Journal of the Royal Statistical Society Series B: Statistical Methodology, 2024, **86**, 1435–1454
https://doi.org/10.1093/jrsssb/qkae038
Advance access publication 22 May 2024
Original Article

ROYAL STATISTICAL SOCIETY
DATA | EVIDENCE | DECISIONS
Journal of the Statistics Society
Series B
Statistical Methodology
B

Neural networks meet random forests

Rui Qiu[†], Shuntuo Xu[†] and Zhou Yu[†]

School of Statistics, KLATASDS-MOE, East China Normal University, Shanghai 200062, People's Republic of China

Address for correspondence: Zhou Yu, School of Statistics, KLATASDS-MOE, East China Normal University, Shanghai 200062, People's Republic of China. Email: zyu@stat.ecnu.edu.cn

Abstract

Neural networks and random forests are popular and promising tools for machine learning. This article explores the proper integration of these two approaches for nonparametric regression to improve the performance of a single approach. Specifically, we propose a neural network estimator with local enhancement provided by random forests. It naturally synthesizes the local relation adaptivity of random forests and the strong global approximation ability of neural networks. Based on the classical empirical risk minimization framework, we establish a nonasymptotic error bound for the estimator. By utilizing advanced *U*-process theory and an appropriate network structure, we can further improve the convergence rate to the nearly minimax rate. Also with the assistance of random forests, we can implement gradient learning with neural networks. Comprehensive simulation studies and real data applications demonstrate the superiority of our proposal.

Keywords: neural networks, nonparametric regression, random forests, sufficient dimension reduction

1 Introduction

Machine learning is becoming increasingly essential in scientific and social advancements. Random forests (Breiman, 2001), introduced by Leo Breiman, are an ensemble learning method that proceeds by averaging the forecasts of multiple randomized decision trees grown in parallel. The success of random forests in practice has aroused many theoretical investigations to explore intrinsic mechanisms. The early research mainly focused on the consistency (Biau et al., 2008; Denil et al., 2014) of simplified random forest algorithms. Scornet et al. (2015) proved the asymptotic consistency of Breiman' random forests, ensembles of randomized classification and regression trees (CART), for the first time under the assumption of additive model structure. Recent years have witnessed a significant breakthrough in the asymptotic distribution based on the theory of *U*-statistics. Mentch and Hooker (2016) gave the first central limit theorem for Breiman's random forests, while Wager and Athey (2018) and Peng et al. (2022) gave similar results under more general conditions. To better capture smooth signals, Friedberg et al. (2020) proposed local linear forests and established its asymptotic normality. Regarding non-asymptotic convergence theory, Biau (2012) and Klusowski (2021) focused on simplified random forests and obtained the convergence rate adapted to sparsity. And Gao and Zhou (2020) contributed to the rate analysis of the classification scenario. Klusowski and Tian (2024) provided finite sample consistency rates of the original Breiman's random forests in a high-dimensional additive model setting. Recently, an efficient online random forest algorithm called Mondrian forests was proposed by Lakshminarayanan et al. (2014) based on the Mondrian process. Then, Mourtada et al. (2020) made a detailed statistical analysis of Mondrian forests in a batch learning setting and demonstrated for the first time that some particular random forests can achieve the minimax optimal rate established by Stone (1982).

[†] R. Q., S. X., and Z. Y. contributed equally.

Received: April 20, 2023. **Revised:** April 17, 2024. **Accepted:** April 18, 2024

Deep learning is another rapidly growing branch of machine learning that adopts neural networks as function approximators, with a particular architecture of stacking many layers of structurally similar components. They possess a strong feature learning capability and tend to be state-of-the-art learners in computer vision and natural language processing tasks. To unveil the mystery of neural networks, numerous studies have been devoted to their theoretical understanding. Farrell et al. (2021) established non-asymptotic bounds for deep feedforward architecture nets using general Lipschitz loss functions, with the demand that the regression function lies in a Sobolev ball. For the normally distributed response variable Y, Bauer and Kohler (2019) derived the prediction error bound when adopting specific structured sigmoid activated multilayer neural networks to fit a similar structured regression function. Schmidt-Hieber (2020) proved that sparsely connected and well-designed neural networks with ReLU activation function can achieve the nearly minimax convergence rate under a general composition assumption on the regression function. Jiao et al. (2023) improved the prefactor of optimal prediction error bounds of ordinary neural networks to make it more meaningful in the high-dimensional setting. To circumvent the curse of dimensionality, the underlying regression function is assumed to satisfy certain hierarchical structure assumption (Bauer & Kohler, 2019; Schmidt-Hieber, 2020), low-dimensional manifold support assumption (Chen et al., 2022; Jiao et al., 2023), or locally low dimensionality assumption (Kohler et al., 2019).

