ON ADAPTIVE KERNEL LEARNING FOR NONPARAMETRIC REGRESSION, CLASSIFICATION AND BEYOND

A PREPRINT

Shuntuo Xu
KLATASDS-MOE
School of Statistics
East China Normal University
Shanghai, China
oaksword@163.com

Zhou Yu *
KLATASDS-MOE
School of Statistics
East China Normal University
Shanghai, China
zyu@stat.ecnu.edu.cn

Yan Zhong
KLATASDS-MOE
School of Statistics
East China Normal University
Shanghai, China
yzhong@fem.ecnu.edu.cn

January 19, 2024

ABSTRACT

Weighted average estimation is a powerful methodology of supervised learning that has been extensively developed over time. In this paper, we proposed a novel tool, named Adank, for performing weighted average estimation that leverages the exceptional flexibility of neural networks. Adank entails the establishment of weights through a generalized kernel function that is implemented via neural network techniques. The kernel learned serves as an inherent association between two inputs, especially as an adaptive similarity metric in the context of classification. Notably, our numerical results demonstrate that Adank is superior to existing, cutting-edge techniques, particularly when the sample size is relatively limited. Furthermore, our study incorporates an extensive theoretical investigation of excess risk, for which we have obtained a standard minimax optimal upper bound under specific conditions.

1 Introduction

Weighted average estimation is an essential technique in nonparametric estimation, which has found widespread applications across various domains Bataillou et al. [1995], Wang et al. [2005], Jiang et al. [2012]. In supervised learning, the primary objective of using the weighted average estimation method is to construct an estimator that is formulated as a weighted average of target values. The establishment of weights is typically achieved using a suitable kernel that quantifies the level of similarity between two predictors. This presupposes that predictors that are close to each other will produce similar outputs, with the closeness being measured by the chosen kernel. Therefore, the selection of the kernel is crucial in this process. Based on the approaches to constructing kernels, the weighted average estimation technique consists of two categories: the deterministic kernel and the adaptive kernel.

The deterministic kernel-based methods concentrate on the spatial distance between two predictors. Furthermore, the corresponding weights are typically presented explicitly, potentially incorporating hyperparameters. Noteworthy works in this domain include Nadaraya-Waston estimation, k-nearest neighbor estimation, and kernel ridge regression in a reproducing kernel Hilbert space (RKHS) Cai [2001], Kramer and Kramer [2013], Vovk [2013]. On the other hand, in adaptive kernel-based methods, researchers strive to capture the intrinsic relationship between predictors and targets. To accomplish this, the methods formulated in this framework integrate the pertinent information of both predictors and targets into the generation of the kernels. Consequently, the weights obtained depend on both predictors and targets. Works that have successfully advanced the application of this approach include tree-based methods Safavian and Landgrebe [1991], Biau and Scornet [2016].

Although deterministic kernels demonstrate the convenience and interpretability in various methods, they suffer from inherent limitations related to their lack of flexibility. Specifically, deterministic kernels are unable to discern the relative importance of individual predictor variables, which leads to the potential for underfitting. Decision trees, on the other hand, provide an adaptive kernel solution that offers a graphical representation of a set of rules guiding decision-making.

Through their use of recursive partitioning of the data into subsets based on the most significant attribute or feature, decision trees can construct a model that predicts the target variable more accurately. However, decision trees can be limited in their capacity for generalization due to a proclivity towards overfitting, leading to poor performance on new data Timofeev [2004], Leiva et al. [2019]. To address these limitations, ensemble methods such as random forest, XGBoost, and LightGBM have been developed to enhance the tree-based models' capacity for better generalization. However, it should be noted that the establishment of trees often involves a compromise against exhaustive exploration, and tree-based methods are ill-suited for handling tensor inputs, such as images.

The purpose of this paper is to provide a new approach to constructing kernels that exhibits optimal flexibility. In practice, determining the optimal kernel function is often challenging due to a lack of prior information. Our novel methodology involves using neural networks to generate the kernel and then subsequently training the network through a supervised loss function. The exceptional approximation capabilities of neural networks enable them to represent complex functions Hornik [1993], Csáji et al. [2001], Elbrächter et al. [2019], DeVore et al. [2021], Shen et al. [2021a], which is a valuable asset in our pursuit of accurately representing kernels. Moreover, recent developments in various neural network variants, such as multilayer perceptrons, convolutional neural networks, and graph neural networks Gu et al. [2018], Wu et al. [2020], have enabled the processing of diverse input categories. Therefore, the utilization of neural networks also enhances the flexibility of our method in handling multiple types of inputs. We believe that the integration of neural networks in our current work will lead us to engage in deeper exploration and achieve more appropriate weights.

Upon further theoretical analysis of our proposed methodology, we discovered a significant correlation with the selection of an adaptable kernel for functional reproducing in RKHS. Furthermore, unlike traditional approaches, our methodology does not impose constraints on the nonnegativity of kernels whenever there is no prior information that indicates the necessity of nonnegativity, thereby enhancing its potential for nuanced discoveries. We found that our approach enables the obtained kernel to take the form of a unique reproducing kernel, a transformation of the conditional mean, or a measurement that captures the similarity between two inputs, thus providing even greater flexibility in the modeling process. This higher degree of flexibility enables our methodology to offer an improved representation of the underlying data, which is critical for accurate prediction and estimation processes. Consequently, our findings offer crucial insights into the potential applications of our proposed methodology in the weighted average estimation landscape, and pave the way for new avenues of research and experimentation.

Our contributions are delineated below:

- Our work introduces a novel and unified framework for conducting weighted average estimation. This framework demonstrates a strong ability of generalization across various types of input and output data, including Euclidean vectors, images, graphs, and non-Euclidean elements.
- We developed a strong theoretical foundation for our proposed method. Our analysis shows that the excess risk induced by our estimator achieves the minimax optimal rate of upper bound under certain smooth assumptions. In addition, a comprehensive set of experiments was presented to demonstrate the highly competitive capacity of our method in comparison to other state-of-the-art methods.
- Our work aims to incorporate neural networks with classical weighted average formula, thus inherently combining the flexibility of deep learning with the smoothness of local representation. For the compound model, we designed a query-based algorithm for efficient optimization.

The rest sections of this paper are organized as follows. In Section 2, we employ mathematical models to explicate the relevant literature in detail and specifically concentrate on the development of weights. Section 3 comprehensively delineates our approach and its implementation. To demonstrate the predictive capabilities of our methodology, Section 4 presents a multitude of experimental results. Theoretical analysis is subsequently illustrated in Section 5. Lastly, in Section 6, we provide a succinct conclusion summarizing the key contributions of our work and outlining the potential implications of our findings. All the detailed mathematical derivations are compiled in the Appendix.

2 Related works

The primary objective of the weighted average estimation framework is to generate accurate predictions based on a collection of labeled training data. To be more precise, when given a cluster of labeled data $(X_1, Y_1), \ldots, (X_n, Y_n)$, the anticipated target value Y of a new observation X is estimated using the formula

$$\tilde{Y} = \sum_{i=1}^{n} w_i Y_i, \tag{1}$$

where w_i quantifies the contribution of the i-th observation. It should be noted that equation (1) may serve as the ultimate outcome in regression analysis, but some adjustments are needed to obtain a definite label in classification tasks. In binary classification, where the target $Y \in \{-1, 1\}$, the final estimate $\hat{Y}$ is determined by the sign of $\tilde{Y}$ after an offset π, which can be expressed as $\hat{Y} = \text{sgn}(\tilde{Y} - \pi)$. In multi-class classification, where the target Y is represented by a one-hot vector with length K (the number of categories), the final estimate $\hat{Y}$ is obtained by selecting the index of the maximum value in $\tilde{Y}$, i.e., $\hat{Y} = \max_{i \in \{1,\dots,K\}} \tilde{Y}_{[i]}$, where $\tilde{Y}_{[i]}$ represents the i-th element in $\tilde{Y}$.

Evidently, the formulation of w_i has a direct impact on the predictive performance, and its establishment has undergone extensive development. To provide a comprehensive overview of the relevant methods, we also include those approaches that are typically not considered part of weighted average estimation. However, we argue that these methods can still be expressed in the form of a weighted average formula.

Linear model and its spline variants. Linear model Montgomery et al. [2021] is a venerable tool in the field of statistical learning. Let $\boldsymbol{X}$ denote the design matrix, with the i-th row being the transpose of $\vec{X}_i = (1, X_i^\top)^\top$. Then, the estimate of outcome for a new observation X is calculated as $\vec{X}^\top(\boldsymbol{X}^\top\boldsymbol{X})^{-1}\boldsymbol{X}^\top Y$, and can be expressed using formula (1) by letting

$$w_i = \frac{1}{n}\vec{X}^\top(\boldsymbol{X}^\top\boldsymbol{X})^{-1}\vec{X}_i.$$

To introduce more nonlinear features, spline variants seek to project the raw predictor variables onto a larger space using spline basis or piecewise polynomial functions Marsh and Cormier [2001], Sangalli et al. [2013].

Nadaraya-Watson estimation. Nadaraya-Watson estimation Cai [2001], Demir and Toktamiş [2010] is a nonparametric kernel regression technique in the classical statistic inference. The estimator employs a weighted average of the observed outcomes, where the weight is proportional to the distance between the observed predictor and the new predictor of interest. Specifically, for a pre-defined kernel function $\mathcal{K}(\cdot)$, the weight for the i-th observation is given by

$$w_i = \frac{\mathcal{K}(\|X_i - X\|/h)}{\sum_{j=1}^n \mathcal{K}(\|X_j - X\|/h)}.$$

Here, $h > 0$ is the bandwidth parameter. The kernel function $\mathcal{K}$ is used for spreading the weight across neighboring observations, and the optimal bandwidth parameter is typically selected through cross-validation.

***K*-nearest neighbors estimation.** K-nearest neighbors (KNN) algorithm Laaksonen and Oja [1996], Kramer and Kramer [2013] is a nonparametric machine learning technique utilized for both regression and classification tasks. The approach involves predicting the value of a new data point by aggregating its k-nearest neighbors within the training set. To be more specific, let $\mathscr{N}_k$ represent the k-nearest neighbors of X, and the weight w_i is defined as

$$w_i = \frac{1}{k}\mathbf{1}(X_i \in \mathscr{N}_k).$$

The choice of k value is crucial to the performance.

Kernel ridge regression in RKHS. Kernel ridge regression model in RKHS framework An et al. [2007], Vovk [2013] aims to project the predictor variables onto a high-dimensional or even infinite-dimensional Hilbert space. This is achieved through the utilization of the kernel trick. Denote the design matrix as $\boldsymbol{X}$, the kernel matrix generated by $\boldsymbol{X}$ as $K(\boldsymbol{X}, \boldsymbol{X}) \in \mathbb{R}^{n\times n}$, and the kernel matrix generated by $\boldsymbol{X}$ and X as $K(X, \boldsymbol{X}) \in \mathbb{R}^n$. It then follows that the weight w_i is the i-th component of $K(\boldsymbol{X}, \boldsymbol{X})[K(\boldsymbol{X}, \boldsymbol{X})^2 + \lambda K(\boldsymbol{X}, \boldsymbol{X})]^{-1}K(X, \boldsymbol{X})$, where λ is a regularization parameter.

