eta+1}, ..., x_{(n+1)\eta-1}\}, n = 0, 1, ..., p/\eta - 1$. For example, at state S_0, u_1^0 represents D_s, u_2^0 stands for X_0, pseudo labels y_2^0 for X_0 is determined by action a_0 and the loss $r(s_0, a_0) = |f_0(x_c) - y_c|$, where $f_0 = T(D_s \bigcup (X_0, y_2^0))$.

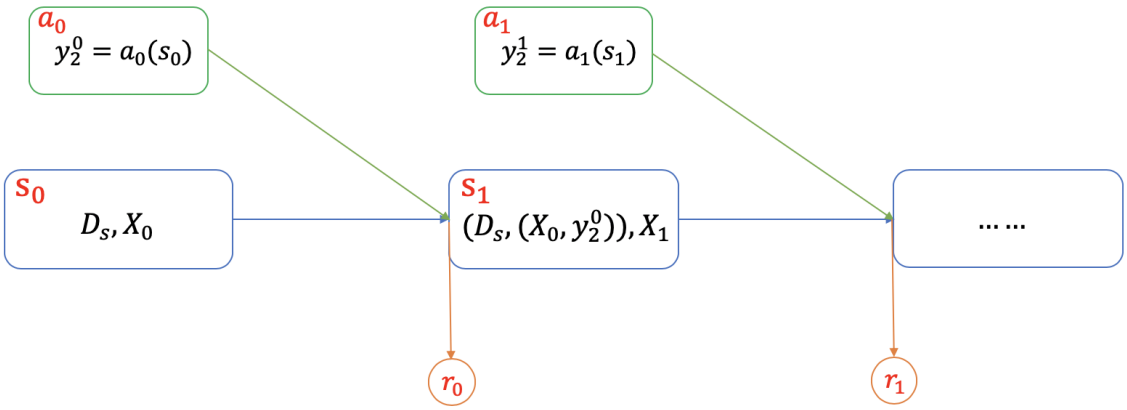


Figure 4.3: First two states of the system.

Our goal is to find the optimal policy $\{a_n\}, n = 0, 1, ..., p/\eta - 1$, that minimizes the sum of

loss, i.e.,

$$\min_{\{a_n\}} \sum_{i=0}^{n} r(s_n, a_n). \tag{4.6}$$

Now we consider problem (4.4) on the set $F := \{f_0, f_1, ..., f_n\}$, $n = 0, 1, \ldots, p/\eta - 1$. Then the problem is equivalent to the dynamic programming problem (4.6).

Solving (4.6) exactly is usually still computationally intractable. Thus it is reasonable to consider approximate algorithms to tackle (4.6). The ISDA can be regarded an simple greedy algorithm to cope with (4.6), that is, to solve

$$\min_{a_n} r(s_n, a_n), \tag{4.7}$$

step by step, for $n = 0, 1, \ldots, p/\eta - 1$ if we assume the action function as follows

$$y_2^n = f_n^*(u_2^n), \tag{4.8}$$

where $f_n^* = T(u_1^n \bigcup (x_c, y_c))$ and T is a stable basic learner. This function means if we need to find suitable labels for u_2^n to minimize the loss in (4.5) we can just use the predictor trained by $\{u_1^n \bigcup (x_c, y_c)\}$ to predict the labels. And the solution to the previous step is assumed as known and fixed in the subsequent steps. The detailed process can be seen as Figure 4.4. At the sate S_0, the pseudo label for X_0 is given by the action a_0 where the predictor f_0^* is trained from $D_s \bigcup (x_c, y_c)$, i.e., $f_n^* = T(u_1^n \bigcup (x_c, y_c))$. At the same time, the action a_0 will result in a loss r_0. Then the iteration continues.

As a simple greedy algorithm, the ISDA has the following deficiency: once a pseudo labeling error occurs, it can never be corrected. If the errors at early iterations are large, the entire downstream process can be highly disturbed, which is also called mis-labeling issue [73; 74]. This phenomenon will be shown in our experiments.

We propose a modification of the ISDA action function, which enables a self-correcting ca-

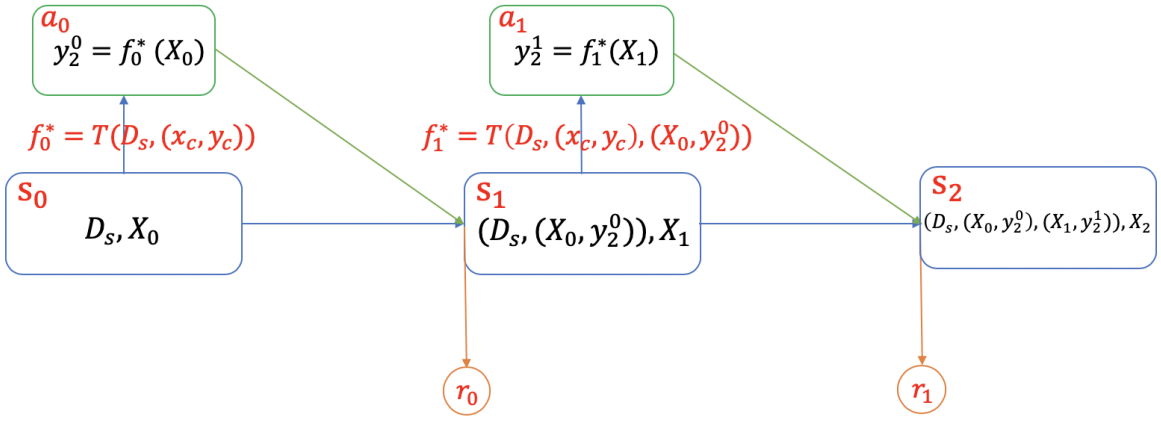


Figure 4.4: Iteration process for action function (4.8).

pability of the method and stabilizes the iteration process. Let $u_1^n = \{D_s, d^n\}$, where $d^n = (d_t^{n-1}, y_{pl}^{n-1})$, d_t^{n-1} is the set of target samples added in former steps, and y_{pl}^{n-1} is the set of their pseudo labels given by former steps. The modified action function consists of two components:

$$y_2^n = f_n^*(u_2^n), \tag{4.9}$$

and

$$y_{pl}^n = f_n^*(d_t^{n-1}), \tag{4.10}$$

where $f_n^* = T(u_1^n \bigcup H^t)$ and T is a stable basic learner. In other words, in each epoch, we first pseudo label the newly added target samples, and then renew the pseudo labels of target samples added before. The first step (4.9), following the ISDA algorithm, explores the labels for the newly added points. The second step (4.10) updates the former pseudo labels in order to correct any possible errors. Note that (4.10) will not dramatically change the pseudo labels of former points because the difference between y_{pl}^n and y_{pl}^{n-1} should be small, which promises a stable iteration process. We call this new method *Renewing Iterative Self-labeling Domain Adaptation* (Re-ISDA).

The iteration process of Re-ISDA is illustrated in Figure 4.5. For example, at the state s_1 there

will be two actions: a_1 and a_1': a_1 gives the pseudo label of x_1 based on the predictor f_1^* trained on $\{D_s \bigcup H^* \bigcup (X_0, y_2^0)\}$; a_1' renews the label y_2^0 (y_{pl}^0) by y_2^{0*} (y_{pl}^1) based on f_1^*.

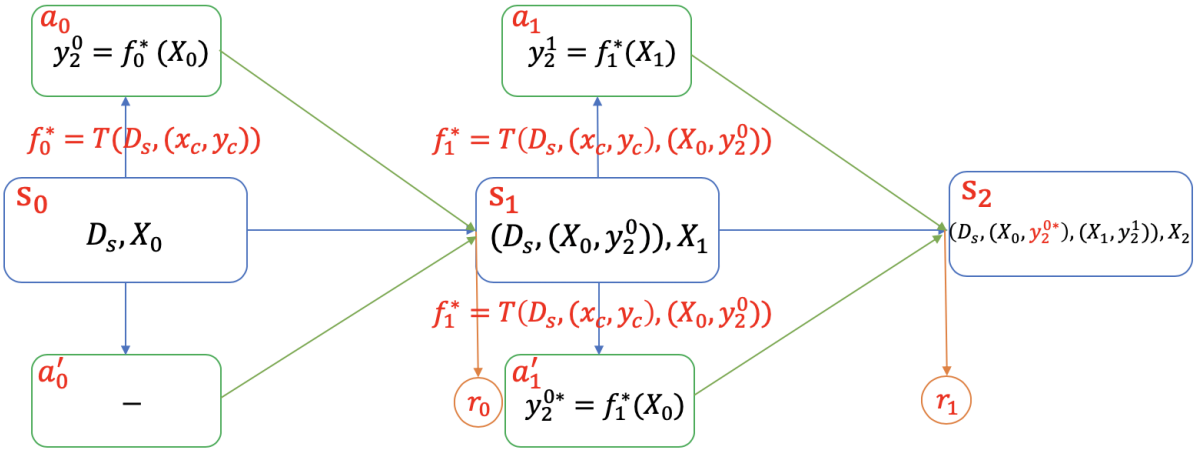


Figure 4.5: Iteration process for action functions (4.9) and (4.10).

The detailed algorithm of Re-ISDA is shown in Algorithm 2. We need to choose a basic learner Q, the calibration point (x_c, y_c), the order of adding target samples and the size of added samples η at each step when implementing the Re-ISDA. Some suggestions for implementation details of the proposed algorithm will be provided in Section 4.4.2.

Algorithm 2 Training steps for Re-ISDA

Input: ordered training set $S = \{(x_{s_1}, y_{s_1}), (x_{s_2}, y_{s_2}), ..., (x_{s_n}, y_{s_n})\}$, a labeled calibration point (x_c, y_c), testing samples $X_t = (x_{t_1}, x_{t_2}, ..., x_{t_m})$, the number of newly added testing samples η, number of epochs P, initial iterative set $S_0 = S \bigcup (x_c, y_c)$, a basic learner Q.

Output: Final predictor $\hat{f}_P(\cdot)$

1: **for** p in $1 : P$ **do**
2: Train a predictor $\hat{f}_p(\cdot)$ based on S_{p-1} and Q
3: Pseudo-predict set $G_p = \{(x_{t_1}, \hat{f}_p(x_{t_1})), (x_{t_2}, \hat{f}_p(x_{t_2})), ..., (x_{t_{rp}}, \hat{f}_p(x_{t_{rp}}))\}$
4: Renew iterative set $S_p = S_0 \cup G_p$
5: **end for**

4.4.2 Implementation details for Re-ISDA

In this section, we offer some practical guidelines for implementing the proposed Re-ISDA method.

4.4.2.1 Basic learner Q

We recommend using feed-forward neural networks (FNNs) as a basic learner. According to [107], FNNs are robust learners. Besides, FNNs can handle huge data sets and do not require complex transformation or representation learning for the raw data before training [5]. In order to get a well-performing FNN, one needs to choose suitable hyper-parameters such as the number of layers and nodes, learning rate, the number of training epochs, and so on. Detailed tuning guidelines for FNNs can be found in [108; 6; 109].

4.4.2.2 Calibration points (x_c, y_c)

In domain adaptation problems we may face two different situations: 1) no labeled target samples can be used; 2) a few labeled target samples are available. Here we give some guidelines for choosing the calibration points under different occasions:

- If we only have the labels for the source domain, we recommend choosing the source sample as the calibration point which has the least average distance between the target samples;
- If we only have one labeled target sample, we can choose this one as the calibration point;
- If we have several labeled target samples, we can choose one point randomly from them as the calibration point or implement Re-ISDA for each labeled target sample and ensemble their results to give the final prediction results.

4.4.2.3 Order of added target samples

We make the suggestions below for the order of added target samples under two types of data:

- If the data is in a time-series format, we should maintain the original order to keep the dynamic properties.

- If the data does not have a temporal structure, we recommend sorting the target samples from small to large according to their distance to the calibration point.

4.4.2.4 Size of added samples in each step η

The numerical experiments in Section 4.6 show that the Re-ISDA with smaller η will lead to more precise prediction results and a more stable iteration process. On the other hand, a smaller η will require more iteration steps and thus increase the computational cost. Therefore, we recommend setting η as small as possible provided that the computational cost is affordable.

4.5 Simulation studies

In this experiment we assess the performance of proposed Re-ISDA method and compare the results with TCA, KMM, ISDA and the situation without domain adaptation. We use the Friedman Function [31; 110] as the test function, which is defined as:

$$f(\mathbf{z}) = 10\sin(\pi z_1 z_2) + 20(z_3 - 0.5)^2 + 10z_4 + 5z_5. \tag{4.11}$$

Suppose the source domain is $[0.2, 1.2]^5$. A Halton sequence [90] with 80 samples in $[0.2, 1.2]^5$ and the corresponding labels are used as the training set, denoted as $X_s = \{(x_1, y_1), (x_2, y_2), ..., (x_{80}, y_{80})\}$. From the target domain we choose 41 samples given by adding 0.2 to each dimension of the inputs of the first 41 samples in X_s, i.e., $X_t = \{x_1 + \mathbf{0.2}, x_2 + \mathbf{0.2}, ..., x_{41} + \mathbf{0.2}\}$, where $\mathbf{0.2} = (0.2, 0.2, 0.2, 0.2, 0.2)^T$. Additionally, the calibration point is $(x_c, y_c) = (x_1 + \mathbf{0.2}, f(x_1 + \mathbf{0.2}))$, where f is the function in (4.11). Further, we reorder the testing samples from the smallest to the largest by their distance from (x_c, y_c). The performance of each predictor is evaluated by the root mean squared error (RMSE) defined as:

$$RMSE = \sqrt{\frac{1}{n}\sum_{n}^{i=1}(\hat{y}_i - y_i)^2}, \tag{4.12}$$

where n is the size of testing set, $\hat{y}_i$ is the predicted value of i-th testing sample and y_i is its true value. The implementation details are given below:

- The NN without domain adaptation uses a neural network with structure $c(5, 10, 5, 1)$, i.e., a fully connected neural network with four layers, in which the first layer has five dimensions; the second layer has ten dimensions; and so on. Its learning rate is 0.1, and the number of training epochs is 300. This network is also used as the base learner in the following algorithms;

- The KMM method uses the Gaussian kernel function with parameter 0.5 to re-weight the source samples, and then the NN used above is implemented to learn from the re-weighting set;

- The TCA method transforms the source input and target input to a new source input set and a new target input set, respectively, to minimize the distance between their distributions0 in a reproducing kernel Hilbert space. The new source and target input set have five dimensions. After the transformation, the new source set is used to train the base learner mentioned before and then to predict the values of the new target set;

- The ISDA pseudo labels two samples in each step according to the new order of target domain, and the predictor learner is the neural network with the same structure as mentioned before;

- The Re-ISDA is implemented similarly to ISDA except that it renews labels of all added target samples at the current state.