Both random forests and neural networks have enjoyed immense empirical success and are suitable for handling relatively high-dimensional inputs (Chen & Ishwaran, 2012; Hinton & Salakhutdinov, 2006). From the structure, they are similar: a tree appears like a vertically growing neural network. While neural networks have shown strong empirical performance for complex data types, this dominance is not always seen with more standard tabular data, which is the stronghold of random forests. The random forest excels at handling local characteristics and requires less training data due to its input space partitioning mechanism and fewer tuning parameters. From this point of view, random forests can supplement neural networks to some extent. In fact, the concept of combining these two excellent models has already appeared in the literature. Biau et al. (2019) translated each decision tree of a random forest into a three-layer neural network and proposed two different ways to combine all individual networks extracted from all trees. They established the consistency of the proposed neural random forests method under some assumptions about the data distribution and the regression function. Zhou and Feng (2019) proposed a non-differentiable style deep model called deep forests, which has a cascade forest structure. It is an ensemble learning approach based on random forests without using backpropagation. The two approaches described above are essentially quite different. The former exploits prior knowledge of regression trees for the initialization of neural networks and still relies on neural networks for predictions. The latter uses random forests instead of neurons to verify the possibility of implementing effective deep learning with non-differentiable modules, which is not essentially a neural network and is mainly applied in classification tasks. Other studies investigating the intersection of random forests and neural networks include works by Kontschieder et al. (2015) and Arnould et al. (2021), and the references in these papers.

Like neural random forests (Biau et al., 2019), this article still uses the traditional neural network structure and focuses on the regression task. Local regression takes into account the contributions of sample points in the vicinity of the target, with the local contribution typically measured by kernel functions (Fan & Gijbels, 2018). However, kernel smooth methods tend to fail in situations with moderately high-dimensional predictors. Recently, scholars have discovered that random forests can also generate local weights adaptively (Lin & Jeon, 2006; Scornet, 2016), which remain valid even in relatively high-dimensional situations. Our method considers incorporating the local modelling capability of random forests into the global regression framework of neural networks, resulting in a novel global regression model with local enhancement. The implicit incorporation of a horizontally growing neural network and a vertically growing random forest will further improve the performance of neural networks even with only a small size of training samples. In particular, we capitalize on the ability of random forests to recognize patterns in tabular data through an input space partitioning mechanism to complement neural networks in tabular data analysis. CART chooses the best split dimension and location from all candidate splits at each node by maximizing information gains in a greedy manner. Due to the complex recursive splitting, the latest consistency results regarding Breiman's random forests halt at the additive model setting

(Klusowski & Tian, 2024; Scornet et al., 2015) or rely on strong assumptions concerning the regression model and tree construction (Chi et al., 2022). In contrast, the tree growth mechanism of Mondrian forests presents an excellent geometric foundation, facilitating the explicit computation of some quantities related to the structure of Mondrian partitions. And Mondrian forests exhibit favourable convergence properties under mild conditions (Mourtada et al., 2020). Therefore, the theoretical investigation in this article will centre on the Mondrian forest.

The remainder of the article is organized as follows. In Section 2, we introduce some notations and propose a new neural network type of regression method with the assistance of random forests. In Section 3, we establish non-asymptotic error bounds of the proposed estimator and discuss the problem of circumventing the curse of dimensionality. In Section 4, some extensions including gradient learning are elaborated to complete the main content of the article. In Sections 5 and 6, comprehensive comparisons among relevant methods by numerical experiments and real data applications are presented. Finally, Section 7 concludes the article with some discussions. All proofs are relegated to online supplementary material.

2 Methods

2.1 Notations

We first summarize the notations and concepts that will appear later in this article. Vectors are denoted by lowercase letters. As usual, $\|x\|_r$ and $\|x\|_\infty$ represent the L_r norm and supremum norm of x, respectively. And random vectors are denoted by uppercase letters. We use $\mathbb{E}$ to denote the expectation taken with respect to all randomness in the target. If the expectation is taken for only part of the randomness, we will indicate it by an index. Let $L_2(\mathbb{P}_X)$ be the space of measurable functions defined on Ω_X that are square integrable with respect to $\mathbb{P}_X$. For any function $f \in L_2(\mathbb{P}_X)$, we define $\|f\|_{L_2(\mathbb{P}_X)} = \{\mathbb{E}|f(X)|^2\}^{1/2}$ and $\|f\|_\infty = \sup_x |f(x)|$ as the L_2 norm and supremum norm of f, respectively.

The Mondrian process plays a crucial role in the construction of Mondrian trees (see Mourtada et al., 2020 for detail). A Mondrian partition of the feature space C is sampled from the Mondrian process distribution MP(λ, C), which can be achieved by applying the recursive procedure SampleMondrian(C, $\tau = 0$, λ) described in the online supplementary material. λ is called the lifetime parameter for controlling the complexity of the partition, in the sense that larger values of λ correspond to deeper trees. Any sample drawn from MP(λ, C) generates a recursive partition of C, thereby resulting in a Mondrian tree. A Mondrian forest is an ensemble of multiple Mondrian trees. In this article, we utilize Mondrian Forests to generate local weights rather than for prediction.