Decision tree. Decision tree Safavian and Landgrebe [1991], Kotsiantis [2013] is a nonparametric method that learns a hierarchy of if-then rules that classify or predict a response variable based on a set of input variables. The decision tree algorithm builds a tree-like model that recursively partitions the data into subsets based on the values of the input variables, with each partition induced by a decision rule such as entropy elimination. Therefore, the partitions are completely data-driven and can not be predetermined. Assume a decision tree has been constructed and that the new observation X is situated within a specific leaf $\mathcal{L}(X)$, which contains some training data. The weight w_i for each training data point X_i in regression tasks is defined as

$$w_i = \frac{1}{|\mathcal{L}(X)|}\mathbf{1}(X_i \in \mathcal{L}(X)),$$

where $|\mathcal{L}(X)|$ represents the number of training data points within the leaf $\mathcal{L}(X)$. The primary difference between decision trees and the aforementioned methods lies in the process of weight generation. Decision trees use adaptive weights learned from both predictors and outcomes, while the other methods mentioned use explicitly calculated weights without observing the outcomes.

Random forest. Random forest Breiman [2001], Biau and Scornet [2016], Speiser et al. [2019] is an ensemble method where a large number of decision trees are constructed and combined to produce a robust and accurate model. Each decision tree is built on a randomly selected subset of the training data and features, which leads to the creation of diverse base models. During the prediction phase, the outputs of these trees are averaged, and the final prediction is obtained by voting or averaging the individual tree outputs. Specifically, assuming that a set of k distinct decision trees have been constructed, and given the new observation X, it is classified into the leaf node $\mathcal{L}_j(X)$ for the j-th tree. Utilizing this information, we arrive at the following expression

$$w_i = \frac{1}{k}\sum_{j=1}^{k}\frac{1}{|\mathcal{L}_j(X)|}\mathbf{1}(X_i \in \mathcal{L}_j(X)),$$

in the context of regression.

XGBoost and LightGBM. XGBoost Chen and Guestrin [2016] and LightGBM Ke et al. [2017] are two state-of-the-art gradient boosting frameworks for statistical learning applications. Gradient boosting is an ensemble approach where a set of weak learners, typically decision trees, are trained sequentially, with each new learner attempting to correct the mistakes of the previous learner. In a nutshell, given k sequentially-generated decision tree regressors, denoted as $f_1, \cdots, f_k$, the predicted output $\hat{Y}$ can be expressed as the sum of the outputs of these regressors for the given input X, i.e., $\hat{Y} = \sum_{j=1}^{k} f_j(X)$. To be more specific, if X falls into the leaf node $\mathcal{L}_j(X)$ of the j-th tree, we define $\omega_j(X)$ as the normalized vector of the indicator function of X belonging to $\mathcal{L}_j(X)$ for all features, i.e.,

$$\omega_j(X) = \frac{1}{|\mathcal{L}_j(X)|}\left(\mathbf{1}(X_1 \in \mathcal{L}_j(X)), \ldots, \mathbf{1}(X_n \in \mathcal{L}_j(X))\right)^\top,$$

and we also define $\omega_j(\boldsymbol{X}) = (\omega_j(X_1), \cdots, \omega_j(X_n))^\top$. By recursively letting $\tilde{\omega}_1(X) = \omega_1(X), \tilde{\omega}_1(\boldsymbol{X}) = \omega_1(\boldsymbol{X})$ and

$$\begin{aligned}\tilde{\omega}_j(X) &= \left[I_n - \sum_{\ell=1}^{j-1}\tilde{\omega}_\ell(\boldsymbol{X})\right]^\top \omega_j(X),\\ \tilde{\omega}_j(\boldsymbol{X}) &= \omega_j(\boldsymbol{X})\left[I_n - \sum_{\ell=1}^{j-1}\tilde{\omega}_\ell(\boldsymbol{X})\right], 2 \le j \le k,\end{aligned}$$

we obtain that w_i is the i-th component of $\sum_{j=1}^{k}\tilde{\omega}_j(X)$. Here, $\tilde{\omega}_j(X)$ is computed using the previously defined weight vectors and the identity matrix I_n.

In summary, as the development of statistical learning continues to evolve, we observe a growing trend towards the utilization of increasingly complex and abstract weight establishment methodologies. Furthermore, the aforementioned methods can be categorized into two subsets, as presented in Table 1. The first subset consists of methods that generate weights through closed forms, potentially with tuned hyperparameters. The second subset includes methods that generate weights through implicit forms that require learning from data.

Table 1: The categorization of related works according to the technique of weight generation.

Method	Closed Form	Implicit Form
Linear model	✓	
Nadaraya-Watson estimation	✓	
K-nearest neighbors estimation	✓	
Kernel ridge regression in RKHS	✓	
Decision tree		✓
Random forest		✓
XGBoost and LightGBM		✓

To further enhance the flexibility of the weight generation process, we propose the integration of neural networks into the framework. This integration is expected to yield superior performance through the introduction of nonlinearity and the ability to incorporate a larger scope of intricate features. Ultimately, our objective is to push the limits of statistical learning to the most flexible level possible, paving the way for more efficient and effective modeling techniques.

3 Method

3.1 Notations

To ensure clarity and comprehensibility, we would like to introduce a series of fundamental notations that underpin our proposed methodology.

Vetors and norms. In this paper, nonrandom vectors shall be represented in lowercase letters, whereas random vectors shall be denoted in uppercase letters. The L_r norm of a vector x shall be expressed as $\|x\|_r$, while the supremum norm of x shall be denoted as $\|x\|_\infty$. For a function f that maps from Ω to $\mathbb{R}$, the supremum norm of f shall be defined as $\|f\|_\infty := \sup_{x\in\Omega} |f(x)|$, and its L_r norm shall be defined as $\|f\|_{L_r(\mathbb{P}_X)} := (\mathbb{E}|f(X)|^r)^{1/r}$ with respect to the distribution of X, which is denoted as $\mathbb{P}_X$.

Modulus of continuity. The definition of the modulus of a function $f : [0,1]^d \to \mathbb{R}$ is given by the expression

$$\omega_f(r) := \sup\left\{|f(x) - f(y)| : x, y \in [0,1]^d, \|x-y\|_2 \le r\right\},$$

for any $r \ge 0$. By varying the form of $\omega_f(r)$, different equicontinuous families of functions can be characterized. For example, the modulus $\omega_f(r) = \gamma r$ represents γ-Lipschitz continuity for $\gamma > 0$, while the modulus $\omega_f(r) = \lambda r^\alpha$ for $\lambda, \alpha > 0$ characterizes Hölder continuity. These forms of modulus are of particular interest in the study of function spaces and their properties.

Function family. The function family is represented in mathematical calligraphy style, denoted by symbols such as $\mathcal{F}, \mathcal{G}, \mathcal{H}$. Specifically, the Hölder smooth function family is denoted as $\mathcal{H}^{\lambda,\alpha}([0,1]^d)$ for parameters $\lambda, \alpha > 0$, with its constituent element f defined on the unit hypercube $[0,1]^d$ and a modulus $\omega_f(r) = \lambda r^\alpha$, such that for all $f \in \mathcal{H}^{\lambda,\alpha}([0,1]^d)$, $|f(x) - f(y)| \le \lambda \|x-y\|_2^\alpha$.

3.2 Adank

Our research was inspired by the observation that equation (1) incorporates a kernel w_i that plays an important role in accurately measuring the relationship between two data points. As determining the optimal kernel function is often challenging in practice, we propose the use of a neural network function to represent the kernel function. This enables us to leverage the learning capabilities of the neural network to discover the optimal parameters associated with the kernel function, and thereby improve the accuracy of predictions. By incorporating neural network-based functions in this way, we aim to enhance the overall performance and applicability of our methodology. Reflecting the unique features and innovations underlying our proposed methodology, we have given this approach a distinctive name, Adank, short for adaptive neural kernels.

We commence with the technique of sample splitting. Given a labeled sample set $\mathscr{D}_n = ((X_1, Y_1), \cdots, (X_n, Y_n))$, we partition it into two subsets, specifically a training subset and an evaluating subset. The purpose of the training subset is to learn a kernel function f, while the evaluating subset is utilized to generate the final estimate. The index of training subset is then referred to as $\mathscr{I}_1 \subset \{1, \cdots, n\}$ with a size of n_1, and the index of evaluating subset is similarly referred to as $\mathscr{I}_2 = \{1, \cdots, n\} \backslash \mathscr{I}_1$ with a size of n_2. When obtaining a desired kernel function f, for a new observation $X \in \Omega$, we construct the (potential) final prediction as

$$\hat{Y} = \frac{1}{n_2} \sum_{i \in \mathscr{I}_2} f(X_i, X) Y_i. \tag{2}$$

Here, the function $f : \Omega \times \Omega \to \mathbb{R}$ is selected from a prespecified family of neural network functions, denoted by $\mathcal{F}$. In practical applications, the choice of neural network architecture is determined by the specifics of the input data. Multilayer perceptrons and convolutional neural networks are commonly employed architectures that have proven effective in diverse contextsHornik [1993], Shen et al. [2021a], Gu et al. [2018]. The selection and configuration of these networks is guided by factors such as the complexity of the data, the presence of spatiotemporal or structural patterns, and the desired level of interpretability, among other considerations.

In order to acquire the most optimal kernel, a suitable loss function, denoted as ϕ, is designated to ensure that the empirically desired function, represented by

$$\hat{f} = \underset{f \in \mathcal{F}}{\operatorname{argmin}} \frac{1}{n_1} \sum_{i \in \mathscr{I}_1} \phi\left(Y_i, \hat{Y}_i\right), \tag{3}$$

is attained, where

$$\hat{Y}_i = \frac{1}{n_1} \sum_{j \in \mathscr{I}_1} f(X_j, X_i) Y_j.$$

In regression tasks, the loss function ϕ is defined as the least squares loss, which can be expressed as

$$\hat{f} = \underset{f \in \mathcal{F}}{\operatorname{argmin}} \frac{1}{n_1} \sum_{i \in \mathscr{I}_1} \left(Y_i - \hat{Y}_i \right)^2 . \tag{4}$$

In the cases of classification tasks, when the outcome Y is a binary variable within $\{-1, 1\}$, a candidate loss function can be the least squares loss, the hinge loss, the exponential loss, or the logistic loss. Furthermore, in order to establish a unified framework for classification tasks, it is recommended to transform Y into a one-hot vector and then apply the cross entropy loss. For example, in a K-categorical classification task, the corresponding Y belongs to $\{e_1, \cdots, e_K\}$, where e_i is the i-th one-hot vector of length K. The optimization paradigm can then be written as

$$\hat{f} = \underset{f \in \mathcal{F}}{\operatorname{argmin}} -\frac{1}{n_1} \sum_{i \in \mathscr{I}_1} \sum_{j=1}^{K} \mathbf{1}(Y_{ij} = 1) \log \frac{\exp(\hat{Y}_{ij})}{\sum_{\ell=1}^{K} \exp(\hat{Y}_{i\ell})}, \tag{5}$$

where Y_{ij} indicates the j-th component in vector Y_i. The final prediction here should be the index of largest value in $\hat{Y}$.