The RMSEs of the four methods are given in Table 4.1. Figure 4.6 gives the box plot of the absolute error of five methods. Table 4.1 and Figure 4.6 show that the proposed Re-ISDA has the lowest overall prediction error.

The pointwise predictive value of five methods for every point is shown in Figure 4.7 where the black points represent the true values. We can make the following observations from the Figure 4.7:

Table 4.1: RMSEs for the five methods on Friedman Function.

	NN without DA	TCA	ISDA	Re-ISDA	KMM
RMSE	3.623	8.901	4.070	2.033	2.70

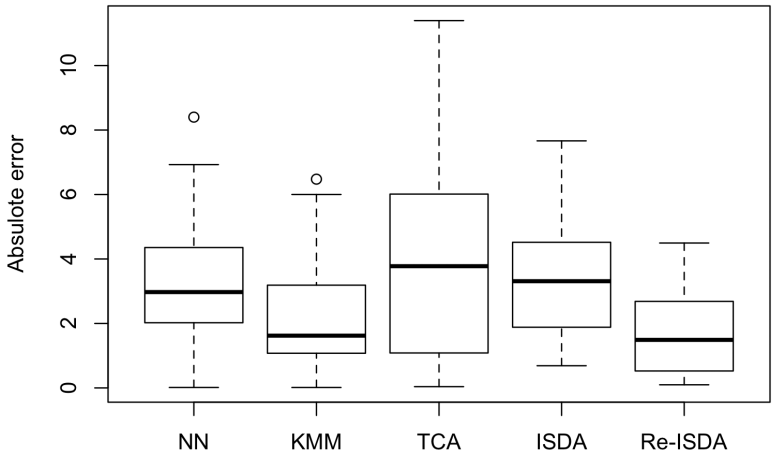


Figure 4.6: Absolute error of the five methods for Friedman Function.

- The brown line (NN) is almost all below the black one (true value) and can not follow the changing trend of the true value;
- The difference between the purple line (KMM) and the black line is significant, especially for the former half. The possible reason might be the loss of information caused by re-weighting;
- The blue line (TCA) vibrates heavily and has large errors for some points. As a result, the performance of TCA is not desirable;
- The performance of green line (ISDA) is much better than NN and TCA. However, its performance for the latter half points is much worse than the former half ones. The reason might be the mis-labeling issue [73; 74];
- The red line (Re-ISDA) is distributed much evenly on both sides of the black line compared

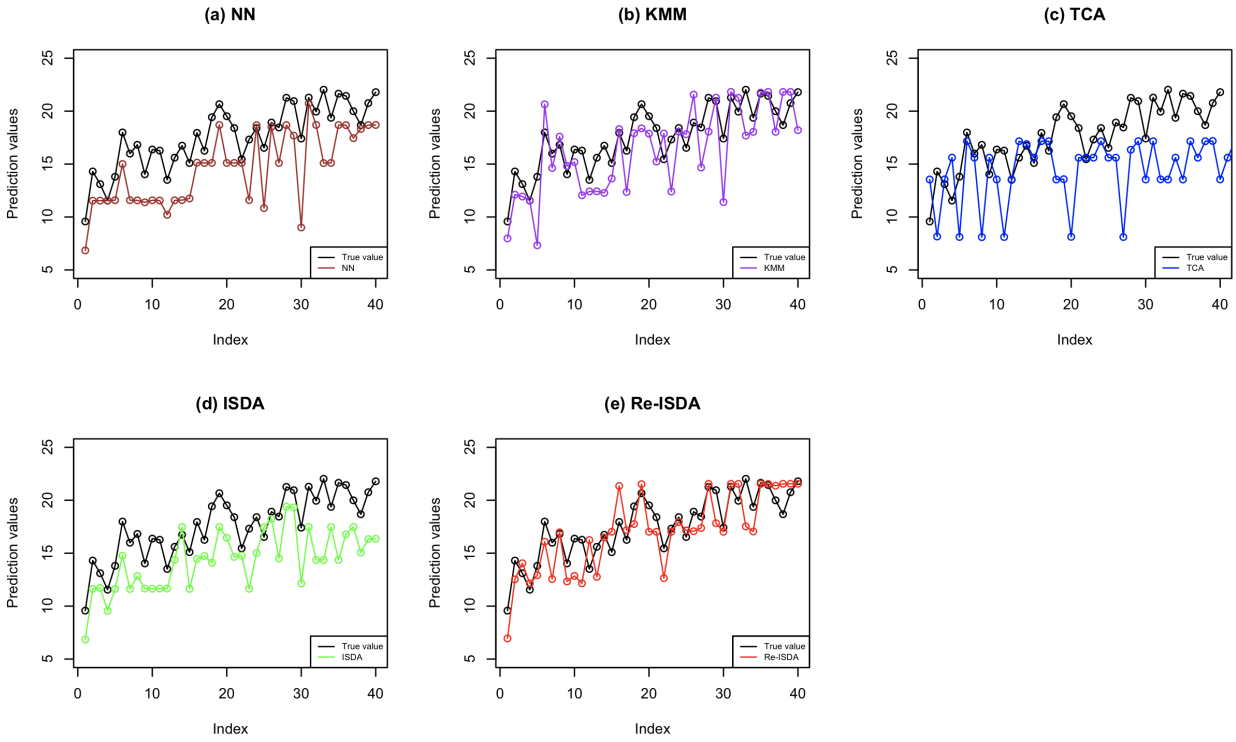


Figure 4.7: Prediction values of the five methods for Friedman Function.

with the above methods. Additionally, it follows the changing trend of true value much better and has better performance for the latter half points than ISDA. As a result, the Re-ISDA gains the lowest RMSE among the four methods.

4.6 Cervical spine motion prediction

In this section, we apply the proposed Re-ISDA method to the cervical spine motion prediction problem. The numerical results show that the Re-ISDA is more suitable for the cervical spine motion prediction problem compared with some prevailing transfer learning methods.

4.6.1 Background

Cervical spine, a highly complex multi-joint structure, supporting the head weight and providing the mobility and flexibility of the head [111], is susceptible to injuries that occur acutely and chronically. To design effective injury prevention and control, an accurate characterization of the cervical spine motion is needed for a better understanding of the neck biomechanics as well as the

pathomechanics of neck injury and neck pain [111; 112; 113].

Conventional approaches to measuring *in vivo* three-dimensional (3D) skeletal motion and position often involve using optical markers systems and inertial sensors placed on the body surface. Such surface-based measurements are subject to soft tissue motion artifacts and therefore do not reflect the underlying skeletal motion accurately [114; 115; 116; 117]. In addition, marker-based or sensor-based approaches are more suited for measuring the kinematics of long bones (i.e., extremities). For structures such as the cervical spine where relatively small individual bones are inter-connected by complex joints, surface-based measurements are simply too error-prone. Dynamic stereo-radiography (DSX) has proven to be an accurate, effective, and non-invasive method of quantifying 3D *in vivo* cervical spine motion and is regarded as the gold standard 4. However, the technology is not widely accessible and incurs tremendous cost both in development and use.

Previous studies have focused on using surface markers to estimate 3D bone motion or accessing the correlations between measurements from surface markers and from rigid markers (i.e., the markers are mounted on the bones) but are all limited to the lower extremity [118; 119; 120]. In a recent study, Nichoson et. al developed machine learning algorithms to predict scapular kinematics using optical motion measurement data [121], which further inspires us to use statistical or machine learning approaches to estimate and predict 3D cervical spine motion using optical motion measurements. This and our recent success in measuring cervical spine motion using DSX and optical motion capture inspire us to pursue new machine learning approaches to estimate cervical vertebrae motion.

Specifically, There are two fundamentally different approaches to the measurement of cervical spine motion: 1) surface-based approach, which is most widely used and more practical but is subject to significant error sources such as soft tissue motion artifact in estimating the underlying skeletal (i.e., rigid-body) movement; 2) dynamic radiography, which provides the "gold standard" measurement of skeletal movement but is costly, not well accessible and predisposes human subjects to radiation exposure. In view of the practical challenges in acquiring the dynamic radiography data, we hope to build a subject-specific machine learning model that can predict subjects'

skeletal motion (output) based only on their surface-based measurement (input).

4.6.2 Domain adaptation in the spine motion prediction problem

In the spine motion prediction problem, the joint distribution of the surface-based measurement (input features) for each subject may vary with one's distinct BMI, sex and other characteristics (i.e., $P_s(X) \neq P_t(X)$). At the same time, we assume the relationship between the surface-based measurement and the skeletal movement for all subjects, characterized by the conditional distributions, are the same (i.e., $P(Y_s|X_s) = P(Y_t|X_t)$). Based on the above two assumptions, we can formulate the spine motion prediction problem as a domain adaptation problem.

4.6.3 Experimental data acquisition

In this experiment, participants performed self-paced head-neck flexion-extension tasks (i.e., dynamic free range-of-motion tasks) with the head-neck moving primarily in the sagittal plane while their cervical spine region was imaged continuously using a DSX system [122]. A 12-camera Vicon motion capture system (Vantage-Series, Vicon Motion Labs, Oxford, UK) was used to monitor and record the head-neck motions from retro-reflective spherical surface markers placed on ten anatomical landmarks: glabellae, inferior border of each orbit (left and right), the tragion notches (left and right), acromion processes (left and right), suprasternal notch, sternum, and C7 spinous process as shown in Figure. 4.8.

The recorded trajectories of the surface markers were tracked in a global laboratory coordinate system, resulting in time-histories of three coordinates (x, y, and z) in each marker.

Three-dimensional (3D) orientation and position of each vertebra were determined via a previously validated volumetric model-based tracking algorithm [123; 124; 125; 126]. Cervical sagittal spinal curvature was measured at each time frame using the Cobb method [127; 128; 129] where the Cobb angle was defined by the angle between the two inferior endplates of the vertebrae. However, due to the lack of the endplate in C1 (the topmost vertebra), the cervical sagittal spinal curvature in this study was calculated by the Cobb angle between C2 and C7 inferior endplates as shown in Figure 4.9. The raw data consists of the following parts:

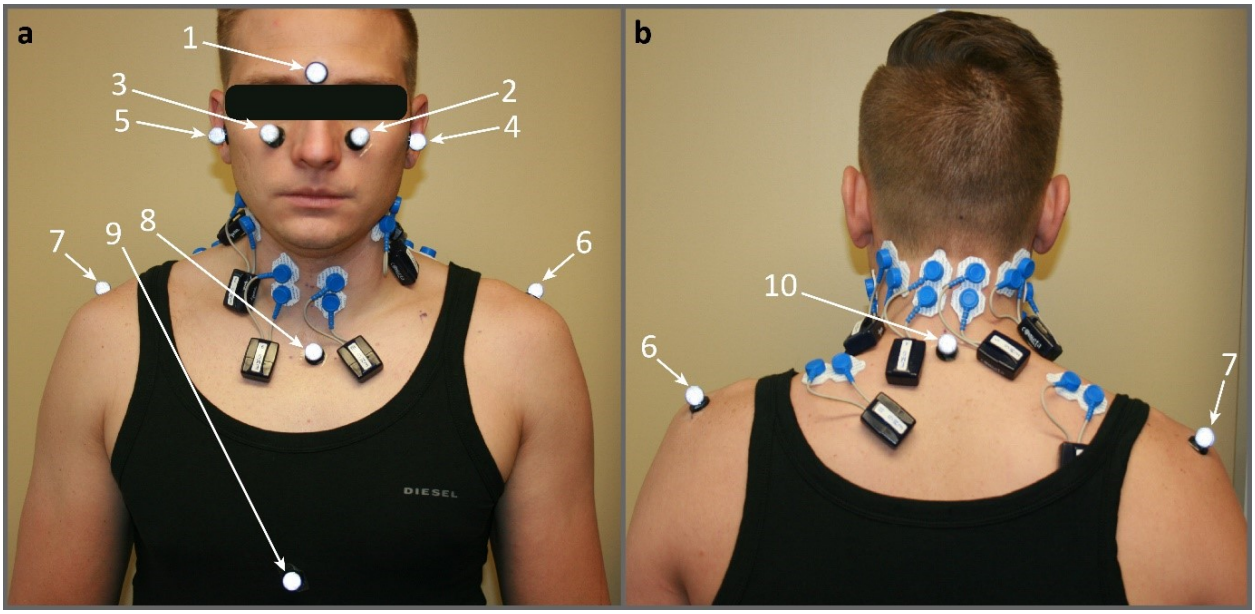


Figure 4.8: Retro-reflective spherical surface marker placements: a the front view; b the back view. 1 glabella (marker Forehead), 2&3 inferior border of each orbit on both sides (marker LORBIT and marker RORBIT), 4&5 the tragion notches of both sides (marker LMP and marker RMP), 6&7 acromion processes (marker LSHO and marker RSHO), 8 suprasternal notch (marker CLAV), 9 sternum (marker STRN), and 10 C7 spinous process (marker C7).

- The coordinates information given by 9 markers which named as C7, CLAV, LMP, LORBIT, LSHO, RMP, RORBIT, RSHO, STRN (9×3 in total);
- BMI information and C7-MMP statistic which is the distance between marker C7 and the midpoint of marker RMP and marker LMP calculated at the neutral static posture;
- Neck angle, measured as the angle between the horizontal plane and the plane defined by the marker C7, LMP, and RMP;
- Head angle, measured as the angle between Franfort plane and the horizontal plane, where the Frankfort was defined by the surface markers LMP, RMP, LORBIT, and RORBIT;
- C7-MMP, the distance between the marker C7 and the midpoint of marker RMP and marker LMP at each frame during the dynamic head-neck flexion-extention motion trial;
- Cobb angle as measured by dynamic biplane radiography.

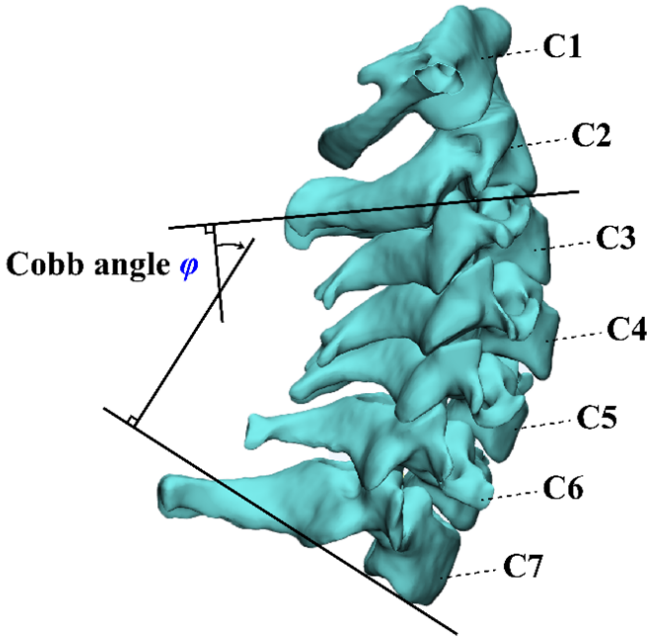


Figure 4.9: Cervical spinal curvature determined by Cobb angle.