Multi-layer perceptrons are a crucial and widely utilized class of feedforward neural networks in practical applications. A multi-layer perceptron with depth $\mathcal{D}$ can be expressed as a composition of linear transformations $\mathcal{L}(\cdot)$ and activation functions $\sigma(\cdot)$ like

$$f(\cdot) = \mathcal{L}_\mathcal{D} \circ \sigma \circ \mathcal{L}_{\mathcal{D}-1} \circ \sigma \circ \cdots \circ \sigma \circ \mathcal{L}_1 \circ \sigma \circ \mathcal{L}_0(\cdot),$$

where $\mathcal{L}_i(x) = W_i x + b_i$, $i = 0, 1, \ldots, \mathcal{D}$, represent the connections between the ith layer and $(i+1)$th layer. In the above ith linear transformation $\mathcal{L}_i$, $W_i \in \mathbb{R}^{k_{i+1} \times k_i}$ is called weight matrix and $b_i \in \mathbb{R}^{k_{i+1}}$ is called the bias vector with k_i being the width (the number of neurons or nodes) of the ith layer. The input layer is the 0th layer where input vectors enter and the output layer is the last one which gives the final calculation results. The remaining $\mathcal{D}$ layers are hidden layers performing nonlinear transformations on the input. The width of each layer of the above neural network $f(\cdot)$ can be characterized by a $(\mathcal{D}+2)$-vector $(k_0, k_1, \ldots, k_\mathcal{D}, k_{\mathcal{D}+1})^\top$. The following notations are also closely related to the structure of neural networks:

- Width $\mathcal{W}$: the maximum width of hidden layers, i.e. $\mathcal{W} = \max\{k_1, \ldots, k_\mathcal{D}\}$;
- Size $\mathcal{S}$: the total number of parameters, i.e. $\mathcal{S} = \sum_{i=0}^{\mathcal{D}} \{(k_i + 1) \times k_{i+1}\}$.
- Number of neurons $\mathcal{U}$: the number of nodes in hidden layers, i.e. $\mathcal{U} = \sum_{i=1}^{\mathcal{D}} k_i$.

Let $\mathcal{F}_{\mathcal{D},\mathcal{W},\mathcal{S},\mathcal{U}}$ be the class of multi-layer perceptrons with depth $\mathcal{D}$, width $\mathcal{W}$, size $\mathcal{S}$ and number of neurons $\mathcal{U}$ using the rectified linear unit (ReLU) activation function $\sigma(x) = \max\{0, x\}$. For notation simplicity, we will abbreviate $\mathcal{F}_{\mathcal{D},\mathcal{W},\mathcal{S},\mathcal{U}}$ to $\mathcal{F}_n$ throughout the article.

2.2 Neural networks with random forest enhancement

We consider a regression framework. For a random vector $(X, Y) \sim \mathbb{P}$, where $X \in [0, 1]^p$ and Y is $\mathbb{R}$-valued with $\mathbb{E}Y^2 < \infty$, our target is to find a (measurable) function $f^* : \mathbb{R}^p \to \mathbb{R}$ that minimize the mean squared error

$$f^* = \underset{f}{\operatorname{argmin}}\, R(f) = \underset{f}{\operatorname{argmin}}\, \mathbb{E}[\{Y - f(X)\}^2].$$

Actually, such a function has an explicit expression $f^*(x) = \mathbb{E}(Y \mid X = x)$ called the regression function. Given the data $D_n = \{(X_1, Y_1), \ldots, (X_n, Y_n)\}$ from $\mathbb{P}$, the regression function f^* can be estimated by the least squares method

$$\hat{f}_n^{ls} = \underset{f \in \mathcal{F}_n}{\operatorname{argmin}}\, \frac{1}{n} \sum_{i=1}^{n} \{Y_i - f(X_i)\}^2 \tag{1}$$

among the function class $\mathcal{F}_n$ introduced before.

Let's recall another local nonparametric method known as Nadaraya–Watson regression

$$\hat{f}_n^{nw}(x) = \underset{y}{\operatorname{argmin}}\, \frac{1}{n} \sum_{i=1}^{n} K_h(X_i - x)(Y_i - y)^2, \tag{2}$$

where K is a smoothing kernel such as the Epanechnikov kernel or Gaussian kernel and h is a bandwidth, with $K_h(\cdot) = h^{-1}K(\cdot/h)$. This method relies on the rationality that Y should be informative for the estimation of $f^*(x)$ if the corresponding X is close to x. The kernel measures the contribution of Y_i according to the distance between X_i and x. However, the accuracy of the classical local nonparametric regression is limited for moderately high-dimensional problems. (The popular core R function loess allows only 1–4 predictors Friedberg et al., 2020.) Random forests can be also regarded as a local kernel type approach with a more powerful weighting kernel function, especially for relatively high-dimensional settings. A natural idea is to combine the advantages of global neural network regression (1) and random forest kernel version of local regression (2). Specifically,