3.3 Ensemble Model

Taking into account all previous research aforementioned in Section 2, Adank is quite similar to a decision tree. This similarity affords Adank suitability as a base learner for more complicated prediction algorithms. By employing external bagging approach to build a more hierarchical variant of our model, it is of possibility to further augment performance. Specifically, this bagging approach involved initializing several instances of our model, with randomly sampling observations and features fed into each instance, and producing the final output by averaging the results of each individual instance. In a regression task with input feature $X \in \mathbb{R}^d$, let us consider the establishment of s instances of our model. Initially, we generate s random subsets of $\{1, \ldots, n\}$, denoted as $\{\mathscr{T}_1^n, \ldots, \mathscr{T}_s^n\}$, as well as s random subsets of $\{1, \ldots, d\}$, denoted as $\{\mathscr{T}_1^d, \ldots, \mathscr{T}_s^d\}$. Then, for the i-th instance, we input the sample $\mathscr{T}_i^n$ with each feature reserved to its $\mathscr{T}_i^d$ dimensions, resulting in an output $\hat{Y}^i$. Consequently, the final estimate is expressed as $\hat{Y}_{\text{Ens}} = \sum_{i=1}^{s} \hat{Y}^i / s$. In Section 4 of our paper, we will provide a illustration of how our methodology can be deployed with a similar procedure to random forest algorithms.

3.4 Non-Euclidean Tasks

In contemporary times, there has been a surge in interest towards non-Euclidean statistical analysis, predominantly driven by the exigencies of modern-day applications. These applications include but are not limited to the covariance or correlation matrices for functional brain connectivity in neuroscience, as well as the probability distributions in CT hematoma density data. Such heightened attention towards non-Euclidean statistical analysis is indicative of its emerging significance in the current academic and scientific landscape. Relevant works can be found in Hein [2009], Bronstein et al. [2017], Petersen and Müller [2019], Chen and Müller [2022], Lin et al. [2023].

Here, we propose that Adank can be expanded to non-Euclidean tasks given that a sensible weighted average representation can be devised.

For illustration, let us consider a situation in which the target Y is a symmetric positive-definite (SPD) matrix, while the predictor X remains in Euclidean space. In this scenario, the range of Y, denoted as $\mathcal{Y}$, does not simply form a trivial subset of the Euclidean space, but rather represents a property that every element M in $\mathcal{Y}$ satisfies the condition that $\alpha^\top M \alpha > 0$ for any vector α consisting of real numbers. Therefore, an appropriate approach to addressing this type of regression problem should comprehensively leverage the characteristic of $\mathcal{Y}$, which meets the strength of the method we propose.

When applying our method to SPD regression tasks, in order to enhance the suitability of the SPD matrix space, we endeavored to design a loss function that utilizes information from the SPD manifold. Specifically, For two SPD matrices U_1 and U_2, their Jeffrey divergence Harandi et al. [2014] is defined as

$$d_J^2(U_1, U_2) = \frac{1}{2} \text{Tr}(U_1^{-1} U_2) + \frac{1}{2} \text{Tr}(U_2^{-1} U_1) - m \ ,$$

where Tr represents the trace operator of a square matrix and m represents the row number of U_1. The Jeffrey divergence actually denotes a pseudodistance between two SPD matrices. Then, our loss function incorporating the Jeffrey divergence is given by

$$\mathscr{L}_J(f) = \frac{1}{n} \sum_{i=1}^{n} d_J^2(Y_i, \hat{Y}_i) \ .$$

Here, $\hat{Y}_i$ is the estimate of Y, i.e.,

$$\hat{Y}_i = \sum_{j=1}^{n} \frac{f(X_j, X_i)}{\sum_{l=1}^{n} f(X_l, X_i)} Y_j \ ,$$

with the constraint that f should be positive to ensure the positive-definiteness of $\hat{Y}_i$. Additionally, empirical evidence suggests that normalizing weights in SPD regression tasks can lead to performance improvements. In a similar manner, we begin by minimizing the function $\mathscr{L}_J(f)$ with respect to f to obtain a suitable kernel function $\hat{f}$, followed by the construction of weighted average estimates using $\hat{f}$ and normalized weights in the evaluation phase. Similar technique can be utilized in the case where the target Y is a graph laplacian, with nonnegative f to construct suitable weights. Please refer to the Supplementary material for more details. Several interconnected examples are discussed upon in Section 4.

3.5 Optimization

A significant challenge associated with neural network-based methods is their potential for intensive computation. When working with large datasets, the computational burden of performing the calculation for each estimate $\hat{Y}_i$ can be overwhelming. Evidently, in our model, it is necessary to execute neural network evaluations n times in order to generate a solitary representation, a process that can become exceedingly computationally demanding as the sample size n increases. Accordingly, we devised a query-based algorithm to mitigate the computational burden.

Precisely, in the phase of training, within each iteration, a random selection of n_r observations is designated as reference samples, while another set of n_q observations are selected as query samples. The calculation of the estimation of each query point is performed using the reference samples in the formula of equation (2), which is subsequently denoted as $\hat{Y}^q$. The objective function, as expressed in formula (3), is established by utilizing the true target values in the query set in conjunction with the predicted values of $\hat{Y}^q$. We provide detailed descriptions of the algorithmic procedure in Algorithm 1 and Algorithm 2 in Supplementary material.

The query-based algorithm is a modification of mini-batch paradigm. Due to the potential calculation load brought by our representational formula (2), the engaged observations for weighted average are supposed to be reduced to guarantee computational efficiency. As a result, we actually make predictions for a few selected observations within a single iterative. Additionally, the sampling approach ensures that reference set and query set are (approximately) independent, thereby mimicing the prediction procedure and improving generalization capacity. It is important to note that, in the prediction phase, we also employ a potentially small subset of the whole sample to perform weighted average predictions, which represents a tradeoff between the expenses incurred in computation and the level of precision achieved. The size (n_2) of the evaluation subset is shown to be significantly smaller than the size (n_1) of the training subset, as demonstrated in Section 5. Specifically, $n_2/n_1 \to 0$ as $n \to \infty$.

3.6 Discussion

In our methodology, we refrain from imposing any specific constraints on the function class $\mathcal{F}$, allowing for maximum flexibility and adaptability across differing tasks. Nonetheless, we recognize the potential benefit of some constraints in certain applications, particularly where auxiliary information is available. On the other hand, the inclusion of symmetry affords considerable convenience in proofs, as demonstrated in Section 5. Interestingly, our approach diverges from the traditional perspective in that we highlight the potential for a kernel function not to be nonnegative, and demonstrate, in Section 5, that this fact does not necessarily undermine the effectiveness of our model.

An essential aspect of our methodology is understanding what is learned in the training process. This question naturally arises as we seek to gain insights into the effectiveness and generalizability of our model. While we recognize the importance of in-depth technical details and interpretations, we limit ourselves to providing a general overview of our approach in this section, without delving into the specific technicalities. As such, we defer more detailed technical discussions and explicit interpretations to Section 5, where we provide a thorough exposition of the technical details of our method and elaborate on the key findings and insights derived from our approach.

4 Experiments

To objectively and comprehensively evaluate the performance of our proposed methodology, we present a series of simulation and real data results in this section. In doing so, we aim to benchmark our approach against several state-of-the-art techniques, including random forest (RF), XGBoost, LightGBM, and deep neural network models (DNN). Additionally, to explore the impact of bagging algorithms on our methodology, we implemented a version of our model that incorporates bagging, denoted as Adank-Ens. Through this rigorous and comprehensive evaluation process, we aim to gain deeper insights into the performance of our methodology across varying data types, contexts, and modeling challenges.

4.1 Simulation

4.1.1 Euclidean Regression

In this part, we evaluated the efficacy of our proposed methodology for Euclidean regression tasks using four distinct artificial datasets, each representing intricate data structures. The datasets were generated to reflect different scenarios involving nonlinearity, sparsity, and local structure. To model these datasets, we adopted an additive approach comprising a conditional mean and Gaussian noise, and we varied the strength of noise across different signal-to-noise ratios (SNRs). Specifically, we employed low, moderate and high SNRs corresponding to SNR values of 1, 5 and 10, respectively. Lower SNR indicates a weaker signal-to-noise ratio, thus providing a more challenging modeling task. We opted to employ the mean squared error (MSE) as the primary evaluation metric.

For a d-dimensional predictor X, the i-th entry of X is denoted as $X_{[i]}$, i.e., $X = (X_{[1]}, \cdots, X_{[d]})^\top$. The models utilized to generate artificial datasets are as follows.

(A) $Y = 3\sin(\pi(X_{[1]} - X_{[2]})) \cdot \mathbf{1}(X_{[1]} > X_{[3]}) + \sigma\epsilon$;

(B) $Y = \sum_{i=1}^{d-1} \exp(X_{[i]} - X_{[i+1]}) \log \frac{X_{[i+1]}}{X_{[j]}} + (1 + X_{[j]})^2 + \sigma\epsilon$;

(C) $Y = \frac{(X_{[1]}+2X_{[2]}X_{[4]})^3}{5(1+\exp(X_{[3]}))} \cdot \mathbf{1}(5X_{[1]}^3 > X_{[2]} - X_{[3]}) + \sigma\epsilon$;

(D) $Y = \max(3x_{[1]}^2 - \sin(\pi(x_{[2]} + x_{[3]})), e^{x_{[4]}+x_{[5]}} - x_{[6]}) + \sigma\epsilon$;

where X is distributed uniformly over the interval $[0, 1]^d$, ϵ is normally distributed with mean 0 and variance 1, and σ is defined as the square root of the variance of $\mathbb{E}(Y|X)$ divided by the SNR. For each model setup, we generated n artificial observations for training, and evaluated MSE on the testing dataset, comprising of 1000 independent sampels. Furthermore, the hyperparameters for RF, XGBoost, LightGBM, KNN and DNN were tuned through a validation process. Each setting was replicated in 100 times. Please refer to the Supplementary material for details.

Table 2: The results of averaged mean squared errors generated by our methods, in comparison with other methods. The standard deviations are presented in brackets. The top 2 smallest scores are shown in bold text. Each setting was replicated a total of 100 times.