4.6.4 Pre-processing

We shall use the marker coordinates information, BMI information, C7-MMP statistic, neck angle, head angle, and C7-MMP as the input data to predict the corresponding Cobb angle. In contrast to the BMI information and C7-MMP, which do not evolve in time, the rest of the input data are time series. They capture the position of the corresponding markers and are collected sequentially with fixed time intervals.

As a pre-processing step, we take the difference of every two consecutive frames of the time-series data. We believe that this difference can be used to enhance the prediction performance because it reflects the velocity of each marker. We include this computed "velocity information" of the time-series input variables as part of the input data for each candidate learning method. In summary, the input dimension is 62.

4.6.5 Objective and problem formulation

We have labeled information (both marker information and its corresponding Cobb angle information) of six people: subject #3, 4, 17, 31, 34, 38. The goal is to identify the suitable method to predict the Cobb angle information based on the corresponding marker information, which is required and necessary for a better understanding of the neck biomechanics. To this end, we partition the dataset into two parts:

- Source domain (training set): subject #3, 4, 31, 34, 38 (called $D_{s1}, D_{s2}, D_{s3}, D_{s4}, D_{s5}$);
- Target domain (testing set): subject #17 (called D_t).

Additionally, we assume that a labeled target sample $\{x^*, y^*\}$ is available (working as the calibration point), which is the first sample of subject #17 with its true label. This assumption is reasonable in real-world applications because we only need one frame of the biplane radiography data, which can be collected via a conventional static radiography equipment in an inexpensive experiment.

In this work, we consider four candidates of domain adaption methods: the proposed Re-ISDA and other three prevailing transfer learning methods. The performance of each method is measured by the RMSE under the testing data.

4.6.6 Data analysis

First, we apply the max-min normalization on both the source and target domain data set to stabilize the forthcoming training process. Next, we employ the principal component analysis (PCA) to reduce the dimensions. Figure 4.10 shows the variances of the first ten components given by PCA. The eleventh principal component accounts for only about 1% of the total variance, which is almost negligible. In this experiment, we choose the first ten components as the new design. Figure 4.11 shows the distribution of the first two components of #3, 4, 31, 34, 38, 17 against their true Cobb angle information, which can approximately reflect their distribution in the 63 dimensions space. From Figure 4.11, the data from each subject can be fitted by a separate curve.

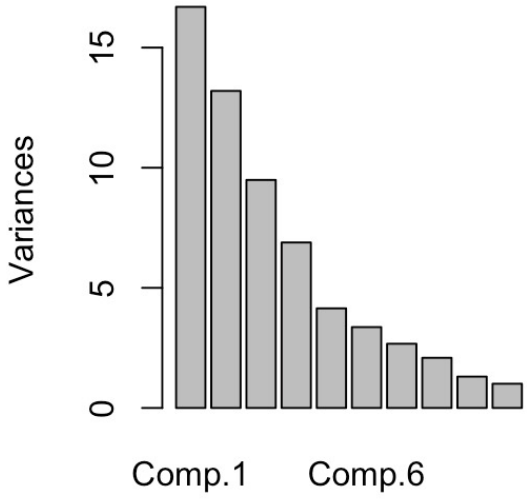


Figure 4.10: The variance of components in PCA.

Thus, we consider the task of prediction as a multi-source transform learning problem. Denote the function trained by each source as $f_i = T(D_{si}), i = 1, 2, 3, 4, 5$, and the prediction function for target domain D_t as f_t. For convenience, we choose f_t according to the following rule:

$$f_t = \arg\min_{f_i} |f_i(x^*) - y^*|, \tag{4.13}$$

In this experiment, we choose f_5 as f_t and the initial source domain $D_s = D_{s5}$.

Then we implement the four methods as follow:

- the NN without domain adaptation uses a neural network with structure $c(10, 16, 16, 8, 1)$. Its learning rate is 0.1, and the number of training epochs is 1500. This network is also used as the base learner in the following algorithms;
- the KMM method uses Gaussian kernel function with parameter 2 to re-weight the source samples, and the NN used above is implemented to learn from the re-weighting set;
- the TCA method skips the PCA, and is implemented directly on the original input with 63 dimensions. The new source and target input set have ten dimensions. After the transforma-

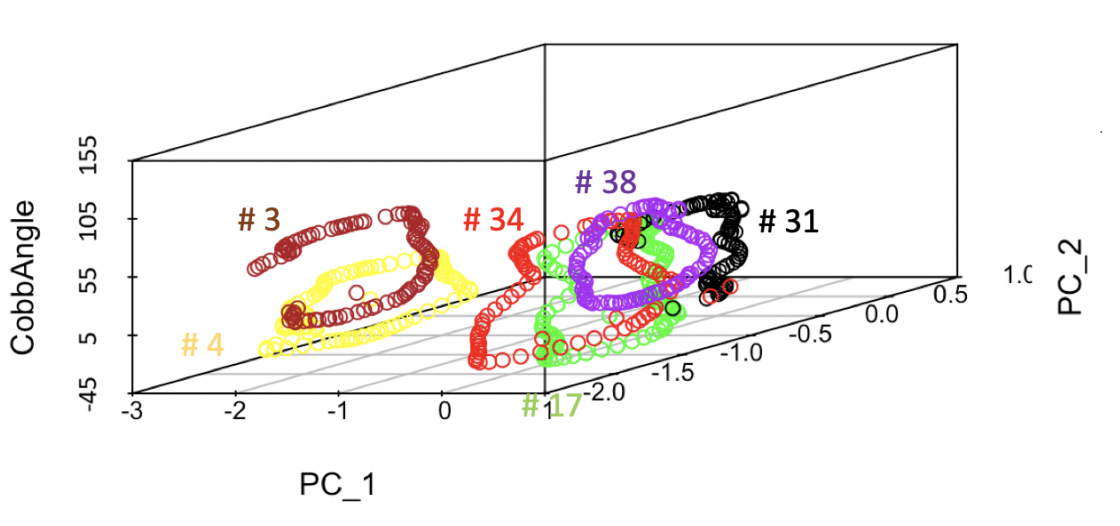


Figure 4.11: The distributions of first two components of #3, 4, 31, 34, 38, 17 against their Cobb angle (in degrees).

tion, the new source set is used to train the base learner mentioned before and then to predict the values of the new target set;

- the ISDA pseudo labels two samples in each step according to the new order of target domain, and the predictor learner is the neural network with the same structure as mentioned before;
- the Re-ISDA is implemented similarly to ISDA except that it renews labels of all added target samples at the current state.

The RMSEs of the four methods are given in Table 4.2 and Figure 4.12 shows the absolute error of the five methods in the box plot. These two figures prove that the Re-ISDA has the best performance compared with the other four methods.

Table 4.2: RMSEs for the five methods in the spine motion prediction problem

	NN without DA	TCA	ISDA	Re-ISDA	KMM
RMSE	8.19	28.88	6.21	2.68	13.2

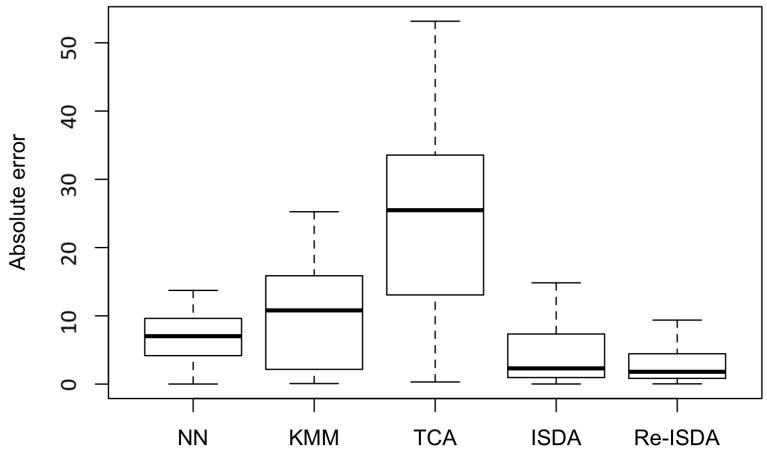


Figure 4.12: Absolute error of the five methods in the spine motion prediction problem.

The pointwise predictive value for every point is shown in Fig.4.13 where the black points represent the true values. Clearly, we can see from the Fig.4.13:

- The brown line (NN) deviates far away from the black one (true value), especially for the latter half, representing that the traditional predictor can not work well for this situation;
- The performance of the purple line (KMM) is worse than NN. In this experiment, the difference between weights of different source samples is up to 10^{10}. As a result, many points with tiny weights are actually deleted from the training set, which causes the under-fitting of the model;
- The blue line (TCA) is approximately a straight line. We have tried different kernel functions and their parameters for the TCA but get some straight lines just like the line shown here. The possible reason might be that the TCA makes the source domain harder to learn for a regression mission;
- The green line (ISDA) works much better than the above three methods and is very close to the black line. However, its performance for the former half part is not desirable. The

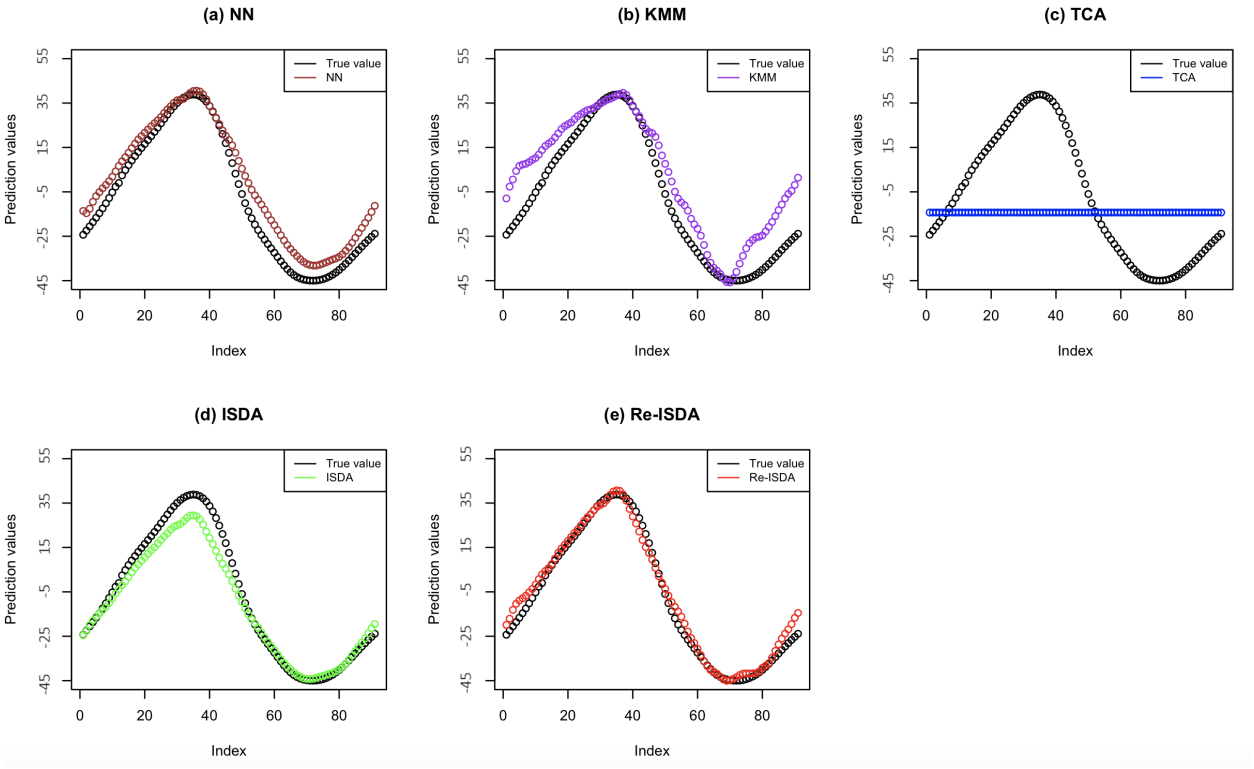


Figure 4.13: Prediction values of the five methods in the spine motion prediction problem.

possible reason for it might be the little flexibility of ISDA so that it can not modify the labels of the former samples based on the points coming afterward;

- The red line (Re-ISDA) is approximately the same as the black one, which indicates that our method works highly well. This result shows that the proposed Re-ISDA method can transfer the predictor through different marginal distributions and is flexible enough to modify the pseudo labels in the former steps.

Additionally, we test the performance of Re-ISDA for different η. Figure 4.14 shows the RMSEs for the labeled points (both newly labeled and labeled in previous steps) at each step of Re-ISDA when $\eta = 2, 3, 5$. In the iteration processes, the RMSEs of predictors increase at the beginning and then fall, proving that the Re-ISDA modifies the prediction error of the added points through the new coming samples. Besides, if we choose the smaller η, the performance of the predictor changes more smoothly and ends at a lower RMSE. However, a smaller η means

more training epochs and higher computation cost, so that one needs to make a trade-off between precision and cost.

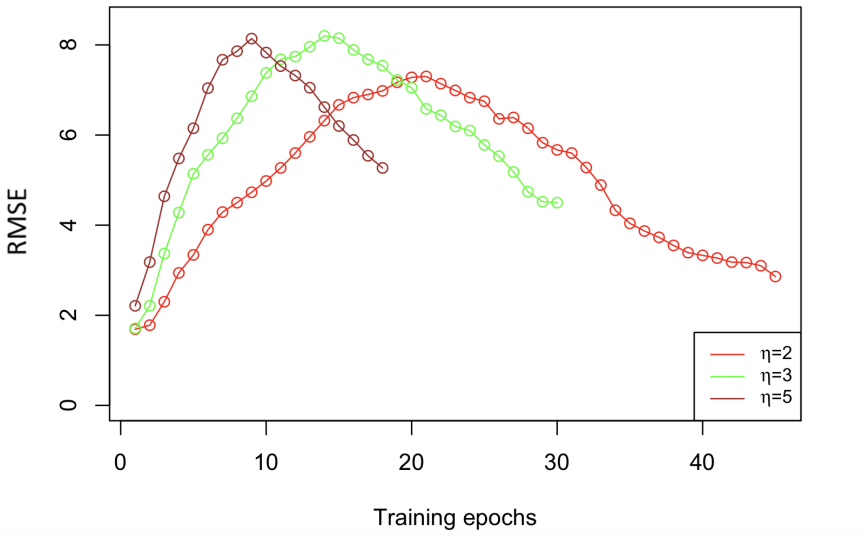


Figure 4.14: Performance of Re-ISDA for different η during their training processes.