1. For some fixed integer $M > 0$, M partitions $\Pi_\lambda^{(1)}, \ldots, \Pi_\lambda^{(M)}$ sampled from $\mathrm{MP}(\lambda, [0, 1]^p)$ decide M Mondrian trees and then form a Mondrian forest, which can generate the kernel function (Scornet, 2016)

$$w_\lambda(x, x') = \frac{1}{M} \sum_{j=1}^{M} \mathbb{I}\{x' \in A(x, \Pi_\lambda^{(j)})\}, \tag{3}$$

   where $A(x, \Pi_\lambda^{(j)})$ is the leaf node containing x of the jth Mondrian tree based on the partition $\Pi_\lambda^{(j)}$. This kernel is equivalent to the empirical probability that x and x' share a leaf node within the random forest.
2. Consider the following target and optimize over the class of neural networks

$$\hat{f}_{\tau,n} = \underset{f \in \mathcal{F}_n}{\operatorname{argmin}}\, \frac{\tau}{n} \sum_{k=1}^{n} \{Y_k - f(X_k)\}^2 + \frac{1 - \tau}{n(n-1)} \sum_{i \neq j} \{Y_j - f(X_i)\}^2 w_\lambda(X_i, X_j), \tag{4}$$

   where $0 < \tau \leq 1$ is a trade-off parameter governing the proportion of two parts.

The first part of (4) takes into account the fitting error from the paired data (X_k, Y_k), while the second part considers the contribution of (X_j, Y_j) to the estimation of $f^*(X_i)$. Additionally, the

second part implicitly incorporates a regularization effect that encourages $f(X_j)$ to be close to $f(X_i)$ if $w_\lambda(X_i, X_j)$ is large, given that $\{Y_j - f(X_i)\}^2$ can be expressed as $\{Y_j - f(X_j) + f(X_i) - f(X_i)\}^2$. We call (4) the random forest weighted neural network (RWN for short when there is no ambiguity). The hyperparameters of the whole algorithm above mainly involve the trade-off parameter τ, the number M of trees, the lifetime parameter λ of trees, as well as the depth $\mathcal{D}$ and the width $\mathcal{W}$ of neural networks. Some of them can be used empirically with fixed values and the rest can be selected by cross-validation. The details are covered in the simulation section.

The predictive capability of $\hat{f}_{\tau,n}$ can be measured by the excess risk, defined as

$$R(\hat{f}_{\tau,n}) - R(f^*) = \mathbb{E}_X[\{\hat{f}_{\tau,n}(X) - f^*(X)\}^2].$$

We next study the non-asymptotic upper bounds of the excess risk $R(\hat{f}_{\tau,n}) - R(f^*)$ and the prediction error $\mathbb{E}_{D_n,w_\lambda}\{R(\hat{f}_{\tau,n}) - R(f^*)\}$, where the expectation is taken with respect to the training data D_n and Mondrian partitions from the random forest kernel $w_\lambda(\cdot, \cdot)$. Actually, the prediction error is $\mathbb{E}[\{\hat{f}_{\tau,n}(X) - f^*(X)\}^2]$.

3 Theory

We first decompose the excess risk into two terms. Let $z = (x, y)$, $z' = (x', y')$, and

$$h_f(z, z') = \frac{\{y - f(x')\}^2 w_\lambda(x', x) + \{y' - f(x)\}^2 w_\lambda(x, x')}{2},$$

which is a symmetric function. Then, we can write (4) as

$$\begin{aligned}\hat{f}_{\tau,n} &= \operatorname*{argmin}_{f\in\mathcal{F}_n} \frac{\tau}{n}\sum_{k=1}^{n}\{Y_k - f(X_k)\}^2 + \frac{1-\tau}{n(n-1)}\sum_{i\neq j} h_f(Z_i, Z_j)\\ &\triangleq \operatorname*{argmin}_{f\in\mathcal{F}_n} \tau\cdot R_n(f) + (1-\tau)\cdot L_n(f)\\ &\triangleq \operatorname*{argmin}_{f\in\mathcal{F}_n} R_{\tau,n}(f).\end{aligned} \tag{5}$$

To facilitate our analysis, we need to introduce the population form of (5)

$$\begin{aligned}f^*_\tau &= \operatorname*{argmin}_{f} \tau\cdot\mathbb{E}[\{Y - f(X)\}^2] + (1-\tau)\cdot\mathbb{E}\{h_f(Z, Z')\}\\ &\triangleq \operatorname*{argmin}_{f} \tau\cdot R(f) + (1-\tau)\cdot L(f)\\ &\triangleq \operatorname*{argmin}_{f} R_\tau(f).\end{aligned} \tag{6}$$

Based on f^*_τ, we have the following upper bound for the excess risk

$$R(\hat{f}_{\tau,n}) - R(f^*) \le 2\mathbb{E}_X[\{\hat{f}_{\tau,n}(X) - f^*_\tau(X)\}^2] + 2\mathbb{E}[\{f^*_\tau(X) - f^*(X)\}^2].$$

We call $\mathbb{E}_X[\{\hat{f}_{\tau,n}(X) - f^*_\tau(X)\}^2]$ the estimation term and $\mathbb{E}[\{f^*_\tau(X) - f^*(X)\}^2]$ the bias term. To bound the bias term, the following assumptions are needed:

(A1) The function f^* is a Hölder continuous function of order $\alpha \in (0, 1]$ with Hölder constant $\gamma > 0$, i.e. $|f^*(x_1) - f^*(x_2)| \le \gamma\|x_1 - x_2\|_2^\alpha$ for any $x_1, x_2 \in [0, 1]^p$.
(A2) X has a bounded density f_X with respect to the Lebesgue measure, and let $\overline{p} = \sup_{x\in[0,1]^p} f_X(x)$.