	(n, p, SNR)	Adank	Adank-Ens	RF	XGBoost	LightGBM	KNN	DNN
(A)	(200, 20, 10)	**0.496 (0.099)**	**0.477 (0.078)**	0.684 (0.103)	0.903 (0.076)	0.733 (0.079)	1.545 (0.080)	0.943 (0.159)
	(200, 20, 5)	**0.551 (0.118)**	**0.509 (0.091)**	0.704 (0.094)	0.636 (0.089)	0.730 (0.076)	1.542 (0.085)	0.980 (0.162)
	(200, 20, 1)	1.883 (0.328)	1.431 (0.217)	**1.087 (0.140)**	**1.237 (0.135)**	1.131 (0.141)	1.739 (0.106)	2.584 (0.304)
	(500, 30, 5)	0.477 (0.122)	**0.391 (0.054)**	0.440 (0.0460)	**0.402 (0.045)**	0.407 (0.036)	1.582 (0.082)	0.560 (0.159)
(B)	(200, 20, 10)	0.047 (0.008)	**0.039 (0.003)**	0.062 (0.005)	**0.044 (0.004)**	0.045 (0.004)	0.068 (0.006)	0.051 (0.013)
	(200, 20, 5)	0.049 (0.008)	**0.040 (0.003)**	0.062 (0.005)	0.046 (0.004)	0.047 (0.004)	0.069 (0.005)	**0.044 (0.006)**
	(200, 20, 1)	0.102 (0.023)	0.074 (0.010)	**0.068 (0.006)**	0.070 (0.006)	**0.064 (0.006)**	0.082 (0.008)	0.128 (0.052)
	(500, 30, 5)	0.032 (0.006)	**0.025 (0.002)**	0.042 (0.002)	**0.027 (0.002)**	0.029 (0.002)	0.049 (0.003)	0.028 (0.004)
(C)	(200, 20, 10)	**0.013 (0.006)**	**0.012 (0.004)**	0.015 (0.004)	0.015 (0.005)	0.015 (0.003)	0.030 (0.004)	0.014 (0.007)
	(200, 20, 5)	0.015 (0.009)	**0.012 (0.003)**	0.018 (0.005)	0.019 (0.003)	0.016 (0.003)	0.031 (0.004)	**0.012 (0.003)**
	(200, 20, 1)	0.033 (0.010)	**0.024 (0.005)**	**0.023 (0.006)**	0.036 (0.010)	**0.024 (0.005)**	0.036 (0.005)	0.048 (0.011)
	(500, 30, 5)	0.014 (0.009)	0.011 (0.003)	**0.008 (0.002)**	**0.008 (0.002)**	0.009 (0.002)	0.033 (0.004)	0.011 (0.003)
(D)	(200, 20, 10)	**0.106 (0.044)**	**0.083 (0.018)**	0.231 (0.028)	0.161 (0.016)	0.156 (0.017)	0.727 (0.065)	0.126 (0.056)
	(200, 20, 5)	**0.132 (0.039)**	**0.103 (0.018)**	0.202 (0.025)	0.182 (0.019)	0.168 (0.020)	0.747 (0.064)	0.153 (0.033)
	(200, 20, 1)	1.038 (0.178)	0.654 (0.099)	**0.402 (0.063)**	0.428 (0.077)	**0.372 (0.061)**	0.858 (0.088)	1.198 (0.200)
	(500, 30, 5)	0.125 (0.066)	**0.072 (0.011)**	0.144 (0.014)	0.127 (0.012)	**0.086 (0.010)**	0.750 (0.048)	0.129 (0.118)

The results are presented in Table 2 and Figure 1. Remarkably, our methodologies exhibited superior performance compared to RF, XGBoost, LighGBM, KNN, and DNN in the majority of cases that features limited sample sizes and moderate to high SNRs. Additionally, it is evident that our methodologies are particularly applicable for cases in which the conditional mean functions admit local manifold characteristics as shown in setting (A) and (D). This highlights the advantage of employing the weighted average technique, compared to global approaches such as DNN. Furthermore, the ensemble technique holds great potential in offering more precise predictions while simultaneously eliminating variation. In situations characterized by extremely low SNRs, our methods remain a viable option.

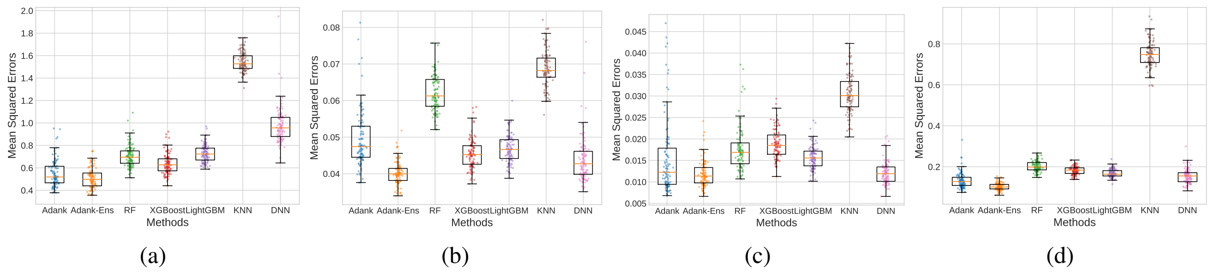


Figure 1: The boxplots of recorded mean squared errors generated by all the methods in Euclidean regression tasks, where (a) setting A with $n = 200, p = 20$, SNR=5, (b) setting B with $n = 200, p = 20$, SNR=5, (c) setting C with $n = 200, p = 20$, SNR=5, and (d) setting D with $n = 200, p = 20$, SNR=5.

4.1.2 SPD Regression

We conducted a further investigation into the performance of our method in the context of SPD targets. In order to establish a reliable representation, we incorporated a batch normalization layer and a Sigmoid activation layer into the neural network to ensure that the kernel, denoted as f in equation (2), is strictly positive. Furthermore, we normalized the kernels $f(X_j, X_i)$ with respect to j to ensure that they added up to one.

To generate datasets with SPD targets, we set the predictor variables $X \in U([0,1]^d)$ and $\log Y \sim N_{mm}(M(X), \sigma^2)$. In more detail, if a matrix A follows $N_{mm}(M, \sigma^2)$, A can be expressed as $\sigma^2 Z + M$, where Z is a random matrix with independent $N(0,1)$ diagonal elements and $N(0,1/2)$ off-diagonal elements, and both M and Z are symmetric Zhang et al. [2021]. We set $\sigma^2 = 0.2$ and varied $M(X)$ among cases as follows.

(E) $\rho_1(X) = 0.5\cos(\pi(X_{[1]} - X_{[2]} + 2X_{[3]}))$, $\rho_2(X) = 0.5\sin(\pi(2X_{[1]} - e^{X_{[2]}}))$, and

$$M(X) = \begin{pmatrix} 1 & \rho_1(X) & \rho_2(X) \\ \rho_1(X) & 1 & \rho_1(X) \\ \rho_2(X) & \rho_1(X) & 1 \end{pmatrix} \; ;$$

(F) $M(X) = (m_{ij}(X))_{5\times5}$ where $m_{ij}(X) = 0.8^{|i-j|\rho(X)}$ and $\rho(X) = 5/\|X\|_2^2$.

In our study, the Log-Cholesky metric d_L was employed to assess the precision of predictions Lin [2019]. The adoption of this metric is deemed more appropriate as it is able to reflect the Riemannian geometry of SPD matrices. We implemented the loss function utilizing Jeffrey divergence Harandi et al. [2014], as illustrated in Supplementary material. As a point of reference, we also evaluated the Nadaraya-Waston estimation (NW) with kernel $\mathcal{K}(x, x') = \exp(-\|x - x'\|_2^2/\gamma^2)$ where γ is a tunable parameter and the Fréchet mean (FM) of all training targets with Log-Cholesky metric Petersen and Müller [2019], which completely disregarded the information contained in X. Various training sample sizes n and the dimension p of X were considered. To evaluate the effectiveness of the models, we generated a testing set of 100 independent observations. Specifically, for each testing target Y_i and its corresponding prediction $\hat{Y}_i$ ($i = 1, \cdots, 100$), the final evaluation metric was defined as follows

$$\bar{d}_L^2 = \frac{1}{100}\sum_{i=1}^{100} d_L^2(Y_i, \hat{Y}_i).$$

Table 3: The results of averaged Log-Cholesky metric generated by our methods, in comparison with Nadaraya-Waston estimation (NW) and Fréchet mean (FM). The standard deviations are presented in brackets. The best scores are shown in bold text. Each setting was replicated a total of 100 times.

	(n, p)	Adank	NW	FM
	(200, 5)	**0.291(0.058)**	0.659(0.039)	0.958(0.029)
(E)	(500, 10)	**0.297(0.085)**	0.815(0.035)	0.963(0.030)
	(1000, 20)	**0.494(0.092)**	0.927(0.041)	0.957(0.031)
	(200, 5)	**0.195(0.038)**	0.339(0.029)	0.834(0.051)
(F)	(500, 10)	**0.227(0.056)**	0.441(0.027)	0.814(0.064)
	(1000, 20)	**0.263(0.040)**	0.396(0.026)	0.533(0.044)

It is evident that Adank outperformed both the Nadaraya-Watson estimation and direct averaging techniques (see Table 3), thus demonstrating its efficacy in the precise forecasting of SPD matrices. This outcome lends credence to the viability of our method in non-Euclidean target tasks, and its potential extension to a myriad of non-Euclidean applications.

4.2 Real Data

4.2.1 Image Classification

As a novel and promising tool for supervised learning, Adank has the potential to address a wide range of real-world problems. To test the applicability of our approach to classification problems, we conducted experiments on two well-known datasets, namely, the MNIST and FashionMNIST datasets Xiao et al. [2017]. To adapt our methodology to image inputs, we employed convolutional neural networks (CNNs), which are widely recognized for their effectiveness in image-based classification tasks. Specifically, to measure the association between two input images, we stacked them along with the first channel. As an illustrative example, two $1 \times 28 \times 28$ images were concatenated into a $2 \times 28 \times 28$ tensor to be processed by the CNN. This approach represents one of the feasible ways to implement a function that handles two inputs, and its adaptability could be further enhanced through various modifications.

To commence, we selected five distinct subsets from the MNIST dataset, each comprising of images depicting digits 0 and 6, 1 and 7, 3 and 5, 4 and 9, 7 and 9, respectively. The pairs of digits included in the subsets were similar in terms of recognition. We conducted a comparative analysis of four types of loss functions, namely logistic loss, least squares loss, hinge loss, and exponential loss, with the current state-of-the-art benchmark of a conventional CNN method. It is quite evident that the least squares loss function outperformed other three types of loss functions in terms of its exceptional accuracy and robustness (see Table 4). The CNN method has exhibited remarkable performance, and Adank using the least squares loss function has fallen only marginally behind the CNN method with the absolute error less than 3‰. In section 5, we shall delve deeper into the theoretical underpinnings of utilizing Adank for classification with the least squares loss function. In contrast, we do not recommend using the exponential loss function in practical applications. The results obtained through the utilization of logistic loss were satisfactory.