5. SUMMARY AND DISCUSSIONS

In this thesis, we propose novel data-efficient methodologies for three different challenges regarding modeling or experimental designs in the experimentation field.

In the first work, we propose a projection pursuit approach based on Gaussian process regression to fit deterministic computer outputs. The proposed method has a better model prediction and generalization power when the input dimension is high, and the sample size is small.

Despite its advantages, the proposed method has a few issues to be addressed in future investigations. First, PPGPR involves quite a few hyper-parameters. Although we have provided a few guidelines regarding the choice of these hyper-parameters, how to better choose or tune these parameters requires further investigation. Second the current algorithm can only handle moderate data sets due to its high computational cost. We believe that this issue can be mitigated by implementing the following techniques: 1) parallel or GPU computation, 2) the recent advances in scalable GP inference and prediction [130].

In the second work, we propose a novel design methods for multi-fidelity computer experiments. We first assume a sequence of independent Gaussian process with shrinking variance for the autoregressive model. By incorporating the error bound for Gaussian process regression given by [57], we come up with the optimal designs for a fixed accuracy. Additionally, we prove that our well-designed multi-fidelity experiments will require much lower cost compared with the single-fidelity experiments in the order pf magnitude to reach the same accuracy level. The numerical studies show the superiority of our MLGP method over the single-fidelity method and the LHD method.

Since the error bound in Theorem 3.4.3 is proved based on Matern kernel, all the follow-up results include the smoothness parameter ν. As a result, the proposed MLGP method may have trouble when Gaussian kernels (with $\nu = \infty$) are used. However, the examples in 3.5.1 show that the MLGP method can work well for large ν's. Given this, we recommend substituting a large enough ν to generate designs when implementing the MLGP method for Guassian kernels.

In the last work, we propose a novel method called Re-ISDA for the domain adaptation problem with limited labeled samples based on the ISDA algorithm. We first formulate the learning problem as a dynamic programming model and then propose a greedy algorithm to solve it efficiently. The proposed method updates all labels of the target samples (both newly added and added in former iterations) at each iteration, leading to higher flexibility and fewer mis-labeling issues than ISDA. Our numerical experiments and a cervical spine motion prediction example confirm the prediction performance of the proposed method.

The computational cost of the proposed method is slightly higher than the other domain adaptation methods we have compared with. Thus the proposed method should be more suitable for moderate-sized data. Besides, the Re-ISDA is more suitable for the time-series data because it can incorporate the time-series information. In this work, we only consider prediction problems, but the proposed method, with proper modifications, should also be applicable to classification problems.

REFERENCES

[1] C. J. Wu and M. S. Hamada, *Experiments: planning, analysis, and optimization*. John Wiley & Sons, 2011.

[2] T. Santner, B. Williams, and W. Notz, *The Design and Analysis of Computer Experiments*. Springer Verlag, 2003.

[3] R. B. Gramacy, *Surrogates: Gaussian process modeling, design, and optimization for the applied sciences*. Chapman and Hall/CRC, 2020.

[4] C. K. Williams and C. E. Rasmussen, *Gaussian processes for machine learning*, vol. 2. MIT press Cambridge, MA, 2006.

[5] Y. LeCun, Y. Bengio, and G. Hinton, "Deep learning," *Nature*, vol. 521, no. 7553, pp. 436–444, 2015.

[6] I. Goodfellow, Y. Bengio, and A. Courville, *Deep Learning*. MIT press, 2016.

[7] M. C. Kennedy and A. O'Hagan, "Predicting the output from a complex computer code when fast approximations are available," *Biometrika*, vol. 87, no. 1, pp. 1–13, 2000.

[8] S. Mak, C.-L. Sung, X. Wang, S.-T. Yeh, Y.-H. Chang, V. R. Joseph, V. Yang, and C. J. Wu, "An efficient surrogate model for emulation and physics extraction of large eddy simulations," *Journal of the American Statistical Association*, vol. 113, no. 524, pp. 1443–1456, 2018.

[9] G. Rennen, B. Husslage, E. R. Van Dam, and D. Den Hertog, "Nested maximin latin hypercube designs," *Structural and Multidisciplinary Optimization*, vol. 41, no. 3, pp. 371–395, 2010.

[10] C. Rasmussen and C. Williams, *Gaussian Processes for Machine Learning*. The MIT Press, 2006.

[11] S. Ba, V. R. Joseph, *et al.*, "Composite Gaussian process models for emulating expensive functions," *The Annals of Applied Statistics*, vol. 6, no. 4, pp. 1838–1860, 2012.

[12] R. B. Gramacy and H. K. H. Lee, "Bayesian treed Gaussian process models with an ap-

plication to computer modeling," *Journal of the American Statistical Association*, vol. 103, no. 483, pp. 1119–1130, 2008.

[13] M. J. Heaton, W. F. Christensen, and M. A. Terres, "Nonstationary Gaussian process models using spatial hierarchical clustering from finite differences," *Technometrics*, vol. 59, no. 1, pp. 93–101, 2017.

[14] L.-H. Lin and V. Roshan Joseph, "Transformation and additivity in Gaussian processes," *Technometrics*, pp. 1–11, 2019.

[15] N. Durrande, D. Ginsbourger, O. Roustant, and L. Carraro, "Anova kernels and rkhs of zero mean functions for model-based sensitivity analysis," *Journal of Multivariate Analysis*, vol. 115, pp. 57–67, 2013.

[16] J. E. Oakley and A. O'Hagan, "Probabilistic sensitivity analysis of complex models: a Bayesian approach," *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, vol. 66, no. 3, pp. 751–769, 2004.

[17] A. Saltelli, P. Annoni, I. Azzini, F. Campolongo, M. Ratto, and S. Tarantola, "Variance based sensitivity analysis of model output. design and estimator for the total sensitivity index," *Computer Physics Communications*, vol. 181, no. 2, pp. 259–270, 2010.

[18] J. M. Hokanson and P. G. Constantine, "Data-driven polynomial ridge approximation using variable projection," *SIAM Journal on Scientific Computing*, vol. 40, no. 3, pp. A1566–A1589, 2018.

[19] A. Glaws, P. G. Constantine, and R. D. Cook, "Inverse regression for ridge recovery: a data-driven approach for parameter reduction in computer experiments," *Statistics and Computing*, vol. 30, no. 2, pp. 237–253, 2020.

[20] A. Pinkus, "Approximating by ridge functions," *Surface fitting and multiresolution methods*, pp. 279–292, 1997.

[21] C. Linkletter, D. Bingham, N. Hengartner, D. Higdon, and K. Q. Ye, "Variable selection for Gaussian process models in computer experiments," *Technometrics*, vol. 48, no. 4, pp. 478–490, 2006.

[22] M. Gu, "Jointly robust prior for gaussian stochastic process in emulation, calibration and variable selection," *Bayesian Analysis*, vol. 14, no. 3, pp. 857–885, 2019.

[23] P. G. Constantine, E. Dow, and Q. Wang, "Active subspace methods in theory and practice: applications to kriging surfaces," *SIAM Journal on Scientific Computing*, vol. 36, no. 4, pp. A1500–A1524, 2014.

[24] M. Stein, *Interpolation of Spatial Data: Some Theory for Kriging*. Springer Verlag, 1999.

[25] X. Deng, C. D. Lin, K.-W. Liu, and R. Rowe, "Additive Gaussian process for computer models with qualitative and quantitative factors," *Technometrics*, vol. 59, no. 3, pp. 283–292, 2017.

[26] D. K. Duvenaud, H. Nickisch, and C. E. Rasmussen, "Additive Gaussian processes," in *Advances in Neural Information Processing Systems*, pp. 226–234, 2011.

[27] É. Lebarbier, "Detecting multiple change-points in the mean of Gaussian process by model selection," *Signal Processing*, vol. 85, no. 4, pp. 717–736, 2005.

[28] I. Delbridge, D. Bindel, and A. G. Wilson, "Randomly projected additive gaussian processes for regression," in *International Conference on Machine Learning*, pp. 2453–2463, PMLR, 2020.

[29] N. Durrande, D. Ginsbourger, and O. Roustant, "Additive covariance kernels for high-dimensional gaussian process modeling," in *Annales de la Faculté des sciences de Toulouse: Mathématiques*, vol. 21, pp. 481–499, 2012.

[30] R. Tripathy, I. Bilionis, and M. Gonzalez, "Gaussian processes with built-in dimensionality reduction: Applications to high-dimensional uncertainty propagation," *Journal of Computational Physics*, vol. 321, pp. 191–223, 2016.

[31] J. H. Friedman and W. Stuetzle, "Projection pursuit regression," *Journal of the American statistical Association*, vol. 76, no. 376, pp. 817–823, 1981.

[32] F. Ferraty, A. Goia, E. Salinelli, and P. Vieu, "Functional projection pursuit regression," *Test*, vol. 22, no. 2, pp. 293–320, 2013.

[33] E. Gilboa, Y. Saatçi, and J. Cunningham, "Scaling multidimensional Gaussian processes us-

ing projected additive approximations," in *International Conference on Machine Learning*, pp. 454–461, 2013.

[34] C.-L. Li, K. Kandasamy, B. Póczos, and J. Schneider, "High dimensional Bayesian optimization via restricted projection pursuit models," in *Artificial Intelligence and Statistics*, pp. 884–892, 2016.

[35] P. Z. Qian and C. J. Wu, "Bayesian hierarchical modeling for integrating low-accuracy and high-accuracy experiments," *Technometrics*, vol. 50, no. 2, pp. 192–204, 2008.

[36] S. Xiong, P. Z. Qian, and C. J. Wu, "Sequential design and analysis of high-accuracy and low-accuracy computer codes," *Technometrics*, vol. 55, no. 1, pp. 37–46, 2013.

[37] G. Pilania, J. E. Gubernatis, and T. Lookman, "Multi-fidelity machine learning models for accurate bandgap predictions of solids," *Computational Materials Science*, vol. 129, pp. 156–163, 2017.

[38] K. Cutajar, M. Pullin, A. Damianou, N. Lawrence, and J. González, "Deep gaussian processes for multi-fidelity modeling," *arXiv preprint arXiv:1903.07320*, 2019.

[39] R. Tuo, C. J. Wu, and D. Yu, "Surrogate modeling of computer experiments with different mesh densities," *Technometrics*, vol. 56, no. 3, pp. 372–380, 2014.

[40] K. Fang, R. Li, and A. Sudjianto, *Design and Modeling for Computer Experiments*, vol. 6. Chapman & Hall/CRC, 2005.

[41] P. Z. Qian, "Nested latin hypercube designs," *Biometrika*, vol. 96, no. 4, pp. 957–970, 2009.

[42] X. He and P. Z. Qian, "Nested orthogonal array-based latin hypercube designs," *Biometrika*, vol. 98, no. 3, pp. 721–731, 2011.

[43] P. Z. Qian, "Sliced latin hypercube designs," *Journal of the American Statistical Association*, vol. 107, no. 497, pp. 393–399, 2012.

[44] Y. Hwang, X. He, and P. Z. Qian, "Sliced orthogonal array-based latin hypercube designs," *Technometrics*, vol. 58, no. 1, pp. 50–61, 2016.

[45] Z. Qian, C. C. Seepersad, V. R. Joseph, J. K. Allen, and C. Jeff Wu, "Building surrogate models based on detailed and approximate simulations," 2006.

[46] A. Ehara and S. Guillas, "An adaptive strategy for sequential designs of multilevel computer experiments," *arXiv preprint arXiv:2104.02037*, 2021.

[47] X. He and P. Z. Qian, "A central limit theorem for nested or sliced latin hypercube designs," *Statistica Sinica*, pp. 1117–1128, 2016.

[48] M. B. Giles, "Multilevel monte carlo path simulation," *Operations research*, vol. 56, no. 3, pp. 607–617, 2008.

[49] M. B. Giles, "Multilevel monte carlo methods," *Acta numerica*, vol. 24, pp. 259–328, 2015.

[50] S. Heinrich, "Multilevel monte carlo methods," in *International Conference on Large-Scale Scientific Computing*, pp. 58–67, Springer, 2001.

[51] L. W. Ng and K. E. Willcox, "Multifidelity approaches for optimization under uncertainty," *International Journal for numerical methods in Engineering*, vol. 100, no. 10, pp. 746–772, 2014.

[52] A.-L. Haji-Ali, F. Nobile, and R. Tempone, "Multi-index monte carlo: when sparsity meets sampling," *Numerische Mathematik*, vol. 132, no. 4, pp. 767–806, 2016.

[53] B. Peherstorfer, T. Cui, Y. Marzouk, and K. Willcox, "Multifidelity importance sampling," *Computer Methods in Applied Mechanics and Engineering*, vol. 300, pp. 490–509, 2016.

[54] B. Peherstorfer, K. Willcox, and M. Gunzburger, "Optimal model management for multifidelity monte carlo estimation," *SIAM Journal on Scientific Computing*, vol. 38, no. 5, pp. A3163–A3194, 2016.

[55] G. Geraci, M. S. Eldred, and G. Iaccarino, "A multifidelity multilevel monte carlo method for uncertainty propagation in aerospace applications," in *19th AIAA non-deterministic approaches conference*, p. 1951, 2017.

[56] A. L. Teckentrup, P. Jantsch, C. G. Webster, and M. Gunzburger, "A multilevel stochastic collocation method for partial differential equations with random input data," *SIAM/ASA Journal on Uncertainty Quantification*, vol. 3, no. 1, pp. 1046–1074, 2015.

[57] R. Tuo and W. Wang, "Kriging prediction with isotropic Matérn correlations: Robustness and experimental designs.," *J. Mach. Learn. Res.*, vol. 21, pp. 187–1, 2020.

[58] S. J. Pan and Q. Yang, "A survey on transfer learning," *IEEE Transactions on Knowledge and Data Engineering*, vol. 22, no. 10, pp. 1345–1359, 2009.

[59] K. Weiss, T. M. Khoshgoftaar, and D. Wang, "A survey of transfer learning," *Journal of Big data*, vol. 3, no. 1, p. 9, 2016.

[60] J. Huang, A. Gretton, K. Borgwardt, B. Schölkopf, and A. J. Smola, "Correcting sample selection bias by unlabeled data," in *Advances in Neural Information Processing Systems*, pp. 601–608, 2007.

[61] W. Dai, Q. Yang, G.-R. Xue, and Y. Yu, "Boosting for transfer learning," in *Proceedings of the 24th International Conference on Machine Learning*, pp. 193–200, 2007.