The assumption (A1) about the continuity of the regression function is common and mild in many existing works such as (Jiao et al., 2023; Mourtada et al., 2020). We now investigate the impact on

the optimal solution when introducing the random forest kernel into the classical squared error. Utilizing the exact distribution of the leaf node $A(x, \Pi_\lambda^{(j)})$ containing x from the Mondrian partition, as detailed in Mourtada et al. (2020), we can derive precise upper bounds on the volumes and diameters of these leaf nodes. Such advantageous geometric properties are generally unavailable for other complex random forests, such as Breiman's random forests. Leveraging these bounds (determined by λ) and the smooth assumption of the regression function f^*, the variability of f^* within the leaf nodes can be effectively controlled. Finally, by the fact $R_\tau(f_\tau^*) \le R_\tau(f^*)$, this control can be translated to the gap between f_τ^* and f^*.

Theorem 1 Let f_τ^* be the minimizer of the population risk $R_\tau(f)$. Under the assumption (A1) and (A2), we have

$$\mathbb{E}[\{f_\tau^*(X) - f^*(X)\}^2] \le \frac{1-\tau}{\tau} \cdot C_1 \overline{p} \gamma^2 \frac{1}{\lambda^{p+2\alpha}},$$

where C_1 is a constant depending on p and α.

When the parameter λ increases, the bias tends to decrease. Larger λ leads to deeper trees, which results in smaller leaf size and smaller value of the random forest kernel function, thus weakening the local power of random forests for the regression with neural networks. For the estimation term $\mathbb{E}_X[\{\hat{f}_{\tau,n}(X) - f_\tau^*(X)\}^2]$, the classical theory of empirical risk minimization is instrumental to our analysis. The optimization objective $R_{\tau,n}$ in (5) is a weighted summation of the i.i.d. sum $R_n(f)$ and the U-statistic $L_n(f)$ for any fixed f. The core of obtaining the convergence rate presented below is to deal with $L_n(f)$ through the form of an independent sum.

3.1 Slow rate

A common decomposition about R_τ in (6) is

$$R_\tau(\hat{f}_{\tau,n}) - R_\tau(f_\tau^*) = \left\{ R_\tau(\hat{f}_{\tau,n}) - \inf_{f \in \mathcal{F}_n} R_\tau(f) \right\} + \left\{ \inf_{f \in \mathcal{F}_n} R_\tau(f) - R_\tau(f_\tau^*) \right\},$$

The first term of the right-hand side is referred to as the statistical error, and the second term is called the approximation error. The statistical error measures the risk distance between the estimator $\hat{f}_{\tau,n}$ and the best function in $\mathcal{F}_n$. It indicates how good the sample estimate is under the scope of $\mathcal{F}_n$. The approximation error measures the extent to which the function f_τ^* can be approximated by functions of $\mathcal{F}_n$ in the sense of risk R_τ. It shows how dense $\mathcal{F}_n$ is in the function class where f_τ^* belongs to. The approximation theory of neural networks has been well researched in many works, especially some recent papers (Shen et al., 2019, 2020, 2022; Yarotsky, 2017, 2018). As for the statistical error, a common approach is to analyse the upper bound for a uniform distance between the population risk R_τ and corresponding empirical risk $R_{\tau,n}$ over $\mathcal{F}_n$ instead, followed by the empirical process theory. Overall, we shift attention to the following relationship

$$\begin{aligned} & R_\tau(\hat{f}_{\tau,n}) - R_\tau(f_\tau^*) \\ & \quad \le 2 \sup_{f \in \mathcal{F}_n} |R_{\tau,n}(f) - R_\tau(f)| + \left\{ \inf_{f \in \mathcal{F}_n} R_\tau(f) - R_\tau(f_\tau^*) \right\} \\ & \quad \le 2\tau \cdot \sup_{f \in \mathcal{F}_n} |R_n(f) - R(f)| + 2(1-\tau) \cdot \sup_{f \in \mathcal{F}_n} |L_n(f) - L(f)| + \left\{ \inf_{f \in \mathcal{F}_n} R_\tau(f) - R_\tau(f_\tau^*) \right\}. \end{aligned}$$

To get the convergence rate of $\hat{f}_{\tau,n}$, two more assumptions are needed.