Table 4: The averaged accuracy scores achieved by Adank. We compared Adank models implemented through logistic loss and least squares loss, among which the highest accuracy scores are emphasized in bold text, with the benchmark of vanilla CNN outputs. The standard deviations are presented in brackets.

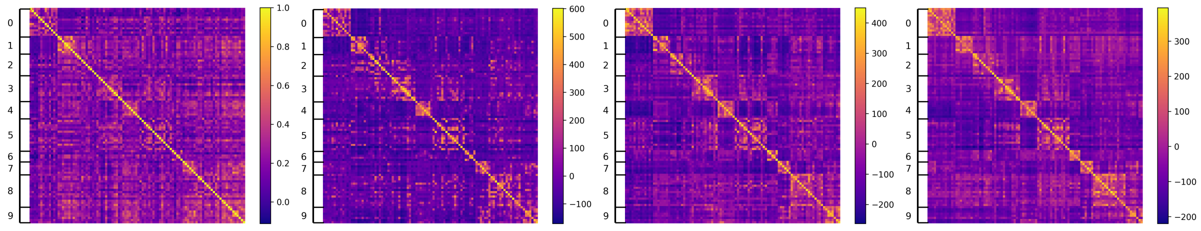
	Adank		
	Logistic	Least Squares	CNN
0-6	0.988(0.004)	**0.994(0.002)**	0.996(0.002)
1-7	0.989(0.004)	**0.996(0.003)**	0.997(0.002)
3-5	0.987(0.006)	**0.994(0.004)**	0.995(0.003)
4-9	0.987(0.006)	**0.992(0.005)**	0.995(0.003)
7-9	0.988(0.007)	**0.994(0.006)**	0.996(0.002)

Following this, our attention shifted towards multiclass classification tasks utilizing the MNIST and FashionMNIST datasets. Recall that when dealing with multiclass classification, we implement cross entropy loss within Adank. Moreover, to compare the performance of Adank against other classical neural networks used in multiclass classification tasks, we tested it against various architectures, including ordinary CNN, LeNet, ResNet, and DenseNet. Additionally, to investigate how the performance of our proposed methodology was influenced by the choice of base CNN architecture, we tested our methodology using two different base CNN architectures, an ordinary CNN and ResNet. It is worthy noting that we conducted experiments that focused on randomly selecting 100, 200, and 500 samples from the training set, thus resetting the problem into a smaller context. The results of these experiments provided us with valuable insights into the potential strengths and limitations of our approach under different training data sizes and informed our future direction of research in this domain. Each setting was replicated in 50 times.

The evaluation results of Adank, as compared to various classical CNN architectures, are presented in Table 5. On the MNIST dataset, experimental results show that our proposed approach consistently outperformed other architectures, including the ordinary CNN, LeNet, ResNet, and DenseNet, particularly in cases where sample sizes were limited to 100 and 200. However, our method demonstrated slightly inferior results compared to DenseNet when the sample size was increased to 500. Notably, we observed a significant accuracy improvement in our approach using an ordinary CNN architecture compared to that achieved by the ordinary CNN itself. With regard to the FashionMNIST dataset, we observed that our approach exhibited an excellent performance across all architectures in use, particularly for small sample sizes of 100 and 200. However, as the sample size increased to 500 in FashionMNIST, while our approach continued to outperform LeNet, it showed a relatively moderate performance. These results suggest that Adank is particularly applicable in small sample size scenarios, and it is advisable to experiment with various foundational neural network frameworks to achieve a more optimal performance.

Table 5: The averaged accuracy scores achieved by Adank and various CNN architectures. Adank-C indicates the model using ordinary CNN architectural, and Adank-R indicates the variant using ResNet. The highest accuracy scores are emphasized in bold text. All of the standard deviations exhibited a comparable magnitude, without exceeding the threshold of 0.067.

	Sample Size	CNN	LeNet	ResNet	DenseNet	Adank-C	Adank-R
	100	0.734	0.785	0.738	0.746	0.803	**0.815**
MNIST	200	0.832	0.839	0.851	0.868	0.879	**0.883**
	500	0.915	0.894	0.926	**0.936**	0.930	0.929
	100	0.660	0.656	0.652	0.640	0.687	**0.694**
FashionMNIST	200	0.726	0.707	0.715	0.711	**0.740**	0.728
	500	**0.785**	0.758	0.771	0.769	0.771	0.763



(a) (b) (c) (d)

Figure 2: (supervised/adaptive kernel, unsupervised/deterministic kernel)The heatmaps of kernel evaluation, which were produced by (a) the SSMI metric, and the model trained on (b) 100 samples, (c) 200 samples, (d) 500 samples. Brighter regions indicate larger values.

Despite the classification performance, the neural network kernel did acquire some intrinsic association between images. In the subsequent analysis after classification, one hundred images were randomly selected and their pairwise kernel evaluations were computed. It was observed that ten distinct diagonal blocks emerged as the training sample size increased (refer to Figure 2 (b)-(d)). With a small number of samples, the high-signal region was confined to the immediate vicinity of the diagonal. However, as the sample size grew, ten prominent diagonal blocks emerged, with their area precisely reflecting the number of selected image categories. In addition, we noted a strong relativity between number 3 and number 5, as well as number 7 and number 9, as indicated by the highlighted regions in Figure 2 (d), which is in accordance with their appearance as perceived by humans. As a benchmark, we calculated the structural similarity index measure (SSIM) Wang et al. [2004] for each pair of images (see Figure 2 (a)). The result indicates that the SSIM only provides a rough representation of the differences among different handwritten digit categories, whereas the kernel generated by our supervised method clearly outperforms it. To conclude, our approach furnishes an adaptable metric to measure the similarity between image instances.

4.2.2 Trip Flow Prediction

The Taxi and Limousine Commission of New York City has made available records pertaining to date and time of pick-up and drop-off, location of pick-up and drop-off, distance of trip, fare breakdown, and passenger count as reported by the drivers for yellow taxis. These data can be accessed through `https://www1.nyc.gov/site/tlc/about/tlc-trip-record-data.page`. Our study focuses on the taxi trips that occurred in Manhattan during May of 2022. Specifically, Manhattan was divided into 10 zones and taxi trips between every two zones that occurred per hour were recorded. This resulted in 10×10 symmetric matrices where each entry indicated the taxi communication across two zones. Covariates included 13 features related to trip information, 16 features related to weather information, and 24 hour indicators ranging from 0 o'clock to 23 o'clock. The trip information features included time used, fare amount, and improvement surcharge, among others. Weather information features included temperature, dewpoint, humidity, windspeed, pressure, and precipitation. The total dataset of 744 observations was further divided into a training set comprising the first 520 instances and a testing set comprising the remaining 224 instances. Normalization was applied to the features in the training set and the same transformation was applied to the features in the testing set.

Following the work of Zhou and MÃ¼ller [2022], the 10×10 matrices were converted into their corresponding graph Laplacians. Hence, our target focused on graph Laplacians that exhibited both symmetry and positive semi-definiteness. Adank was utilized to forecast the trip flows per hour. Noticing that the targets were graph Laplacians, the kernel f was restricted to nonnegative values by adding a batch normalization layer and a sigmoid activation layer. It is obvious that Adank was able to accurately forecast the trip flows (see Figure 3 and 4). To measure the effectiveness, we note that the testing loss using Adank was 0.770, while those of Nadaraya-Waston estimation and direct average were 1.316 and 6.398, respectively.

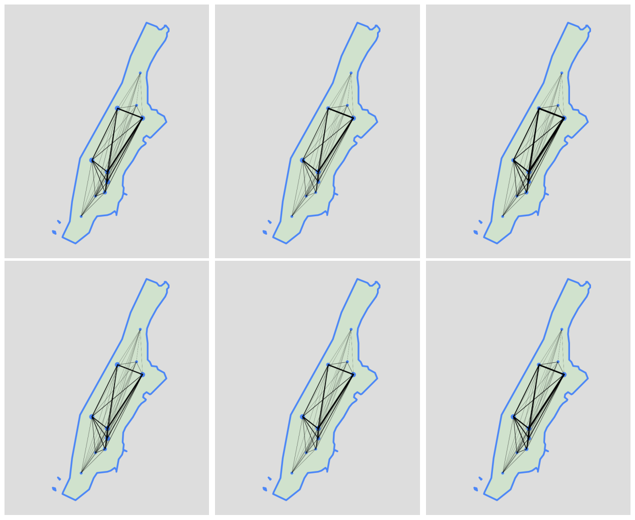


Figure 3: Some examples of true trip flow (top panel) in testing set and the corresponding predicted trip flow using Adank (bottom panel). The width of lines indicates trip frequency.

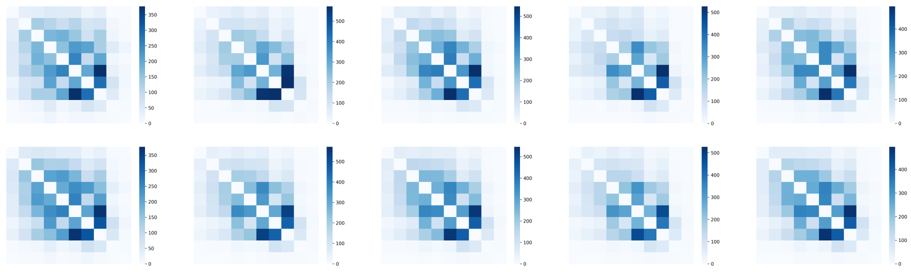


Figure 4: Some examples of true trip flow (top panel) in testing set and the corresponding predicted trip flow using Adank (bottom panel), displayed in heatmaps.

5 Theory

Upon delving into the theoretical investigation of our methodology, the primary obstacle we confronted pertained to the modeling of the association between predictor variables X and response Y. Here, in the context of regression, we contemplate the implementation of a conditional mean model, complemented by a homogeneously variational remainder term, i.e.,

$$Y = h_0(X) + \epsilon, \tag{6}$$

where X is supported on a subset $\mathscr{X} \subset [0,1]^d$ and $\mathbb{E}(\epsilon|X) = 0, \mathrm{Var}(\epsilon|X) = \nu^2$ for a fixed constant ν. In the context of classification, we consider the conditional multinomial model, i.e.,

$$Y|X \sim \mathrm{Multinomial}(K, p(X)), \tag{7}$$

where Y is in the fomula of a one-hot vector of length K and $p(X) = (p_1(X), \cdots, p_K(X))^\top \in [0,1]^K$ with $\sum_{i=1}^K p_i(X) = 1$.