[62] R. Chattopadhyay, Q. Sun, W. Fan, I. Davidson, S. Panchanathan, and J. Ye, "Multisource domain adaptation and its application to early detection of fatigue," *ACM Transactions on Knowledge Discovery from Data*, vol. 6, no. 4, pp. 1–26, 2012.

[63] H. Daumé III, "Frustratingly easy domain adaptation," *arXiv preprint arXiv:0907.1815*, 2009.

[64] L. Duan, D. Xu, and I. W.-H. Tsang, "Domain adaptation from multiple sources: A domain-dependent regularization approach," *IEEE Transactions on Neural Networks and Learning Systems*, vol. 23, no. 3, pp. 504–518, 2012.

[65] S. J. Pan, I. W. Tsang, J. T. Kwok, and Q. Yang, "Domain adaptation via transfer component analysis," *IEEE Transactions on Neural Networks*, vol. 22, no. 2, pp. 199–210, 2010.

[66] M. Long, J. Wang, G. Ding, J. Sun, and P. S. Yu, "Transfer feature learning with joint distribution adaptation," in *Proceedings of the IEEE International Conference on Computer Vision*, pp. 2200–2207, 2013.

[67] S. J. Pan, X. Ni, J.-T. Sun, Q. Yang, and Z. Chen, "Cross-domain sentiment classification via spectral feature alignment," in *Proceedings of the 19th International Conference on World Wide Web*, pp. 751–760, 2010.

[68] Y. Ganin, E. Ustinova, H. Ajakan, P. Germain, H. Larochelle, F. Laviolette, M. Marchand, and V. Lempitsky, "Domain-adversarial training of neural networks," *The journal of ma-*

chine learning research, vol. 17, no. 1, pp. 2096–2030, 2016.

[69] M. Long, Y. Cao, J. Wang, and M. I. Jordan, "Learning transferable features with deep adaptation networks," *arXiv preprint arXiv:1502.02791*, 2015.

[70] M. Long, H. Zhu, J. Wang, and M. I. Jordan, "Deep transfer learning with joint adaptation networks," in *International conference on machine learning*, pp. 2208–2217, PMLR, 2017.

[71] A. Habrard, J.-P. Peyrache, and M. Sebban, "Iterative self-labeling domain adaptation for linear structured image classification," *International Journal on Artificial Intelligence Tools*, vol. 22, no. 05, p. 1360005, 2013.

[72] O. Bousquet and A. Elisseeff, "Stability and generalization," *Journal of Machine Learning Research*, vol. 2, no. Mar, pp. 499–526, 2002.

[73] Q. Wang and T. Breckon, "Unsupervised domain adaptation via structured prediction based selective pseudo-labeling," in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 34, pp. 6243–6250, 2020.

[74] J. Zhang, W. Li, and P. Ogunbona, "Joint geometrical and statistical alignment for visual domain adaptation," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 1859–1867, 2017.

[75] G. E. Hinton and R. R. Salakhutdinov, "Reducing the dimensionality of data with neural networks," *Science*, vol. 313, no. 5786, pp. 504–507, 2006.

[76] D. C. Psichogios and L. H. Ungar, "A hybrid neural network-first principles approach to process modeling," *AIChE Journal*, vol. 38, no. 10, pp. 1499–1511, 1992.

[77] Y. Khoo, J. Lu, and L. Ying, "Solving parametric PDE problems with artificial neural networks," *arXiv preprint arXiv:1707.03351*, 2017.

[78] R. K. Tripathy and I. Bilionis, "Deep UQ: Learning deep neural network surrogate models for high dimensional uncertainty quantification," *Journal of Computational Physics*, vol. 375, pp. 565–588, 2018.

[79] A. G. Wilson, D. A. Knowles, and Z. Ghahramani, "Gaussian process regression networks," *arXiv preprint arXiv:1110.4411*, 2011.

[80] T. Choi, J. Q. Shi, and B. Wang, "A Gaussian process regression approach to a single-index model," *Journal of Nonparametric Statistics*, vol. 23, no. 1, pp. 21–36, 2011.

[81] R. B. Gramacy and H. Lian, "Gaussian process single-index models as emulators for computer experiments," *Technometrics*, vol. 54, no. 1, pp. 30–41, 2012.

[82] Y. Hu, R. B. Gramacy, and H. Lian, "Bayesian quantile regression for single-index models," *Statistics and Computing*, vol. 23, no. 4, pp. 437–454, 2013.

[83] J.-L. Wang, L. Xue, L. Zhu, Y. S. Chong, *et al.*, "Estimation for a partial-linear single-index model," *The Annals of statistics*, vol. 38, no. 1, pp. 246–274, 2010.

[84] J. Friedman, T. Hastie, and R. Tibshirani, *The Elements of Statistical Learning*, vol. 1. Springer series in statistics New York, 2001.

[85] H.-G. Müller and F. Yao, "Functional additive models," *Journal of the American Statistical Association*, vol. 103, no. 484, pp. 1534–1544, 2008.

[86] G. M. James and B. W. Silverman, "Functional adaptive model estimation," *Journal of the American Statistical Association*, vol. 100, no. 470, pp. 565–576, 2005.

[87] H. Wendland, *Scattered Data Approximation*. Cambridge University Press, 2005.

[88] L. Prechelt, "Early stopping–but when?," in *Neural Networks: Tricks of the Trade*, pp. 55–69, Springer, 1998.

[89] W. V. Harper and S. K. Gupta, *Sensitivity/Uncertainty Analysis of A Borehole Scenario Comparing Latin Hypercube Sampling and Deterministic Sensitivity Approaches*. Office of Nuclear Waste Isolation, Battelle Memorial Institute, 1983.

[90] J. H. Halton, "Algorithm 247: Radical-inverse quasi-random point sequence," *Communications of the ACM*, vol. 7, no. 12, pp. 701–702, 1964.

[91] S. Makridakis, "Accuracy measures: theoretical and practical concerns," *International journal of forecasting*, vol. 9, no. 4, pp. 527–529, 1993.

[92] C.-Y. Peng and C. J. Wu, "On the choice of nugget in kriging modeling for deterministic computer experiments," *Journal of Computational and Graphical Statistics*, vol. 23, no. 1, pp. 151–168, 2014.

[93] S. Lawrence and C. L. Giles, "Overfitting and neural networks: conjugate gradient and backpropagation," in *Proceedings of the IEEE-INNS-ENNS International Joint Conference on Neural Networks. IJCNN 2000. Neural Computing: New Challenges and Perspectives for the New Millennium*, vol. 1, pp. 114–119, IEEE, 2000.

[94] N. S. Keskar, D. Mudigere, J. Nocedal, M. Smelyanskiy, and P. T. P. Tang, "On large-batch training for deep learning: Generalization gap and sharp minima," *arXiv preprint arXiv:1609.04836*, 2016.

[95] E. N. Ben-Ari and D. M. Steinberg, "Modeling data from computer experiments: an empirical comparison of kriging with mars and projection pursuit regression," *Quality Engineering*, vol. 19, no. 4, pp. 327–338, 2007.

[96] A. Forrester, A. Sobester, and A. Keane, *Engineering Design via Surrogate Modelling: A Practical Guide*. John Wiley & Sons, 2008.

[97] O. Roustant, D. Ginsbourger, and Y. Deville, "Dicekriging, diceoptim: Two R packages for the analysis of computer experiments by kriging-based metamodeling and optimization," 2012.

[98] G. Ke, Q. Meng, T. Finley, T. Wang, W. Chen, W. Ma, Q. Ye, and T.-Y. Liu, "Lightgbm: A highly efficient gradient boosting decision tree," in *Advances in Neural Information Processing Systems*, pp. 3146–3154, 2017.

[99] A. J. Smola and B. Schölkopf, "A tutorial on support vector regression," *Statistics and Computing*, vol. 14, no. 3, pp. 199–222, 2004.

[100] J. Sacks, W. Welch, T. Mitchell, and H. Wynn, "Design and analysis of computer experiments," *Statistical Science*, vol. 4, no. 4, pp. 409–423, 1989.

[101] A. I. Forrester, A. Sóbester, and A. J. Keane, "Multi-fidelity optimization via surrogate modelling," *Proceedings of the royal society a: mathematical, physical and engineering sciences*, vol. 463, no. 2088, pp. 3251–3269, 2007.

[102] M. Kennedy and A. O'Hagan, "Bayesian calibration of computer models," *Journal of the Royal Statistical Society: Series B*, vol. 63, no. 3, pp. 425–464, 2001.

[103] A. Wilson and R. Adams, "Gaussian process kernels for pattern discovery and extrapolation," in *International conference on machine learning*, pp. 1067–1075, PMLR, 2013.

[104] E. Schulz, M. Speekenbrink, and A. Krause, "A tutorial on gaussian process regression: Modelling, exploring, and exploiting functions," *Journal of Mathematical Psychology*, vol. 85, pp. 1–16, 2018.

[105] I. Redko, E. Morvant, A. Habrard, M. Sebban, and Y. Bennani, *Advances in Domain Adaptation Theory*. Elsevier, 2019.

[106] L. Bruzzone and M. Marconcini, "Domain adaptation problems: A DASVM classification technique and a circular validation strategy," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 32, no. 5, pp. 770–787, 2009.

[107] H. Xu and S. Mannor, "Robustness and generalization," *Machine Learning*, vol. 86, no. 3, pp. 391–423, 2012.

[108] J. Schmidhuber, "Deep learning in neural networks: An overview," *Neural Networks*, vol. 61, pp. 85–117, 2015.

[109] L. Deng and D. Yu, "Deep learning: Methods and applications," *Foundations and Trends in Signal Processing*, vol. 7, no. 3–4, pp. 197–387, 2014.

[110] J. H. Friedman, "Multivariate adaptive regression splines," *The Annals of Statistics*, vol. 19, no. 1, pp. 1–67, 1991.

[111] N. Bogduk and S. Mercer, "Biomechanics of the cervical spine. I: Normal kinematics," *Clinical Biomechanics*, vol. 15, no. 9, pp. 633–648, 2000.

[112] D. F. Huelke and G. S. Nusholtz, "Cervical spine biomechanics: a review of the literature," *Journal of Orthopaedic Research*, vol. 4, no. 2, pp. 232–245, 1986.

[113] E. E. Swartz, R. Floyd, and M. Cendoma, "Cervical spine functional anatomy and the biomechanics of injury due to compressive loading," *Journal of Athletic Training*, vol. 40, no. 3, p. 155, 2005.

[114] A. Cappozzo, F. Catani, A. Leardini, M. Benedetti, and U. Della Croce, "Position and orientation in space of bones during movement: Experimental artefacts," *Clinical Biomechanics*,

vol. 11, no. 2, pp. 90–100, 1996.

[115] K. Li, L. Zheng, S. Tashman, and X. Zhang, "The inaccuracy of surface-measured model-derived tibiofemoral kinematics," *Journal of Biomechanics*, vol. 45, no. 15, pp. 2719–2723, 2012.

[116] T.-Y. Tsai, T.-W. Lu, M.-Y. Kuo, and C.-C. Lin, "Effects of soft tissue artifacts on the calculated kinematics and kinetics of the knee during stair-ascent," *Journal of Biomechanics*, vol. 44, no. 6, pp. 1182–1188, 2011.

[117] D. L. Benoit, D. K. Ramsey, M. Lamontagne, L. Xu, P. Wretenberg, and P. Renström, "Effect of skin movement artifact on knee kinematics during gait and cutting motions measured in vivo," *Gait & Posture*, vol. 24, no. 2, pp. 152–164, 2006.

[118] A. Cappozzo, A. Cappello, U. D. Croce, and F. Pensalfini, "Surface-marker cluster design criteria for 3-d bone movement reconstruction," *IEEE Transactions on Biomedical Engineering*, vol. 44, no. 12, pp. 1165–1174, 1997.

[119] A. Cereatti, U. Della Croce, and A. Cappozzo, "Reconstruction of skeletal movement using skin markers: Comparative assessment of bone pose estimators," *Journal of NeuroEngineering and Rehabilitation*, vol. 3, no. 1, pp. 1–12, 2006.

[120] T. Fukaya, H. Mutsuzaki, H. Ida, and Y. Wadano, "Two different protocols for knee joint motion analyses in the stance phase of gait: Correlation of the rigid marker set and the point cluster technique," *Rehabilitation Research and Practice*, vol. 2012, pp. Article ID 586348, 6 pages, 2012.

[121] K. F. Nicholson, R. T. Richardson, E. A. R. van Roden, R. G. Quinton, K. F. Anzilotti, and J. G. Richards, "Machine learning algorithms for predicting scapular kinematics," *Medical Engineering & Physics*, vol. 65, pp. 39–45, 2019.

[122] Y. Zhou, S. Chowdhury, C. Reddy, B. Wan, R. Byrne, W. Yin, and X. Zhang, "A state-of-the-art integrative approach to studying neck biomechanics in vivo," *Science China Technological Sciences*, vol. 63, no. 7, pp. 1235–1246, 2020.

[123] W. J. Anderst, E. Baillargeon, W. F. Donaldson III, J. Y. Lee, and J. D. Kang, "Validation

of a non-invasive technique to precisely measure in vivo three-dimensional cervical spine movement," *Spine*, vol. 36, no. 6, pp. E393–E340, 2011.

[124] M. J. Bey, R. Zauel, S. K. Brock, and S. Tashman, "Validation of a new model-based tracking technique for measuring three-dimensional, in vivo glenohumeral joint kinematics," *Journal of Biomechanical Engineering*, vol. 128, no. 4, pp. 604–609, 2006.

[125] W. Anderst, R. Zauel, J. Bishop, E. Demps, and S. Tashman, "Validation of three-dimensional model-based tibio-femoral tracking during running," *Medical Engineering & Physics*, vol. 31, no. 1, pp. 10–16, 2009.

[126] W. J. Anderst, J. Y. Lee, W. F. Donaldson III, and J. D. Kang, "Six-degrees-of-freedom cervical spine range of motion during dynamic flexion-extension after single-level anterior arthrodesis: comparison with asymptomatic control subjects," *The Journal of Bone and Joint Surgery. American volume*, vol. 95, no. 6, pp. 497–506, 2013.

[127] J. Cobb, "Outline for the study of scoliosis," *Instructional Course Lectures for the American Academy of Orthopaedic Surgeons*, vol. 5, pp. 261–275, 1948.

[128] D. E. Harrison, R. Cailliet, D. D. Harrison, T. J. Janik, and B. Holland, "Reliability of centroid, Cobb, and Harrison posterior tangent methods: Which to choose for analysis of thoracic kyphosis," *Spine*, vol. 26, no. 11, pp. e227–e234, 2001.