(A3) The function f_τ^* is a Hölder continuous function of order α with Hölder constant $\gamma' > 0$, i.e. $|f_\tau^*(x_1) - f_\tau^*(x_2)| \le \gamma' \|x_1 - x_2\|_2^\alpha$ for any $x_1, x_2 \in [0,1]^p$.
(A4) There exists an absolute constant B, *s.t.* $|Y| \le B$.

The assumption (A4) inherently implies $f_\tau^* \le B$. Let $\mathcal{F}_{n,B} = \{f \in \mathcal{F}_n : \|f\|_\infty \le B\}$. The function class $\mathcal{F}_{n,B}$ suffices for approximating f_τ^* due to $f_\tau^* \le B$. Therefore, without loss of generality, we assume the functions in $\mathcal{F}_n$ are bound by B with respect to the supremum norm from now on.

When $\tau = 1$, $\hat{f}_{\tau,n}$ is just the ordinary least squares estimation using neural networks, which has been studied by some work, such as Bauer and Kohler (2019) and Jiao et al. (2023). For the case $0 < \tau < 1$, by analysing $\sup_{f\in\mathcal{F}_n}|R_n(f) - R(f)|$ and $\sup_{f\in\mathcal{F}_n}|L_n(f) - L(f)|$ separately, we obtain the following theorem.

Theorem 2 For $0 < \tau < 1$, assume (A2)–(A4) hold true. Let $\pi > 0$. For any N, $U \in \mathbb{N}^+$, consider the ReLU multi-layer perceptrons $\mathcal{F}_n$ with width $\mathcal{W} = 12\max\{p\lfloor N^{1/p}\rfloor, N+1\}$ and depth $\mathcal{D} = 12U + 14$. Let $V = \mathcal{W}^2\mathcal{D}^2\log(\mathcal{W}^2\mathcal{D})$. For $n \gtrsim V$, then with probability at least $1-\pi$, the estimator $\hat{f}_{\tau,n}$ from (4) satisfies

$$R_\tau(\hat{f}_{\tau,n}) - R_\tau(f_\tau^*) \le C_2\left((NU)^{-4\alpha/p} + \sqrt{\frac{V}{n}} + \sqrt{\frac{\log(1/\pi)}{n}} + \sqrt{\frac{\log(n(n-1)/\pi)}{M}}\right),$$

where C_2 is a constant which may depend on τ, B, γ', p, $\overline{p}$.

Actually, V defined above is just the nearly tight VC dimension bound of the function class constructed by feedforward neural networks with the piecewise-linear activation function (see Theorem 7 of Bartlett et al. (2019)). Combine the Theorem 1 and 2, then the excess risk of the estimator $\hat{f}_{\tau,n}$ can be determined by choosing suitable $\mathcal{F}_n$ and λ. Here we consider the neural network structure with fixed width $\mathcal{W}$, which has been shown to have the highest efficiency (see Jiao et al., 2023).

Corollary 1 For $0 < \tau < 1$, assume (A1)–(A4) hold true. Let $\pi > 0$. For any $N \in \mathbb{N}^+$, consider the ReLU multi-layer perceptrons $\mathcal{F}_n$ with width $\mathcal{W} = 12\max\{p\lfloor N^{1/p}\rfloor, N+1\}$ and depth $\mathcal{D} = 12\lceil n^{\frac{p}{2p+8\alpha}}\rceil + 14$. Let $V = \mathcal{W}^2\mathcal{D}^2\log(\mathcal{W}^2\mathcal{D})$. For $n \gtrsim V$, if $\lambda \gtrsim n^{\frac{2\alpha}{(p+4\alpha)(p+2\alpha)}}$, then with probability at least $1-\pi$, the excess risk of $\hat{f}_{\tau,n}$ from (4) satisfies

$$\mathbb{E}_X[\{\hat{f}_{\tau,n}(X) - f^*(X)\}^2] \le C_3\left(n^{-\frac{2\alpha}{p+4\alpha}}(\log n)^{\frac{1}{2}} + \sqrt{\frac{\log(1/\pi)}{n}} + \sqrt{\frac{\log(n(n-1)/\pi)}{M}}\right);$$

furthermore, if $M \gtrsim n^{\frac{4\alpha}{p+4\alpha}}$, the prediction error of $\hat{f}_{\tau,n}$ satisfies

$$\mathbb{E}[\{\hat{f}_{\tau,n}(X) - f^*(X)\}^2] \le C_3' n^{-\frac{2\alpha}{p+4\alpha}}(\log n)^{\frac{1}{2}}.$$

The above C_3, C_3' are constants which may depend on τ, B, γ, γ', p, $\overline{p}$, N, α.