5.1 Neural Network

The versatility of neural networks as a powerful computational tool in machine learning is widely documented, owing to their ability to learn and recognize complex patterns from large datasets Silver et al. [2017], Creswell et al. [2018], Floridi and Chiriatti [2020], Saralioglu and Gungor [2022]. The fundamental idea underlying the functionality of the neural network stems from its universal approximation capability. The ability to approximate any continuous function with arbitrary accuracy has been extensively studied, validated, and celebrated in the scientific community. In particular, a typical deep forward neural network, in the multilayer perceptron (MLP) architecture, is composed of a series of iterative affine transformations and nonlinear activations. Defining $\psi_i^m(x) = W_i x + b_i$ where W_i is commonly referred to as the weight matrix, b_i is the bias term, and $\sigma(x) = \max(x, 0)$ serves as the rectified linear unit activation (ReLU), a function f_{MLP} implemented by a MLP can be expressed as

$$f_{MLP}(x) = \psi_L^m \circ \sigma_{L-1} \circ \psi_{L-1}^m \circ \cdots \sigma_1 \circ \psi_1^m(x).$$

Here, L denotes the depth of the neural network.

In contrast to MLPs, convolutional neural networks (CNNs) are specifically designed to handle tensor inputs, such as images. By utilizing convolutional weights to scan over the inputs, CNNs are able to significantly reduce the number of neurons used. Each convolutional weight, denoted as $C \in \mathbb{R}^{r \times s}$ with r filters and each of dimension s, is tailored to capture specific transform invariant features. Let W_i^c represent the structured weight matrix induced by the convolutional weight C_i, b_i^c be the bias vector, and $\psi^c(x) = W_i^c x + b_i^c$. A fully convolutional neural network (FCNN) can be expressed as

$$f_{FCNN}(x) = \psi_L^c \circ \sigma_{L-1} \circ \psi_{L-1}^c \circ \cdots \sigma_1 \circ \psi_1^c(x),$$

where σ denotes the ReLU activation or the max pooling function. Ultimately, it is important to note that a CNN can be expressed in the form of

$$f_{CNN}(x) = \psi_L \circ \sigma_{L-1} \circ \psi_{L-1} \circ \cdots \sigma_1 \circ \psi_1(x),$$

where ψ can be ψ^m or ψ^c, and σ is the ReLU activation if followed by ϕ^m or can be the ReLU activation or max pooling function if followed by ϕ^c.

In the subsequent analysis, we utilize the function class of convolutional neural networks, denoted by $\mathcal{F}_{CNN} := \mathcal{F}_{L,M,N,B_a,B_u}$, as follows. Let $\mathcal{L}_F$ be a subset of indices $\{1, \cdots, L\}$, and let $\mathcal{L}_C = \{1, \cdots, L\} \backslash \mathcal{L}_F$ be its complement. Let $\Psi = \{\psi_1, \cdots, \psi_L\}$ be a collection of L affine transformations, each with its respective input dimension $\{d_1, \cdots, d_L\}$. Specifically, for $l \in \mathcal{L}_F$, we define $\psi_l(x) = W_l x + b_l$ with fully connected weight $W_l \in \mathbb{R}^{d_l \times d_{l+1}}$ and bias $b_l \in \mathbb{R}^{d_{l+1}}$. For $l \in \mathcal{L}_C$, we define $\psi_l(x) = W_l^c x + b_l^c$ with $W_l^c \in \mathbb{R}^{d_l \times d_{l+1}}$ induced by the convolutional weight $C_l \in \mathbb{R}^{r_i \times s_i}$ and bias $b_l^c \in \mathbb{R}^{d_{l+1}}$. Let m_l denote the number of operations performed by each of the r_l filters in convolutional layer $l \in \mathcal{L}_C$. Let $(a_1, \cdots, a_L)$ be positive numbers that are uniformly bounded by $B_a > 0$, and let $\{\mathcal{A}_1, \cdots, \mathcal{A}_L\}$ be sets with $\mathcal{A}_l = \{\psi_l : \|W_l\|_F + \|b_l\|_F \leq a_l\}$ for $l \in \mathcal{L}_F$ and $\mathcal{A}_l = \{\psi_l : \|C_l\|_F + \|b_l^c\|_F \leq a_l\}$ for $l \in \mathcal{L}_C$, where $\|\cdot\|_F$ denotes the Frobenius norm. Consequently, the function class of CNNs is defined as

$$\begin{aligned} \mathcal{F}_{L,M,N,B_a,B_u} = \{f : \mathscr{X} \to \mathbb{R} : f = \psi_L \circ \sigma_{L-1} \circ \psi_{L-1} \circ \cdots \circ \\ \sigma_1 \circ \psi_1, \psi_l \in \mathcal{A}_l, \text{for } l = 1, \cdots, L\}. \end{aligned} \tag{8}$$

Here, L denotes the depth of the neural network, $M = \max(d_1, \cdots, d_L)$ represents the maximum width among layers, $N = \sum_{l \in \mathcal{L}_F}(d_l + 1)d_{l+1} + \sum_{l \in \mathcal{L}_C}(s_l + m_l)r_l$ is the total number of parameters, and $B_u = \sup_{f \in \mathcal{F}_{L,M,N,B_a,B_u}} \|f\|_\infty$ denotes the uniform supremum norm.

As an essential element, we synthesize one of the most advanced findings pertaining to neural network approximation Bao et al. [2014], Shen et al. [2019, 2021b] in Lemma 1.

Lemma 1. *Given a function $h \in \mathcal{H}^{\lambda,\alpha}([0,1]^d)$, for any $\varepsilon > 0$, there exists a function $g \in \mathcal{F}_{CNN}$ such that $\|h - g\|_\infty < \varepsilon$, for sufficiently large L, M, N, B_a, B_u.*

5.2 Regression

In the task of regression, as described in equation (6), our objective is to derive an estimation for the unknown conditional mean function h_0. This is the very reason we have resorted to deploying the least squares loss function, as demonstrated in equation (4). As such, it is of utmost importance to rigorously verify the validity of our principal representation, as laid out in equation (2).

At the population level, our repsentation (2) is expressed as

$$h(X) := \mathbb{E}\left[f(\tilde{X}, X)\tilde{Y}|X\right], \tag{9}$$

where $(\tilde{X}, \tilde{Y})$ is an independent copy of (X, Y). Based on model (6) where $\tilde{Y} = h_0(\tilde{X}) + \tilde{\epsilon}$, it then follows that

$$\begin{aligned} h(X) = \mathbb{E}\left[f(\tilde{X}, X)\tilde{Y}|X\right] &= \mathbb{E}\left[f(\tilde{X}, X)(h_0(\tilde{X}) + \tilde{\epsilon})|X\right] \\ &= \mathbb{E}\left[f(\tilde{X}, X)h_0(\tilde{X})|X\right] \\ &= \left\langle f(\tilde{X}, X), h_0(\tilde{X})\right\rangle_\mu, \end{aligned} \tag{10}$$

where μ is a deduced measure. Hence, to ensure that $h(X)$ equals to $h_0(X)$, a straightforward choice of f can be

$$f(\tilde{X}, X) = \frac{h_0(X)}{h_0(\tilde{X})}.$$

This is the fundamental reason we do not impose nonnegativity upon the kernel function f, thus offering us greater flexibility. Moreover, another potential candidate for f, if assuming h_0 lies in a reproducing kernel Hilbert space with the inner product $\langle \cdot, \cdot \rangle_\mu$, can be the hence determined kernel function. As such, we methodology can be regarded as learning an appropriate kernel that adapts to a particular task. Therefore, in the analysis hereafter, we assume that there exists a smooth function g_0 achieving $h(X) = h_0(X)$, which is summarized in Assumption 1.

Assumption 1. *There exists a function $g_0 \in \mathcal{H}^{\lambda,\alpha}([0,1]^{2d})$ where $\lambda, \alpha > 0$ such that*

$$h_0(X) = \mathbb{E}\left[g_0(\tilde{X}, X)\tilde{Y}|X\right].$$

Additionally, g_0 and h_0 are bounded by a universal constant B_0.

Subsequently, we proceed to expound upon the population and empirical properties of our methodology incorporating with neural networks. Recall that to obtain an optimal f, our objective function is defined as

$$L(f) = \mathbb{E}\left\{Y - \mathbb{E}\left[f(\tilde{X}, X)\tilde{Y}|X\right]\right\}^2,$$

at the population level and

$$\hat{L}(f) = \frac{1}{n_1}\sum_{i \in \mathscr{I}_1}\left[Y_i - \frac{1}{n_1}\sum_{j \in \mathscr{I}_1} f(X_j, X_i)Y_j\right]^2,$$

at the corresponding empirical level.

Let $f_0 = \arg\min_{f \text{ measurable}} L(f)$. It is clear that f_0 attains that $h_0(X) = \mathbb{E}[f_0(\tilde{X}, X)\tilde{Y}|X]$. Consequently, $L(f_0) = L(g_0)$, although f_0 is not necessarily equivalent to g_0 almost surely.

By introducing neural networks, we aim to achieve a satisfactory approximation of f_0 (or g_0). Noting that

$$\begin{aligned}
L(f) &= \mathbb{E}\left\{Y - \mathbb{E}\left[f(\tilde{X}, X)\tilde{Y}|X\right]\right\}^2 \\
&= \nu^2 + \mathbb{E}\left\{h_0(X) - \mathbb{E}\left[f(\tilde{X}, X)\tilde{Y}|X\right]\right\}^2 \\
&= L(g_0) + \mathbb{E}\left\{\mathbb{E}\left[g_0(\tilde{X}, X)\tilde{Y}|X\right] - \mathbb{E}\left[f(\tilde{X}, X)\tilde{Y}|X\right]\right\}^2 \\
&= L(g_0) + \mathbb{E}\left\{\mathbb{E}\left[g_0(\tilde{X}, X)h_0(\tilde{X})|X\right]\right. \\
&\qquad \left. - \mathbb{E}\left[f(\tilde{X}, X)h_0(\tilde{X})|X\right]\right\}^2 \\
&\leq L(g_0) + B_0^2\|g_0 - f\|_\infty^2,
\end{aligned}$$

we have the proposition as follows, indicating that the excess risk optimized in the neural network function class has the potential to be arbitrarily small.