[129] T. Vrtovec, F. Pernuš, and B. Likar, "A review of methods for quantitative evaluation of spinal curvature," *European Spine Journal*, vol. 18, no. 5, pp. 593–607, 2009.

[130] H. Liu, Y.-S. Ong, X. Shen, and J. Cai, "When gaussian process meets big data: A review of scalable gps," *IEEE transactions on neural networks and learning systems*, vol. 31, no. 11, pp. 4405–4423, 2020.

[131] M. D. McKay, R. J. Beckman, and W. J. Conover, "Comparison of three methods for selecting values of input variables in the analysis of output from a computer code," *Technometrics*, vol. 21, no. 2, pp. 239–245, 1979.

[132] W. Wang, R. Tuo, and C. Jeff Wu, "On prediction properties of kriging: Uniform error bounds and robustness," *Journal of the American Statistical Association*, vol. 115, no. 530,

pp. 920–930, 2020.

[133] N. I. Achieser, *Theory of approximation*. Courier Corporation, 2013.

[134] Z.-m. Wu and R. Schaback, "Local error estimates for radial basis function interpolation of scattered data," *IMA Journal of Numerical Analysis*, vol. 13, no. 1, pp. 13–27, 1993.

[135] H. Dette and A. Pepelyshev, "Generalized latin hypercube design for computer experiments," *Technometrics*, vol. 52, no. 4, pp. 421–429, 2010.

[136] J. An and A. Owen, "Quasi-regression," *Journal of complexity*, vol. 17, no. 4, pp. 588–607, 2001.

[137] W. J. Welch, R. J. Buck, J. Sacks, H. P. Wynn, T. J. Mitchell, and M. D. Morris, "Screening, predicting, and computer experiments," *Technometrics*, vol. 34, no. 1, pp. 15–25, 1992.

[138] J. C. Helton and F. J. Davis, "Latin hypercube sampling and the propagation of uncertainty in analyses of complex systems," *Reliability Engineering & System Safety*, vol. 81, no. 1, pp. 23–69, 2003.

[139] V. R. Joseph and Y. Hung, "Orthogonal-maximin latin hypercube designs," *Statistica Sinica*, pp. 171–186, 2008.

APPENDIX A

APPENDIX FOR CHAPTER II

A.1 Rate of convergence for additive models

In this section, we state an upper bound of the uniform prediction error of the additive Gaussian process models and its proof. The mathematical tool involved in the proof, namely, the local polynomial reproduction, originates in the scattered data approximation theory. We make substantial improvements to this tool so that it can accommodate the additive correlations. Although the theory for additive Gaussian process models cannot justify the use of the proposed projection pursuit models, this theory is related to the motivation of the proposed model, as it shows the additive models have a better prediction power than isotropic models. Besides, the theory can serve as a justification for the standard additive Gaussian process models, and its variations, such as TAAG by [14].

In Section A.1.1, we introduce the main theorem (Theorem A.1.2) and the required conditions. In Section A.1.2, we establish a major mathematical tool for the proof of Theorem A.1.2. The proof of Theorem A.1.2 is presented in Section A.1.3.

A.1.1 Conditions and theorems

We suppose that the design set $\{x_1, \ldots, x_n\} \subset [0,1]^d$. For any $x = (x_{(1)}, \ldots, x_{(d)})$ and positive integer m, define vector

$$v_m(x) := (1, x_{(1)}, x_{(1)}^2, \ldots, x_{(1)}^m, x_{(2)}, x_{(2)}^2, x_{(2)}^m, \ldots, x_{(d)}, x_{(d)}^2, \ldots, x_{(d)}^m)^T.$$

Clearly, $v_m(x)$ has $md + 1$ entries.

Definition 2. We call the design set $X = \{x_1, \ldots, x_n\} \subset [0,1]^d$ regular with order m, if the matrix $V := (v_m(x_1), \ldots, x_m(x_n))$ is full row rank.

Obviously, a necessary condition for an n-point design set with order m is that $n \geq md + 1$.

In fact, $n \geq md + 1$ is also "almost" a sufficient condition, in the sense that the design set X without mth regularity form a set with Lebesgue measure zero in $\mathbb{R}^{n\times d}$. To see this, it suffices to consider the case $n = md + 1$. Then V becomes a square matrix and the goal is to check whether $\det V = 0$. Noting that $\det V$ is a (nonzero) polynomial of all entries of the design points (and there are in total nd of them), the set $\{X : \det V = 0\}$ forms a $(nd-1)$-dimensional manifold and thus has Lebesgue measure zero. Therefore, mth regularity can be automatically achieved with probability one by random sampling provided that $n \geq md + 1$. Besides, when $n \geq md + 1$, any non-regular design can be rectified by adding a small perturbation.

We also require the following regularity condition.

Condition 1. The entries of each x_j are distinct.

It turns out that the *projection properties* of the design are critical for additive models. Let $x_{k(j)}$ denote the jth entry of x_k.

Definition 3. Define the jth marginal fill distance of a design set $X \subset [0, 1]^d$ as

$$h_j = \max_{-1\leq t\leq 1} \min_{1\leq k\leq n} \|t - x_{k(j)}\|.$$

Latin hypercube designs [131] are ideal in terms of having small marginal fill distances. To further ensure Condition 1 and the regularity condition in Definition 2, we suggest choosing each point of the Latin hypercube design randomly within the corresponding cell. clearly, for this design, we have

$$\max_{1\leq j\leq d} h_j \leq 2/n.$$

As shown in [132], the uniform upper bound of Gaussian process regression is closely tied to its predictive variance

$$Var[Z(x)|Z(x_1), \ldots, Z(x_n)] = \sigma^2(1 - r^T(x)K^{-1}r(x)) := \sigma^2 P^2(x),$$

where r and K are the same as in (1) in the main article. We give a uniform upper bound of

$P^2(x)$ in Theorem A.1.1. Although we are primarily interested in Matérn correlation functions, we consider a more general characterization of one-dimensional correlation functions in terms of their spectral density, i.e, the function f_Φ such that the correlation Φ can be reconstructed by the inverse Fourier transform of f_Φ as

$$\Phi(x) = \int_{\mathbb{R}} f_\Phi(\omega) e^{i\omega x} d\omega,$$

where $i^2 = -1$, and $f_\Phi(\omega)$ is a real symmetric and nonnegative function according to Bochner's Theorem [24; 87]. It is well-known that one-dimensional Matérn correlation in (2) in the main article has spectral density [10; 2; 24]

$$\pi^{-1/2} \frac{\Gamma(\nu + 1/2)}{\Gamma(\nu)} (4\nu\gamma^2)^\nu (4\nu\gamma^2 + \omega^2)^{-(\nu+1/2)}, \tag{A.1}$$

which is bounded above and below by $(1 + \omega^2)^{-(\nu+1/2)}$ multiplying two constants, respectively.

Theorem A.1.1. *Suppose that Condition 1 is fulfilled, the design X is regular with order m, and $h_j \leq m^{-2}/4$ for each $j = 1, \ldots, d$. Define $h = \max_{1 \leq j \leq d} h_j$. Consider Gaussian process regression model with additive correlation defined in (4) in the main article in which Φ_1 has a spectral density f_{Φ_1} with*

$$f_{\Phi_1}(\omega) \leq c(1 + \omega^2)^{-(\nu+1/2)}, \tag{A.2}$$

for constants $c, \nu > 0$ with $\nu + 1/2 \leq m$. Then its predictive variance has the following property:

$$\sup_{x \in [0,1]^d} P(x) \leq C d h^\nu,$$

where C is independent of d and X.

Combining Theorem A.1.1 with Theorem 1 of [132] yields Theorem A.1.2.

Theorem A.1.2. *Under the conditions of Theorem A.1.1 and denote the Gaussian process in Theorem A.1.1 as Z. Let σ^2 be the variance of Z. Then for any $s > 0$, with probability at least $1 - 2\exp\{-s^2/(2\sigma^2 C_1 dh^\nu)\}$,*

$$\sup_{x\in[-1,1]^d} |Z(x) - \hat{Z}(x)| \leq C_2 dh^\nu \sqrt{\log(1/h)} + s,$$

where C_1, C_2 are independent of d and X.

Corollary A.1.1 below is a special case of Theorem A.1.2 with the design been chosen as, for instance, the aforementioned Latin hypercube design.

Suppose Z is a Gaussian process with mean zero and correlation function defined in (4) in the main article, with Φ_1 a Matérn correlation function with smoothness ν. Then under a suitably chosen design of experiment, we have the rate of convergence

$$\sup_{x\in[0,1]^d} |Z(x) - \hat{Z}(x)| = O_p(n^{-\nu}\sqrt{\log n}). \tag{A.3}$$

A.1.2 Simultaneous local polynomial reproduction

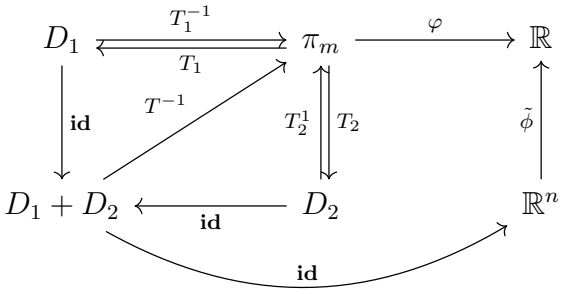
Polynomial reproduction is one of the essential tools to derive error estimates for function approximation. Its main objective is to identify a *stable* scheme to reconstruct polynomials in a finite dimensional space. Specifically, let π_m be the space with real polynomials with degree no more than m. Given design points $\{x_1, \ldots, x_n\} \subset [0,1]^d$ with $n \geq l$, the interest lies in reconstructing any polynomial function in π_l at an untried point x, by identify numbers $\alpha_1, \ldots, \alpha_n$ satisfying the following conditions:

1. (exact recovery) for any $p \in \pi_m$, $p(x) = \sum_{j=1}^n \alpha_j p(x_j)$;
2. (stability) $\sum_{j=1}^n |\alpha_j| < c$ for some c independent of n;
3. (local support) $\alpha_j = 0$ if $|x_j - x| > \rho$ for some (small) $\rho > 0$.

For detailed discussions of local polynomial reproduction, we refer to [87].

It turns out in our analysis that, under the additive context, we need a stronger polynomial reproduction condition called "simultaneous polynomial reproduction", in the sense that we need to recover d polynomials simultaneously with the *same* set of coefficients $\alpha_1, \ldots, \alpha_n$.

It is worth noting that, in our situation, we do not need to construct the coefficients $\alpha_1, \ldots, \alpha_n$ explicitly. We only have to prove the *existence* of these numbers. A nonconstructive approach to get $\alpha_1, \ldots, \alpha_n$ is based on the following observation. Note that x defines a linear functional on π_m as $\varphi(p) := p(x)$. Our goal is to represent $\varphi(p)$ in terms of $p(x_{1(j)}), \ldots, p(x_{n(j)})$ for each j. The commutative diagram below shows the main idea. For simplicity, we show only the case $d = 2$ in this diagram.



Specifically, we take the following steps.

1. Define $T_j : W \to \mathbb{R}^n$ as $T_j(p) = (p(x_{1(j)}), p(x_{2(j)}), \ldots, p(x_{n(j)}))^T$ for each j. Let D_j be the image of T_j. Clearly, T_j is injective under Condition 1. Therefore, the inverse mapping T_j^{-1} exists.

2. We can map D_j to the sum $\sum_{j=1}^d D_j$ using the identity map. Since we assumed that the set set is regular with order m, it can be easily verified that $D_j \cap D_k = V_1$ for any $j \neq k$, where V_1 denotes the vector space $\{(c, c, \ldots, c)^T : c \in \mathbb{R}\}$, i.e., $T_j(\pi_0)$ for any j. Because $T_j|_{\pi_0}$'s are identical for all j, there exists a unique map $T^{-1} : \sum_{j=1}^d D_j \to \pi_m$ such that $T^{-1}|_{D_j} = T_j^{-1}$ for each j.

3. Let I be a compact interval. Now we equip π_m with the $L_\infty(I)$ norm, and equip $\mathbb{R}^n$ as well as its subspaces D_j and $\sum_{j=1}^d D_j$ the l_∞ norm. Because any $\xi \in \sum_{j=1}^d D_j$ can be represented

as $\xi = \sum_{j=1}^{d} \xi_j$ with $\xi_j \in D_j$, we have

$$\begin{aligned} \|T^{-1}\| &:= \sup_{\xi \in \sum_{j=1}^{d} D_j} \frac{\|T^{-1}\xi\|_{L_\infty(I)}}{\|\xi\|_\infty} \\ &= \sup_{\xi_j \in D_j} \frac{\|T^{-1} \sum_{j=1}^{d} \xi_j\|_{L_\infty(I)}}{\| \sum_{j=1}^{d} \xi_j\|_\infty} \\ &\leq \sum_{j=1}^{d} \sup_{\xi_j \in D_j} \frac{\|T^{-1}\xi_j\|_{L_\infty(I)}}{\|\xi_j\|_\infty} = \sum_{j=1}^{d} \|T_j^{-1}\|. \end{aligned} \tag{A.4}$$

4. Define map $\phi : \sum_{j=1}^{d} D_j \to \mathbb{R}$ as $\phi = \varphi \circ T^{-1}$. Then we have

$$\|\phi\| = \|\varphi \circ T^{-1}\| \leq \|\varphi\| \cdot \|T^{-1}\| = \|\varphi\| \cdot \|T^{-1}\| \leq \|T^{-1}\|,$$

where the last inequality follows from the fact that

$$\|\varphi\| = \sup_{p \in \pi_m} |p(x)|/\|p\|_{L_\infty} \leq 1.$$

5. Because $\sum_{j=1}^{d} D_j$ is a subspace of $\mathbb{R}^d$, according to Hahn-Banach theorem, there exists a map $\tilde{\phi} : \mathbb{R}^n \to \mathbb{R}$, such that $\tilde{\phi}|_{\sum_{j=1}^{n} D_j} = \phi$ and

$$\|\tilde{\phi}\| = \|\phi\| \leq \|T^{-1}\|. \tag{A.5}$$

Now we examine what we have got. Because ϕ is a linear functional on $\mathbb{R}^n$, we can express ϕ as

$$\phi(\alpha_1, \ldots, \alpha_n) = \sum_{j=1}^{n} c_j \alpha_j.$$

Then

$$\|\phi\| = \sup_{\alpha_1, \ldots, \alpha_n} \frac{|\sum_{j=1}^{n} c_j \alpha_j|}{\|(\alpha_1, \ldots, \alpha_n)\|_\infty} = \sum_{j=1}^{n} |c_j|. \tag{A.6}$$

Combining (A.4)-(A.6) yields $\sum_{j=1}^{n} |c_j| \leq \sum_{j=1}^{d} \|T_j^{-1}\|$. Our next goal is to bound $\|T_j^{-1}\|$.