Compare to the minimax rate $n^{-\frac{2\alpha}{p+2\alpha}}$ (Stone, 1982), the rate established here is not satisfactory. The treatment of the U-statistic term $\sup_{f\in\mathcal{F}_n}|L_n(f) - L(f)|$ is critical because it dominates the convergence rate of $\hat{f}_{\tau,n}$. Here, we analyse it by splitting the data into two halves and transforming $L_n(f)$ into an independent sum. Then, some classic empirical process results such as Dudley's entropy integral inequality are applied to bound $\sup_{f\in\mathcal{F}_n}|L_n(f) - L(f)|$. To improve the rate, we consider the local Rademacher complexity technique (Bartlett et al., 2005) to handle the statistical error with the exponential bounds of U-processes in the ensuing subsection.

3.2 Fast rate

In this section, more properties of U-statistics are exploited to derive a tighter upper bound for the excess risk. More precisely, $L_n(f)$ is a U-statistic with random kernel h_f due to $w_\lambda(\cdot,\cdot)$. We need to further introduce the following intermediate quantity to eliminate this randomness:

$$\bar{L}_n(f) = \frac{1}{n(n-1)}\sum_{i\neq j}\bar{h}_f(Z_i, Z_j),$$

where $\bar{h}_f(z,z') = \frac{\{y-f(x')\}^2\bar{w}_\lambda(x',x)+\{y'-f(x)\}^2\bar{w}_\lambda(x,x')}{2}$ with $\bar{w}_\lambda(x,x') = \mathbb{E}w_\lambda(x,x') = \frac{1}{M}\sum_{j=1}^{M}\mathbb{I}\{x' \in A(x,\Pi_\lambda^{(j)})\} = \mathbb{P}\{x' \in A(x,\Pi_\lambda)\}$. Here, the partition Π_λ is sampled from the Mondrian process distribution $\mathrm{MP}(\lambda, [0,1]^p)$ and independent of the training data D_n. Set

$$\bar{q}_f(z,z') = \bar{h}_f(z,z') - \bar{h}_{f_\tau^*}(z,z')$$

and consider the following excess risk

$$\begin{aligned}\Lambda_\tau(f) &= R_\tau(f) - R_\tau(f_\tau^*)\\ &= \tau\cdot\mathbb{E}\Big[\{Y-f(X)\}^2 - \{Y-f_\tau^*(X)\}^2\Big] + (1-\tau)\cdot\mathbb{E}\{\bar{q}_f(Z,Z')\}.\end{aligned}$$

The corresponding estimator is

$$\Lambda_{\tau,n}(f) = \frac{\tau}{n}\sum_{k=1}^{n}\Big[\{Y_k-f(X_k)\}^2 - \{Y_k-f_\tau^*(X_k)\}^2\Big] + \frac{1-\tau}{n(n-1)}\sum_{i\neq j}\bar{q}_f(Z_i,Z_j).$$

We consider the Hoeffding decomposition (Hoeffding, 1948) of the second term of $\Lambda_{\tau,n}(f)$

$$\frac{1}{n(n-1)}\sum_{i\neq j}\bar{q}_f(Z_i,Z_j) = \mathbb{E}\{\bar{q}_f(Z,Z')\} + 2T_n(f) + W_n(f),$$

where the first-order term $T_n(f) = \frac{1}{n}\sum_{i=1}^{n}\bar{q}_{1,f}(Z_i)$ is a sum of i.i.d. random variables with $\bar{q}_{1,f}(z) = \mathbb{E}\{\bar{q}_f(z,Z')\} - \mathbb{E}\{\bar{q}_f(Z,Z')\}$, and the second-order term $W_n(f) = \frac{1}{n(n-1)}\sum_{i\neq j}\bar{q}_{2,f}(Z_i,Z_j)$ is a degenerate U-statistic with a symmetric kernel $\bar{q}_{2,f}(z,z') = \bar{q}_f(z,z') - \mathbb{E}\{\bar{q}_f(Z,Z')\} - \bar{q}_{1,f}(z) - \bar{q}_{1,f}(z')$.

In the Hoeffding decomposition of U-statistics, higher order terms are always negligible compared to lower order terms. We combine the first term of $\Lambda_{\tau,n}(f)$ and the first two terms of the above Hoeffding decomposition as a new target

$$v_{\tau,n}(f) = \frac{1}{n}\sum_{i=1}^{n}\ell_\tau\big(f,Z_i\big),$$

where $\ell_\tau(f,z) = \tau\{y-f(x)\}^2 + 2(1-\tau)\mathbb{E}\{\bar{h}_f(z,Z')\} - (1-\tau)\mathbb{E}\{\bar{h}_f(Z,Z')\}$. Here, we have omitted the part related to f_τ^* as it has nothing to do with the optimization. Thus, we transform the original optimization target $R_{\tau,n}(f)$ into an independent sum $v_{\tau,n}(f)$. However, removing $W_n(f)$ and replacing h_f with $\bar{h}_f$ in this transformation process will result in a gap. Actually, this gap is influenced by $\sup_{f\in\mathcal{F}_n}|L_n(f) - \bar{L}_n(f)| + \sup_{f\in\mathcal{F}_n}|W_n(f)|$, which is relatively small with high probability by empirical process theory and U-process theory. Noting that the expectation of $v_{\tau,n}(f)$ is still $R_\tau(f)$, the centred empirical process follows as

$$\bar{v}_{\tau,n}(f) = \frac{1}{n}\sum_{i=1}^{n}\ell_\tau\big(f,Z_i\big) - R_\tau(f), \quad \text{for every } f\in L_2(\mathbb{P}_X).$$