Proposition 1. *For any $\varepsilon > 0$, there exists a function $f_\varepsilon \in \mathcal{F}_{CNN}$ such that $L(f_\varepsilon) - L(g_0) < \varepsilon$, for sufficiently large L, M, N, B_a, B_u related to ε.*

Let $\hat{f} = \arg\min_{f \in \mathcal{F}} \hat{L}(f)$ for a function class $\mathcal{F}$. Then, the final prediction for a new observation X is expressed as $\hat{h}(X) = \sum_{i \in \mathscr{I}_2} \hat{f}(X_i, X)Y_i/n_2$. Hence, the expected excess risk for regression is

$$R^r(\hat{f}) = \mathbb{E}\left[Y - \hat{h}(X)\right]^2 - L(g_0) = \mathbb{E}\left[Y - \hat{h}(X)\right]^2 - \nu^2.$$

The rigorous analysis of the excess risk necessitates the comprehensive and detailed deconstruction of the risk into two constituent parts, generalization error and approximation error. It is noteworthy that we are dealing with a U process with a degree of 3, and our strategy primarily focuses on the meticulous examination of its first-order projection. We will justify that the higher-order remainder term is negligible. Furthermore, we emphasize that the excess risk is effectively a stochastic process that is indexed by an infinitely diverse function class, underscoring the urgent and indispensable need to engage proactively with the latest developments in covering number theory. The integration of this advanced theoretical framework enables us to transform and re-envision the problem from an intricate domain of infinite points into a succinct and manageable set of finite balls.

To simplify the deduction, we focus on a more specific condition whereby g_0 is assumed to be symmetric. This assumption is valid in the context of kernel methods associated with a reproducing kernel Hilbert space, for instance, when g_0 is a radial basis function, such as $(x, y) \mapsto \exp(-\|x - y\|)$, or a polynomial kernel, such as $(x, y) \mapsto (1 + x^\top y)^\gamma$. To incorporate additional information, we utilize a symmetric neural network function to approximate g_0 in a natural manner. Specifically, for any function $f \in \mathcal{F}_{CNN}$, we define its symmetric accompany $f_s(x, y)$ as $f_s(x, y) = [f(x, y) + f(y, x)]/2$, where the subscript s emphasizes that the function is symmetric. This symmetrization induces a function class $\mathcal{F}^s_{CNN}$ defined as

$$\mathcal{F}^s_{CNN} = \left\{ f_s(x, y) = \frac{f(x, y) + f(y, x)}{2} : f \in \mathcal{F}_{CNN} \right\}.$$

We use $\mathcal{F}^s_{CNN}$ as the hypothesis space instead of $\mathcal{F}_{CNN}$. With regards to the approximation property, if $f \in \mathcal{F}_{CNN}$ satisfies $\|f(x, y) - g_0(x, y)\|_\infty < \varepsilon$, then we also have $\|f(y, x) - g_0(y, x)\|_\infty < \varepsilon$, since x and y lie in the same space. Therefore, we have

$$\begin{aligned}\|f_s(x, y) - g_0(x, y)\|_\infty \leq & \frac{1}{2}\|f(x, y) - g_0(x, y)\|_\infty \\ & + \frac{1}{2}\|f(y, x) - g_0(y, x)\|_\infty \leq \varepsilon,\end{aligned}$$

which demonstrates that the approximation property is also achieved for $\mathcal{F}^s_{CNN}$. Hereafter, we denote $\hat{f} = \arg\min_{f \in \mathcal{F}^s_{CNN}} \hat{L}(f)$.

The primary finding is presented in Theorem 1. Here, we consider a common scenario that, as in the kernel method, a favored kernel is the radial basis kernel, which is a mapping that relies on $x - y \in [0, 1]^d$. Therefore, we posit a lower-dimensional structure assumption as stated in Assumption 4, so that a minimax optimal error bound can be derived.

Assumption 2. *The probability measure of X is absolutely continuous with respect to Lebesgue measure.*

Assumption 3. *Y is bounded by B_0.*

Assumption 4. *There exists a function $g_0^\dagger \in \mathcal{H}^{\alpha,\lambda}([0, 2\sqrt{d}])$ such that $g_0(x, y) = g_0^\dagger(\|x - y\|_2)$.*

Theorem 1. *Suppose Assumption 1, 2, 3, 4 hold. Let $\beta = 2\alpha/(d + 2\alpha)$, $n_2 = n^\beta$ and $n_1 = n - n_2 = \mathcal{O}(n)$. Then, for sufficiently large n, it follows that*

$$R^r(\hat{f}) \leq c(\log n)^2 n^{-2\alpha/(d+2\alpha)},$$

where c is a universal constant.

To ensure clarity and brevity, the technical proofs of Theorem 1, which underpin the arguments presented in this section, have been relegated to the Appendix for greater accessibility and convenience. Nonetheless, we provide here a brief overview of the proof sketch, which encapsulates our methodology's key challenges and contributions. This proof sketch primarily comprises five essential steps: risk decomposition, projection, coverage, application of Bernstein's inequality, and balancing. It is worth noting that the assumption that Y is bounded can be further refined, for instance, by assuming that Y is sub-exponential distributed. However, this revised approach necessitates an additional truncation step and requires more nuanced modifications.

The nonasymptotic bounds presented in Theorem 1 may exhibit a sluggish pace in the event that the dimension d of input X grows rapidly. In practice, however, the possibility exists that Y is independent with the other subset of predictor variables contingent upon a particular subset of said predictors. Here, we posit the existence a matrix $A \in \mathbb{R}^{d_\delta/2 \times d}$ such that $g_0(x, y)$ can be expressed as $\tilde{g}_0(Ax, Ay)$ for some λ, α-Hölder smooth function $\tilde{g}_0$. Note that $d_\delta/2$ may have a significantly smaller value than d. We then have Theorem 2.

Assumption 5. *There exists a symmetric function $\tilde{g}_0 : \mathbb{R}^{d_\delta/2} \times \mathbb{R}^{d_\delta/2} \to \mathbb{R}$ that belongs to $\mathcal{H}^{\lambda,\alpha}([0, 1]^{d_\delta})$ where $\lambda, \alpha > 0$ and a matrix $A \in \mathbb{R}^{d_\delta/2 \times d}$ such that*

$$h_0(X) = \mathbb{E}\left[\tilde{g}_0(A\tilde{X}, AX)\tilde{Y}|X\right].$$

Additionally, $\tilde{g}_0$ and h_0 are bounded by a universal constant B_0.

Theorem 2. *Suppose Assumption 2, 3, 5 hold. Let $\beta = 2\alpha/(d_\delta + 2\alpha)$, $n_2 = n^\beta$ and $n_1 = n - n_2 = \mathcal{O}(n)$. Then, for sufficiently large n, it follows that*

$$R^r(\hat{f}) \leq c(\log n)^2 n^{-2\alpha/(d_\delta+2\alpha)},$$

where c is a universal constant.

5.3 Classification

In the context of classification, it follows that

$$\begin{aligned} h(X) &= \mathbb{E}\left[f(\tilde{X}, X)\tilde{Y}|X\right] \\ &= \Big(\mathbb{E}\left[f(\tilde{X}, X)p_1(\tilde{X})|X\right], \\ &\qquad \cdots, \mathbb{E}\left[f(\tilde{X}, X)p_K(\tilde{X})|X\right]\Big)^\top. \end{aligned}$$

To ensure that the index of the maximal value in $h(X)$ equals to that in $\mathbb{E}(Y|X)$, a natural choice for f can be made as

$$f(\tilde{X}, X) = 1\left(\operatorname*{argmax}_{1\le i\le K} p_i(\tilde{X}) = \operatorname*{argmax}_{1\le i\le K} p_i(X)\right).$$

In this instance, f serves as a correlation metric that evaluates the tendency of Y and $\tilde{Y}$ belonging to the same category.

In order to evaluate the effectiveness of a classifier $\mathscr{C}_v$ induce by a function v, specifically, $\mathscr{C}_v := \text{sgn}(v)$, the 0-1 loss is commonly used as the foundational metric, i.e., $L_0(\mathscr{C}_v) = \mathbb{P}(Y \neq \mathscr{C}_v(X))$. However, the computational infeasibility associated with optimizing this loss function renders it intractable. In light of this, surrogate losses, including but not limited to least squares loss, hinge loss, exponential loss, and logistic loss, are frequently utilized in lieu of the 0-1 loss function. Let the surrogate loss be $L_\phi(v) = \mathbb{E}\left[\phi(Y, v(X))\right]$ for an appropriate function ϕ. It is noteworthy that, under mild conditions, the excess risk attributable to the 0-1 loss function, which is expressed as

$$R_0(\mathscr{C}_v) = L_0(\mathscr{C}_v) - \inf_{\bar{v}\text{ measurable}} L_0(\mathscr{C}_{\bar{v}}),$$

can be bounded by that induced by a surrogate loss L_ϕ, which is expressed as

$$R_\phi(v) = L_\phi(v) - \inf_{\bar{v}\text{ measurable}} L_\phi(\bar{v}).$$

In the subsequent analysis, our focus is on the scenario where the response variable Y takes on binary values, specifically $Y \in \{-1, 1\}$. Let $\eta_0(X)$ denote the conditional probability of Y being equal to 1 given X, i.e., $\eta_0(X) = \mathbb{P}(Y = 1|X)$. This leads to the expression

$$\begin{aligned} h(X) = \mathbb{E}\left[f(\tilde{X}, X)\tilde{Y}|X\right] &= \mathbb{E}\left[f(\tilde{X}, X)\mathbb{E}(\tilde{Y}|\tilde{X})|X\right] \\ &= \mathbb{E}\left\{f(\tilde{X}, X)[2\eta_0(\tilde{X}) - 1]|X\right\}. \end{aligned}$$

This resembles the regression case quite closely. To achieve the equality of $h(X)$ and $\mathbb{E}(Y|X) = 2\eta_0(X) - 1$, we can simply take

$$f(\tilde{X}, X) = \frac{2\eta_0(X) - 1}{2\eta_0(\tilde{X}) - 1}.$$

We employ the sign of $h(X)$, i.e., $\mathscr{C}_h = \text{sgn}(h(X))$, as the ultimate classifier to predict the label of the new observation X. It is noteworthy that h_0 is the so-called Bayesian classifier for 0-1 loss, i.e.,

$$h_0 \in \operatorname*{argmin}_{v\text{ measurable}} \mathbb{P}(Y \neq \text{sgn}(v(X))).$$

We further adopt a surrogate loss function induced by the least squares loss $\phi(x) = (1 - x)^2$, i.e.,

$$L_\phi(f) = \mathbb{E}\left\{\left(1 - Y\mathbb{E}[f(\tilde{X}, X)\tilde{Y}|X]\right)^2\right\}.$$

The empirical version of this loss is then expressed as

$$\hat{L}_\phi(f) = \frac{1}{n_1}\sum_{i\in\mathscr{I}_1}\left[1 - \sum_{j\in\mathscr{I}_1} f(X_j, X_i)Y_iY_j\right]^2.$$

In doing so, we obtain that $R_0(\mathscr{C}_h) \le \sqrt{R_\phi(h)}$, which is a direct corollary of Theorem 1 of Bartlett and Mendelson [2006].

Some useful assumptions parallel to those employed in the context of regression are stated as follows.