Lemma A.1.3. *Suppose that Condition 1 is fulfilled and the length of I is no less than $4m^2 h_j$. Then $\|T_j^{-1}\| \leq 2$.*

Proof. It suffices to prove that

$$\|T_j\| = \sup_{1 \leq k \leq n, p \in \pi_m, \|p\|_{L_\infty(I)}=1} p(x_{k(j)}) \geq 1/2. \tag{A.7}$$

Let d_0 be the length of I. Markov's inequality [133] for an algebraic polynomial $p \in \pi_m$ states

$$\max_{t \in I} |p'(t)| \leq \frac{2}{d_0} m^2 \max_{t \in I} p(t). \tag{A.8}$$

Choose $p \in \pi_m$ with $\|p\|_{L_\infty(I)} = 1$. Because I is compact, there exists $t_0 \in I$ such that $|p(t_0)| = 1$. The mean value theorem and (A.8) yields

$$|p(t_0) - p(t)| \leq 2m^2 |t_0 - t|/d_0 \leq |t_0 - t|/(2h_j) \tag{A.9}$$

for all $t \in I$. By the definition of h_j that there exists k, so that $|x_{k(j)} - t_0| \leq h_j$, which, together with (A.9), yields

$$|1 - p(x_{k(j)})| \leq 1/2.$$

Thus $|p(x_{k(j)})| \geq 1/2$, which proves (A.7). □

We now arrive at our main conclusion in this section.

Theorem A.1.4. *Suppose that Condition 1 is fulfilled, the design X is regular with order m, and $h_j \leq m^{-2}/4$ for each $j = 1, \ldots, d$. Then for each $x \in [0,1]^d$, there exist constants $\alpha_1, \ldots, \alpha_n \in \mathbb{R}$, such that*

$$p_j(x) = \sum_{k=1}^{n} \alpha_k p_j(x_{k(j)}),$$

for any $p_i \in \pi_m$ *with* $j = 1, \ldots, d$. *In addition, we have* $\sum_{k=1}^{n} |\alpha_k| \leq 2d$, *and* $\alpha_k = 0$ *if* $|x_{k(j)} - x_{(j)}| \geq 4m^2 h_j$, *where* $x_{(j)}$ *denotes the* j*th entry of* x.

Proof. Take any interval $I \subset [-1, 1]$ with length $4m^2 h_j$. Imagine a new set of design points $X_{new} := \{x_k \in X : x_{k(j) \in I_j, j=1,\ldots,d}$. Because X_{new} is a subset of X, Condition 1 also holds for X_{new}. Thus we can invoke Lemma A.1.3 with the new design X_{new}. Denote $X_{new} = \{x_{n_1}, \ldots, x_{n_m}\}$. Lemma A.1.3 implies that there exist $\alpha_{n_1}, \ldots, \alpha_{n_m}$ such that

$$p_j(x) = \sum_{k=1}^{n} \alpha_{n_k} p_j(x_{n_k(j)}),$$

and $\sum_{k=1}^{n} |\alpha_{n_k}| < 2d$. Now for any $x_k \in X \setminus X_{new}$, we set $\alpha_k = 0$. Then we arrive at all desired results. □

A.1.3 Proof of Theorem A.1.1

The mathematical tools for proving Theorem A.1.1 include the simultaneous polynomial reproducing we developed in Section A.1.2 and the Fourier transforms. The idea is inspired by [134].

Via direct calculations, we can verify that

$$P^2(x) = 1 - r^T(x) K^{-1} r(x) = \min_{u \in \mathbb{R}^n} 1 - 2u^T r(x) + u^T K u =: \min_{u \in \mathbb{R}^n} \mathcal{Q}(u).$$

Because the correlation function is additive, we can write $\mathcal{Q}(u) = \sum_{j=1}^{d} \mathcal{Q}_j(u)/d$ with

$$\mathcal{Q}_j(\alpha) = 1 - 2u^T r_j(x) + u^T K_j u,$$

where $r_j(x) = (\Phi_1(x_{1(j)} - x_{(j)}), \ldots, \Phi_n(x_{1(j)} - x_{(j)}))$, $K_j = (\Phi_1(x_{k(j)} - x_{l(j)}))_{kl}$ and $x_{(j)}$ denotes the $j$the entry of x.

Next we can represent $\mathcal{Q}_j(u)$ in terms of the spectral density f_{Φ_1}. It can be verified by elemen-

tary calculations that

$$\begin{aligned}\mathcal{Q}_j(u) &= \int_{\mathbb{R}} \left| e^{i\omega x_{(j)}} - \sum_{k=1}^{n} u_k e^{i\omega x_{k(j)}} \right|^2 f_{\Phi_1}(\omega) d\omega \\ &= \int_{\mathbb{R}} \left| 1 - \sum_{k=1}^{n} u_k e^{i\omega (x_{k(j)} - x_{(j)})} \right|^2 f_{\Phi_1}(\omega) d\omega,\end{aligned}$$

where u_k denotes the kth entry of u.

Now we choose u_k as α_k defined by Theorem A.1.4 satisfying

$$p(0) = \sum_{k=1}^{n} \alpha_k p_0(x_{k(j)} - x_{(j)}) \tag{A.10}$$

for all $p \in \pi_m$. Consider the mth order Taylor polynomial of e^z, which is $p_0(z) = \sum_{k=0}^{m} z^k/k!$. We write $e^z = p_0(z) + e(z)$. For any $z \in \mathbb{C}$, we can bound $|e(z)|$ as

$$\begin{aligned}|e(z)| &= \left| \sum_{k=m+1}^{\infty} z^k/k! \right| \leq \sum_{k=m+1}^{\infty} |z|^k/k! \\ &= e^{|z|} - p_0(|z|) \leq e^{|z|} |z|^{m+1}/(m+1)!, \end{aligned} \tag{A.11}$$

where the last inequality follows from Taylor's theorem. Now we have

$$\begin{aligned}\sum_{k=1}^{n} \alpha_k e^{i\omega (x_{k(j)} - x_{(j)})} &= \sum_{k=1}^{n} \alpha_k \left\{ p_0\left(\omega(x_{k(j)} - x_{(j)})\right) + e\left(\omega(x_{k(j)} - x_{(j)})\right) \right\} \\ &= 1 + \sum_{k=1}^{n} \alpha_k e\left(\omega(x_{k(j)} - x_{(j)})\right),\end{aligned}$$

where the last equality follows from (A.10) and the fact that $p_0(\omega \cdot)$ is also a polynomial.

New it remains to bound

$$\begin{aligned}\mathcal{Q}_j(u) &= \int_{\mathbb{R}} \left| \sum_{k=1}^{n} \alpha_k e\left(\omega(x_{k(j)} - x_{(j)})\right) \right|^2 f_{\Phi_1}(\omega) d\omega \\ &= \left(\int_{|\omega| \le 1/h_j} + \int_{|\omega| > 1/h_j} \right) \left| \sum_{k=1}^{n} \alpha_k e\left(\omega(x_{k(j)} - x_{(j)})\right) \right|^2 f_{\Phi_1}(\omega) d\omega \\ &=: I_1 + I_2.\end{aligned}$$

To bound I_1, we use (A.11) to get

$$\begin{aligned}I_1 &\le \int_{|\omega| \le 1/h_j} \left| \sum_{k=1}^{n} \alpha_k e^{\omega(x_{k(j)} - x_{(j)})} |\omega(x_{k(j)} - x_{(j)})|^{m+1} \right|^2 f_{\Phi_1}(\omega) d\omega/(m+1)! \\ &\le \int_{|\omega| \le 1/h_j} \left| \sum_{k=1}^{n} \alpha_k e^{4m^2} |4\omega m^2 h_j|^{m+1} \right|^2 f_{\Phi_1}(\omega) d\omega/(m+1)! \\ &\le C_1 \int_{|\omega| \le 1/h_j} \left(\sum_{k=1}^{n} |\alpha_k| \right)^2 |\omega|^{2m+2} h_j^{2m+2} f_{\Phi_1}(\omega) d\omega \\ &\le C_2 d^2 h_j^{2\nu - 2m - 2} h_j^{2m+2} = C_2 d^2 h_j^{2\nu},\end{aligned}$$

where the second inequality follows from the fact that $\alpha_k = 0$ if $x_{k(j)} - x_{(j)} \ge 4mh_j$; and the last inequality follows from $\sum_{k=1}^{n} |\alpha_k| \le 2d$, (A.2) and elementary calculus.

To bound I_2, we use the fact that

$$\left| 1 - \sum_{k=1}^{n} u_k e^{i\omega(x_{k(j)} - x_{(j)})} \right| \le 1 + \sum_{k=1}^{n} |u_k| \le 2d + 1$$

to obtain

$$I_2 \le \int_{|\omega| > 1/h_j} (2d+1)^2 f_{\Phi_1}(\omega) d\omega \le C_3 d^2 h_j^{2\nu}.$$

Then the proof is completed.

A.2 More numerical studies

In this section, more numerical studies are conducted. The results show the superiority of the PPGPR method.

A.2.1 Performance of GPR and PPGPR for more simulation functions

We compare the performance of GPR and PPGPR for three more simulation functions: Dette Pepelyshev (2010) curved function [135], Robor arm function [136] and Welch function [137; 95]. The training sets are generated by Halton sequences [90] with sample size $5d$, and 500 random samples are generated as the testing set in this experiment. We use the package Dicekriging [97] to implement GPR (with product kernel). Table A.1 shows the performance measured by MAPE of GPR and PPGPR. One can see that the proposed PPGPR overperforms GPR a lot for these three functions.

Table A.1: MAPEs of GPR with product correlations and PPGPR for three different simulation functions

	Dette Pepelyshev (2010) curved ($d = 3$)	Robot arm ($d = 8$)	Welch ($d = 20$)
GPR	0.413	0.751	1.246
PPGPR	0.258	0.738	1.066

A.2.2 Performance of GPR and PPGPR under Latin hypercube designs with different sizes

We compare the performance of GPR and PPGPR under Latin hypercube designs [138] with different sample sizes. The Dette Pepelyshev (2010) curved function [135] is chosen as the simulation function. The package lhs (see https://cran.r-project.org/web/packages/lhs/index.html for details) in R is used to generate the Latin hypercube designs using the maximin criterion [139]. The size of the testing set is 500. Figure A.1 shows the MAPEs of GPR and PPGPR under the sample sizes from 40 to 120. It can be easily seen that the PPGPR has lower MAPEs than GPR most time. GPR has a lower MAPE only when the sample size is 63. Figure A.1 proves the superiority of the proposed PPGPR over GPR under the Latin hypercube design with different sample sizes.

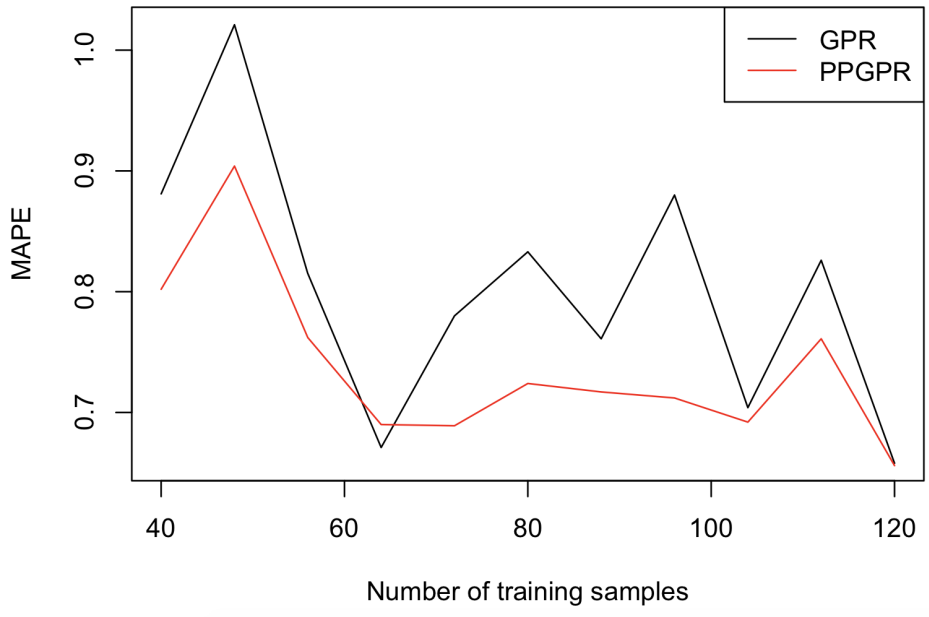


Figure A.1: MAPEs of PPGPR and GPR for Dette Pepelyshev (2010) curved function under Latin hypercube designs with different sample sizes

A.2.3 Performance of PPGPR under different initial ws

In this section, we investigate the performance of the proposed PPGPR starting with different initial ws. The Dette Pepelyshev (2010) curved function [135] is chosen as the simulation function. The training sets are generated by Halton sequences [90] with sample size $5d$ and 500 random samples are chosen as the testing set in this experiment. We fix some hyper-parameters, i.e., $\eta = 10^{-9}$, $P = 150$, and Matérn kernel with $\nu = 2.5$. For each $M = 7, 14, 21, 28, 35, 42, 49, 56$, we train seven PPGPR models with different initial ws and test them on the testing set, as shown in Figure A.2. One can see that the performance of PPGPR models starting with different ws highly varies when $M < 20$, but when $M > 20$ the performance are very close. Since we recommend a large M when implementing PPGPR, the initial ws may not influence the results much.

A.2.4 Computational cost of GPR and PPGPR

We study the computational cost of the proposed PPGPR using Dette Pepelyshev (2010) curved function (d=3). The training samples are generated by the Halton sequences [90] with

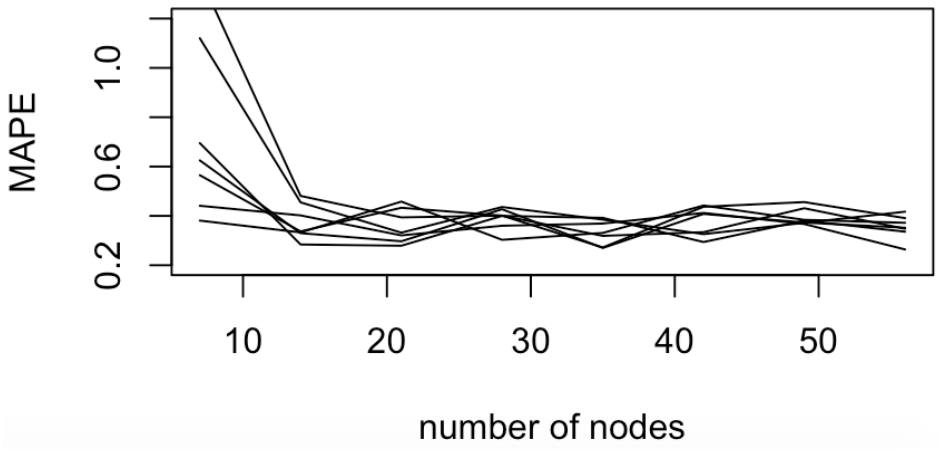


Figure A.2: MAPEs of PPGPR for Dette Pepelyshev (2010) curved function under different initial ws

sample size $5d$. In this experiment, GP costs 0.02s to run while the computational cost of PPGPR ($\eta = 10^{-9}$, $P = 150$, Matérn kernel with $\nu = 2.5$) under different numbers of nodes are shown in Table A.2. The proposed PPGPR is slower than ordinary Gaussian process regression, but it is still affordable for moderate datasets. The higher computational cost of the proposed method is because 1) it requires nonlinear optimization, and 2) dimension expansion incurs additional computational costs.