Assumption 6. *There exists a symmetric function $u_0 : \mathscr{X} \times \mathscr{X} \to \mathbb{R}$ that belongs to $\mathcal{H}^{\lambda,\alpha}([0,1]^{2d})$ where $\lambda, \alpha > 0$ such that*

$$2\eta_0(X) - 1 = \mathbb{E}\left[u_0(\tilde{X}, X)\tilde{Y}|X\right].$$

Additionally, u_0 is bounded by a universal constant $B_0 \geq 1$.

Let $\hat{f} = \arg\min_{f \in \mathcal{F}^s_{CNN}} \hat{L}_\phi(f)$ and $\hat{h}(X) = \sum_{i \in \mathscr{I}_2} \hat{f}(X_i, X)Y_i/n_2$, Theorem 3 can be derived. This theorem establishes the convergence of the excess risk associated with the empirical solution $\hat{f}$ as the sample size n tends to infinity.

Theorem 3. *Suppose Assumption 2 and 6 hold. Then, for sufficiently large n, it follows that*

$$R_0(\mathscr{C}_{\hat{h}}) \leq c(\log n)^2 n^{-\alpha/(2d+2\alpha)},$$

where c is a universal constant.

6 Conclusion

In this paper, we put forward an innovative approach, Adank, for conducting weighted average estimation. The construction of weights was accomplished by implementing a kernel function utilizing a neural network. It should be noted that we purposefully avoided compelling the kernel to uphold either symmetry or nonnegativity assumptions, instead striving to maintain maximum flexibility. The resulting learned kernel embodies an intrinsic correlation between two inputs. In the specific context of classification, our adaptive kernel functions as a specialized metric, enabling the measurement of similarity between two inputs. To demonstrate our approach's efficacy, we conducted a series of numerical experiments, with a particular emphasis on scenarios that had limited observations. In addition, we established nonasymptotic excess risk bounds for both regression and classification applications. This effort can be viewed as a compelling utilization of U process theory. We believe that our proposed methodology offers significant utility for real-world prediction problems and other relevant applications.

References

Eric Bataillou, Eric Thierry, Hervé Rix, and Olivier Meste. Weighted averaging using adaptive estimation of the weights. *Signal Processing*, 44(1):51–66, 1995.

Yi Wang, Jiankun Hu, and Heiko Schroder. A gradient based weighted averaging method for estimation of fingerprint orientation fields. In *Digital Image Computing: Techniques and Applications (DICTA'05)*, pages 29–29. IEEE, 2005.

Liangxiao Jiang, Harry Zhang, Zhihua Cai, and Dianhong Wang. Weighted average of one-dependence estimators. *Journal of Experimental & Theoretical Artificial Intelligence*, 24(2):219–230, 2012.

Zongwu Cai. Weighted nadaraya–watson regression estimation. *Statistics & probability letters*, 51(3):307–318, 2001.

Oliver Kramer and Oliver Kramer. K-nearest neighbors. *Dimensionality reduction with unsupervised nearest neighbors*, pages 13–23, 2013.

Vladimir Vovk. Kernel ridge regression. *Empirical Inference: Festschrift in Honor of Vladimir N. Vapnik*, pages 105–116, 2013.

S Rasoul Safavian and David Landgrebe. A survey of decision tree classifier methodology. *IEEE transactions on systems, man, and cybernetics*, 21(3):660–674, 1991.

Gérard Biau and Erwan Scornet. A random forest guided tour. *Test*, 25:197–227, 2016.

Roman Timofeev. Classification and regression trees (cart) theory and applications. *Humboldt University, Berlin*, 54, 2004.

Rafael García Leiva, Antonio Fernández Anta, Vincenzo Mancuso, and Paolo Casari. A novel hyperparameter-free approach to decision tree construction that avoids overfitting by design. *Ieee Access*, 7:99978–99987, 2019.

Kurt Hornik. Some new results on neural network approximation. *Neural networks*, 6(8):1069–1072, 1993.

Balázs Csanád Csáji et al. Approximation with artificial neural networks. *Faculty of Sciences, Etvs Lornd University, Hungary*, 24(48):7, 2001.

Dennis Elbrächter, Dmytro Perekrestenko, Philipp Grohs, and Helmut Bölcskei. Deep neural network approximation theory. *arXiv preprint arXiv:1901.02220*, 2019.

Ronald DeVore, Boris Hanin, and Guergana Petrova. Neural network approximation. *Acta Numerica*, 30:327–444, 2021.

Zuowei Shen, Haizhao Yang, and Shijun Zhang. Neural network approximation: Three hidden layers are enough. *Neural Networks*, 141:160–173, 2021a.

Jiuxiang Gu, Zhenhua Wang, Jason Kuen, Lianyang Ma, Amir Shahroudy, Bing Shuai, Ting Liu, Xingxing Wang, Gang Wang, Jianfei Cai, et al. Recent advances in convolutional neural networks. *Pattern recognition*, 77:354–377, 2018.

Zonghan Wu, Shirui Pan, Fengwen Chen, Guodong Long, Chengqi Zhang, and S Yu Philip. A comprehensive survey on graph neural networks. *IEEE transactions on neural networks and learning systems*, 32(1):4–24, 2020.

Douglas C Montgomery, Elizabeth A Peck, and G Geoffrey Vining. *Introduction to linear regression analysis*. John Wiley & Sons, 2021.

Lawrence C Marsh and David R Cormier. *Spline regression models*. Number 137. Sage, 2001.

Laura M Sangalli, James O Ramsay, and Timothy O Ramsay. Spatial spline regression models. *Journal of the Royal Statistical Society: SERIES B: Statistical Methodology*, pages 681–703, 2013.

Serdar Demir and Öniz Toktamiş. On the adaptive nadaraya-watson kernel regression estimators. *Hacettepe Journal of Mathematics and Statistics*, 39(3):429–437, 2010.

Jorma Laaksonen and Erkki Oja. Classification with learning k-nearest neighbors. In *Proceedings of international conference on neural networks (ICNN'96)*, volume 3, pages 1480–1483. IEEE, 1996.

Senjian An, Wanquan Liu, and Svetha Venkatesh. Face recognition using kernel ridge regression. In *2007 IEEE Conference on Computer Vision and Pattern Recognition*, pages 1–7. IEEE, 2007.

Sotiris B Kotsiantis. Decision trees: a recent overview. *Artificial Intelligence Review*, 39:261–283, 2013.

Leo Breiman. Random forests. *Machine learning*, 45:5–32, 2001.

Jaime Lynn Speiser, Michael E Miller, Janet Tooze, and Edward Ip. A comparison of random forest variable selection methods for classification prediction modeling. *Expert systems with applications*, 134:93–101, 2019.

Tianqi Chen and Carlos Guestrin. Xgboost: A scalable tree boosting system. In *Proceedings of the 22nd acm sigkdd international conference on knowledge discovery and data mining*, pages 785–794, 2016.

Guolin Ke, Qi Meng, Thomas Finley, Taifeng Wang, Wei Chen, Weidong Ma, Qiwei Ye, and Tie-Yan Liu. Lightgbm: A highly efficient gradient boosting decision tree. *Advances in neural information processing systems*, 30, 2017.

Matthias Hein. Robust nonparametric regression with metric-space valued output. *Advances in neural information processing systems*, 22, 2009.

Michael M Bronstein, Joan Bruna, Yann LeCun, Arthur Szlam, and Pierre Vandergheynst. Geometric deep learning: going beyond euclidean data. *IEEE Signal Processing Magazine*, 34(4):18–42, 2017.

Alexander Petersen and Hans-Georg Müller. Fréchet regression for random objects with Euclidean predictors. *The Annals of Statistics*, 47(2):691 – 719, 2019. doi:10.1214/17-AOS1624. URL `https://doi.org/10.1214/17-AOS1624`.

Yaqing Chen and Hans-Georg Müller. Uniform convergence of local fréchet regression with applications to locating extrema and time warping for metric space valued trajectories. *The Annals of Statistics*, 50(3):1573–1592, 2022.

Z Lin, H-G Müller, and BU Park. Additive models for symmetric positive-definite matrices and lie groups. *Biometrika*, 110(2):361–379, 2023.

Mehrtash Harandi, Mathieu Salzmann, and Fatih Porikli. Bregman divergences for infinite dimensional covariance matrices. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 1003–1010, 2014.

Qi Zhang, Lingzhou Xue, and Bing Li. Dimension reduction and data visualization for fr\'echet regression. *arXiv preprint arXiv:2110.00467*, 2021.

Zhenhua Lin. Riemannian geometry of symmetric positive definite matrices via cholesky decomposition. *SIAM Journal on Matrix Analysis and Applications*, 40(4):1353–1370, 2019.

Han Xiao, Kashif Rasul, and Roland Vollgraf. Fashion-mnist: a novel image dataset for benchmarking machine learning algorithms. *arXiv preprint arXiv:1708.07747*, 2017.

Zhou Wang, Alan C Bovik, Hamid R Sheikh, and Eero P Simoncelli. Image quality assessment: from error visibility to structural similarity. *IEEE transactions on image processing*, 13(4):600–612, 2004.

Yidong Zhou and Hans-Georg MÃ¼ller. Network regression with graph laplacians. *Journal of Machine Learning Research*, 23(320):1–41, 2022. URL `http://jmlr.org/papers/v23/22-0681.html`.

David Silver, Julian Schrittwieser, Karen Simonyan, Ioannis Antonoglou, Aja Huang, Arthur Guez, Thomas Hubert, Lucas Baker, Matthew Lai, Adrian Bolton, et al. Mastering the game of go without human knowledge. *nature*, 550 (7676):354–359, 2017.

Antonia Creswell, Tom White, Vincent Dumoulin, Kai Arulkumaran, Biswa Sengupta, and Anil A Bharath. Generative adversarial networks: An overview. *IEEE signal processing magazine*, 35(1):53–65, 2018.

Luciano Floridi and Massimo Chiriatti. Gpt-3: Its nature, scope, limits, and consequences. *Minds and Machines*, 30: 681–694, 2020.

Ekrem Saralioglu and Oguz Gungor. Semantic segmentation of land cover from high resolution multispectral satellite images by spectral-spatial convolutional neural network. *Geocarto International*, 37(2):657–677, 2022.

Chenglong Bao, Qianxiao Li, Zuowei Shen, Cheng Tai, Lei Wu, and Xueshuang Xiang. Approximation analysis of convolutional neural networks. *work*, 65, 2014.

Zuowei Shen, Haizhao Yang, and Shijun Zhang. Deep network approximation characterized by number of neurons. *arXiv preprint arXiv:1906.05497*, 2019.

Guohao Shen, Yuling Jiao, Yuanyuan Lin, and Jian Huang. Non-asymptotic excess risk bounds for classification with deep convolutional neural networks. *arXiv preprint arXiv:2105.00292*, 2021b.

Peter L Bartlett and Shahar Mendelson. Empirical minimization. *Probability theory and related fields*, 135(3):311–334, 2006.