Table A.2: Computational cost of PPGPR for Dette Pepelyshev (2010) curved function under different number of nodes

Number of nodes	7	14	21	28	35	42	49	56
Time(s)	8.6	13.6	24.5	32.5	40.3	48.4	56.5	64.7

APPENDIX B

APPENDIX FOR CHAPTER III

B.1 Proof for Theorem 1-3

In this section, we will show the proof for them 1-3.

B.1.1 Theorem 1

Proof of Theorem 3.4.1. By definition,

$$\begin{aligned}\hat{y}_K(x) &= \mathbb{E}\left[\sum_{i=0}^{K}\delta_i(x)\Big|y_K(\mathcal{X}_K),...,y_0(\mathcal{X}_0)\right]\\ &= \sum_{i=0}^{K}\mathbb{E}\left[\delta_i(x)\Big|y_K(\mathcal{X}_K),...,y_0(\mathcal{X}_0)\right]\end{aligned}$$

Because δ_i's are mutually independent, for each i we can get

$$\begin{aligned}\mathbb{E}\left[\delta_i(x)\Big|y_K(\mathcal{X}_K),\ldots,y_0(\mathcal{X}_0)\right] &= \mathbb{E}\left[\delta_i(x)\Big|y_K(\mathcal{X}_K)-y_{K-1}(\mathcal{X}_k),\ldots,y_1(\mathcal{X}_1)-y_0(\mathcal{X}_0),y_0(\mathcal{X}_0)\right]\\ &= \mathbb{E}\left[\delta_i(x)\Big|y_i(\mathcal{X}_i)-y_{i-1}(\mathcal{X}_i)\right].\end{aligned}$$

Then we can rewrite the $\hat{y}_K(x)$ as

$$\hat{y}_K(x) = \sum_{i=0}^{K}\mathbb{E}\left[\delta_i(x)\Big|y_i(\mathcal{X}_i)-y_{i-1}(\mathcal{X}_i)\right]$$

□

B.1.2 Theorem 2

Proof of Theorem 3.4.2. By definition,

$$\begin{aligned}
\mathbb{E}\left\|y_K - y_\infty\right\|_{L2}^2 &= \mathbb{E}\left\|\sum_{i=K+1}^{\infty} \delta_i\right\|_{L2}^2 \\
&= \sum_{i=K+1}^{\infty} \mathbb{E}\left\|\delta_i\right\|_{L_2}^2 \\
&= \sum_{i=K+1}^{\infty} \mathbb{E}(\int_{\mathfrak{X}} \delta_i^2 dx) \\
&= \sum_{i=K+1}^{\infty} \int_{\mathfrak{X}} \mathbb{E}(\delta_i^2) dx \\
&= \sum_{i=K+1}^{\infty} C_s \lambda^{2i} \sigma^2 \\
&= C_l^2 \lambda^{2K} \sigma^2.
\end{aligned}$$

where $\mathfrak{X}$ denotes the input domain of x, C_s is the area of $\mathfrak{X}$ and constant $C_l = \sqrt{C_s}/(1-\lambda)$. □

B.1.3 Theorem 3

Proof of Theorem 3.4.3. Based on the Thm 1, we have

$$\begin{aligned}
\mathbb{E}\left\|y_K - \hat{y}_K\right\|_{L_2}^2 &= \mathbb{E}\left\|\sum_{i=0}^{k} (I_i(\delta_i) - \delta_i)\right\|_{L_2}^2 \\
&= \int_{\mathfrak{X}} \sum_{i=0}^{k} (I_i(\delta_i) - \delta_i)^2 dx \\
&= \int_{\mathfrak{X}} \sum_{i=0}^{k} \mathbb{E}((I_i(\delta_i) - \delta_i)^2) dx \\
&\leq \sum_{i=0}^{k} p\sigma^2 \lambda^{2i} n_i^{-\nu/d},
\end{aligned}$$

where n_i is the number of samples in the i-th step, p is a constant only based on the correlation function, d is the dimension of the input space and ν is a constant. □

B.2 Solve the objective function

Consider the optimization problem

$$\min_{n_i} \quad \sum_{i=0}^{K} n_i C_i \tag{B.1}$$

$$\text{s.t.} \quad \sum_{i=0}^{K} \lambda^{2i} \sigma^2 n_i^{-2\nu/d} \leq \epsilon^2/2 \tag{B.2}$$

We form the Lagrange function of (B.1) as

$$L(n_0, n_1, ..., n_K, \eta) = \sum_{i=0}^{K} n_i C_a a^i + \eta(\sum_{i=0}^{K} p\lambda^{2i}\sigma^2 n_i^{-2\nu/d} - \epsilon^2/2). \tag{B.3}$$

To find a relative extremum, we require $\nabla L = 0$, leading to the $K+2$ conditions

$$\frac{\partial L}{\partial n_i} = C_a a^i - \frac{2p\eta\nu}{d}\lambda^{2i}\sigma^2 n_i^{-(2\nu/d+1)} = 0, \quad i = 0, 1, ..., K \tag{B.4}$$

$$\frac{\partial L}{\partial \eta} = \sum_{i=0}^{K} p\lambda^{2i}\sigma^2 n_i^{-2\nu/d} - \epsilon^2/2 = 0. \tag{B.5}$$

Solve the $(K+1)$ equations in (B.4) we can get

$$n_i = (\frac{pC_a d}{2\nu\sigma^2}\frac{a^i}{\lambda^{2i}}\eta^{-1})^{-\frac{d}{d+2\nu}}, \quad i = 0, 1, ..., K. \tag{B.6}$$

Substitute (B.6) into (B.5) we have

$$\eta = \frac{C_a d}{2\nu}(\sigma^2)^{\frac{d}{2\nu}}(\frac{p\epsilon^2}{2S})^{-\frac{d+2\nu}{2\nu}}, \tag{B.7}$$

where

$$S = \sum_{i=0}^{K} (a^i)^{\frac{2\nu}{d+2\nu}} (\lambda^{2i})^{\frac{d}{d+2\nu}}.$$

Substitute (B.7) into (B.6) we get

$$n_i = (\frac{\epsilon^2}{2p\sigma^2 S})^{-\frac{d}{2\nu}} (\frac{a^i}{\lambda^{2i}})^{-\frac{d}{d+2\nu}}, \quad i = 0, 1, ..., K. \tag{B.8}$$

Since n_i is an integer, we choose

$$n_i^* = \lceil n_i \rceil. \tag{B.9}$$

B.3 Analysis of cost for the MLGP

In this section, we make the analysis of the cost based on the K and n_i^* given in the above section. The total cost is given by

$$C_\epsilon = \sum_{i=0}^{K} n_i^* C_i \leq \sum_{i=0}^{K} n_i^* C_a a^i.$$

Since $\lceil n_i \rceil \leq n_i + 1$, we have

$$\begin{aligned} C_\epsilon &\leq \sum_{i=0}^{K} Ca(n_i + 1)a^i \\ &\eqsim \sum_{i=0}^{K} (S\sigma^2(\epsilon^2/2)^{-1})^{\frac{d}{2\nu}} \sum_{i=0}^{K} (a^{\frac{2\nu}{d+2\nu}} \lambda^{\frac{2d}{d+2\nu}})^i + \sum_{i=0}^{K} a^i \\ &= (2\sigma^2\epsilon^{-2})^{\frac{d}{v}} S^{\frac{d+2\nu}{2\nu}} + \sum_{i=0}^{K} a^i. \end{aligned} \tag{B.10}$$

We denote the first part in the right hand side of (B.10) as part (I) and the second part as (II), and consider them separately.

If $(a^i)^{\frac{2\nu}{d+2\nu}}(\lambda^{2i})^{\frac{d}{d+2\nu}} = 1$ (i.e., $2\alpha\nu = d\beta$), then

$$\begin{aligned}(I) &= (2\sigma^2\epsilon^{-2})^{\frac{d}{v}}(K+1)^{\frac{d+2\nu}{2\nu}} \\ &\asymp \epsilon^{-d/\nu}(2\log_{\lambda^2}\frac{\epsilon}{2C_l\sigma})^{\frac{d+2\nu}{2\nu}} \\ &\asymp \epsilon^{-d/\nu}|\ln\epsilon^2|^{\frac{d+2\nu}{2\nu}};\end{aligned}$$

$$\begin{aligned}(II) &= \frac{a^K(1-a^{-(K+1)})}{1-a^{-1}} \\ &\asymp a^K \asymp (\epsilon^2)(\ln a/\ln\lambda^2) \\ &= \epsilon^{-d/\nu}.\end{aligned}$$

Since $\epsilon^{-d/\nu}|\ln\epsilon^2|^{\frac{d+2\nu}{2\nu}} > \epsilon^{-d/\nu}$, $C_\epsilon \lesssim \epsilon^{-d/\nu}|\ln\epsilon^2|^{\frac{d+2\nu}{2\nu}}$.

If $(a^i)^{\frac{2\nu}{d+2\nu}}(\lambda^{2i})^{\frac{d}{d+2\nu}} < 1$ (i.e., $2\alpha\nu < d\beta$), S will converge to a limitation which is independent of K. As a result, we have

$$(I) \lesssim \epsilon^{-d/\nu};$$

$$(II) \lesssim \epsilon^{-2\alpha/\beta}.$$

Since $d/2\nu > \alpha/\beta$, $C_\epsilon \lesssim \epsilon^{-d/\nu}$.

If $(a^i)^{\frac{2\nu}{d+2\nu}}(\lambda^{2i})^{\frac{d}{d+2\nu}} > 1$ (i.e., $2\alpha\nu > d\beta$), we have

$$\begin{aligned}(I) &\lesssim \epsilon^{-d/\nu}(\exp\frac{2\alpha\nu-d\beta}{d+2\nu})^{\frac{K(d+2\nu)}{2\nu}} \\ &\lesssim \epsilon^{-d/\nu}\exp(-\frac{(\ln\epsilon^2(2\alpha\nu-d\beta))}{2\nu\beta}) \\ &= \epsilon^{-2\alpha/\beta};\end{aligned}$$

$$(II) \asymp \epsilon^{-2\alpha/\beta}.$$

As a result, $C_\epsilon \lesssim \epsilon^{-2\alpha/\beta}$.

In summary,

$$C_\epsilon \lesssim \begin{cases} \epsilon^{-d/\nu}, & 2\alpha\nu < d\beta \\ \epsilon^{-d/\nu} |\ln \epsilon^2|^{\frac{d+2\nu}{2\nu}}, & 2\alpha\nu = d\beta \\ \epsilon^{-2\alpha/\beta}, & 2\alpha\nu > d\beta \end{cases} \tag{B.11}$$

B.4 Analysis of cost for single-level

The error caused by single-level approximation can be bounded as

$$\mathbb{E}(\left\|y_\infty - \hat{y}_K^{SL}\right\|^2) = \mathbb{E}(\left\|y_\infty - y_K^{SL}\right\|^2 + \left\|y_K^{SL} - \hat{y}_K^{SL}\right\|^2), \tag{B.12}$$

where $\hat{y}_K^{SL} = I(\delta_K)$. Then the objective function for this single-level approximation can be summaried as

$$\min_{K, n_K} \quad n_K C_K \tag{B.13}$$

$$\text{s.t.} \quad \mathbb{E}\left\|y_\infty - \hat{y}_K^{SL}\right\|_{L^2}^2 \leq \epsilon \tag{B.14}$$

Based on Thm.1 and Thm.2, we have

$$\mathbb{E}\left\|y_\infty - \hat{y}_K^{SL}\right\|^2 \gtrsim \lambda^{2K}\sigma^2 + \sigma^2 n_K^{-2\nu/d}, \tag{B.15}$$

where n_K is the number of samples. It should be sufficient to bound the upper bound of the error by ϵ, that is,

$$\lambda^{2K} + n_K^{-2\nu/d} \lesssim \epsilon^2. \tag{B.16}$$

From the (B.16), we can get the lower bound for n_K, that is,

$$n_K \gtrsim (\epsilon^2 - \lambda^{2K})^{-d/2\nu}. \tag{B.17}$$

Then we have the lower bound for the cost of single-level approximation:

$$C_\epsilon^{SL} = n_K C_k \gtrsim a^K (\epsilon^2 - \lambda^{2K})^{-d/2\nu}. \tag{B.18}$$

We denote $a = \exp(\alpha)$ and $\lambda^2 = \exp(-\beta)$ for convenience, $\alpha, \beta > 0$. Then wen can rewrite (B.18) as

$$\ln C_\epsilon^{SL} \gtrsim \alpha K - \frac{d}{2\nu} \ln(\epsilon^2 - e^{-\beta K}). \tag{B.19}$$

Now we want to find the lower bound of C_ϵ^{SK}, that is, we need to find the lower bound for the right hand side of (B.19). Consider the following function with respect to K:

$$T(K) = \alpha K - \frac{d}{2\nu} \ln(\epsilon^2 - e^{-\beta K}). \tag{B.20}$$

Set the derivative of T with respect to K equal to 0:

$$\frac{dT}{dK} = \alpha + \frac{d\beta}{2\nu}(1 - \frac{\epsilon^2}{\epsilon^2 - e^{-\beta K}}) = 0. \tag{B.21}$$

We can get the solution $K_0 = -\frac{1}{\beta} \ln \frac{2\epsilon^2 \alpha\nu}{2\alpha\nu + \beta d}$. When $K < K_0$, $T(K)$ is decreasing and when $K > K_0$, $T(K)$ is increasing. Substitute $K = 0$ into $T(K)$, we can get the lower bound of the cost for the single-level:

$$C_\epsilon^{SL} \gtrsim \epsilon^{-(\frac{d}{\nu} + \frac{2\alpha}{\beta})}. \tag{B.22}$$