Our ultimate goal is to verify a nondecreasing and continuous function $\phi:(0,\infty)\mapsto(0,\infty)$ such that $x\to\phi(x)/x$ is nonincreasing on $(0,+\infty)$, $\phi(1)\geq 1$ and

$$\sqrt{n}\mathbb{E}\left[\sup_{f\in\mathcal{F}_n, d_\tau\left(f_{\tau,n}^*,f\right)\leq\sigma}\{\bar{v}_{\tau,n}\Big(f_{\tau,n}^*\Big) - \bar{v}_{\tau,n}(f)\}\right] \leq \phi(\sigma),$$

where $f_{\tau,n}^*$ is the minimizer of $R_\tau(f)$ over $\mathcal{F}_n$ and d_τ is some pseudo-distance defined in the proof of the following theorem. The solution to the equation $\phi(\sigma) = \sqrt{n}\sigma^2$ is usually the convergence rate of

the sample minimizer (see Theorem 3.2.5 of Vaart and Wellner (1996)). The above process is effectively used to derive the convergence of M-estimators in the empirical process theory, but in comparison, our problem differs in two aspects. First, the estimator $\hat{f}_{\tau,n}$ is obtained by minimizing $R_{\tau,n}(f)$ instead of the independent sum $v_{\tau,n}(f)$. Here, we only take advantage of the empirical process theory as a bridge. One eventually needs to take into account the effect of the bias caused by ignoring the higher order terms of Hoeffding decomposition and replacing h_f with $\bar{h}_f$. Secondly, in addition to the statistical error analysed by the empirical process theory, the approximation error still needs to be included in the analysis, since the minimizer f_τ^* of the population risk R_τ does not necessarily fall into the class $\mathcal{F}_n$. With these issues well addressed, we reach the following conclusion.

Theorem 3 For $0 < \tau < 1$, assume (A2)–(A4) hold true. Let $\pi > 0$. For any N, $U \in \mathbb{N}^+$, consider the ReLU multi-layer perceptrons $\mathcal{F}_n$ with width $\mathcal{W} = 12\max\{p\lfloor N^{1/p}\rfloor, N+1\}$ and depth $\mathcal{D} = 12U + 14$. Let $V = \mathcal{W}^2\mathcal{D}^2 \log(\mathcal{W}^2\mathcal{D})$. For $n \gtrsim V$, then with probability at least $1-\pi$, the estimator $\hat{f}_{\tau,n}$ from (4) satisfies

$$R_\tau(\hat{f}_{\tau,n}) - R_\tau(f_\tau^*) \le C_4\left((NU)^{-4\alpha/p} + \frac{V}{n} + \frac{V\log(1/\pi)}{n} + \sqrt{\frac{\log(n(n-1)/\pi)}{M}}\right),$$

where C_4 is a constant which may depend on τ, B, λ', p, $\overline{p}$.

Combine this result with Theorem 1, then we get

Corollary 2 For $0 < \tau < 1$, assume (A1)–(A4) hold true. Let $\pi > 0$. For any $N \in \mathbb{N}^+$, consider the ReLU multi-layer perceptrons $\mathcal{F}_n$ with width $\mathcal{W} = 12\max\{p\lfloor N^{1/p}\rfloor, N+1\}$ and depth $\mathcal{D} = 12\lceil n^{\frac{p}{2p+4\alpha}}\rceil + 14$. Let $V = \mathcal{W}^2\mathcal{D}^2\log(\mathcal{W}^2\mathcal{D})$. For $n \gtrsim V$, if $\lambda \gtrsim n^{\frac{2\alpha}{(p+2\alpha)^2}}$, then with probability at least $1-\pi$, the excess risk of $\hat{f}_{\tau,n}$ from (4) satisfies

$$\mathbb{E}_X[\{\hat{f}_{\tau,n}(X) - f^*(X)\}^2] \le C_5\left(n^{-\frac{2\alpha}{p+2\alpha}}\log n\{1+\log(1/\pi)\} + \sqrt{\frac{\log(n(n-1)/\pi)}{M}}\right);$$

further, if $M \gtrsim n^{\frac{4\alpha}{p+2\alpha}}$, the prediction error of $\hat{f}_{\tau,n}$ satisfies

$$\mathbb{E}[\{\hat{f}_{\tau,n}(X) - f^*(X)\}^2] \le C_5' n^{-\frac{2\alpha}{p+2\alpha}}\log n.$$

The above C_5, C_5' are constants which may depend on τ, B, γ, γ', p, $\overline{p}$, N, α.

This result improves the rate $n^{-\frac{2\alpha}{p+4\alpha}}$ from Corollary 1 to $n^{-\frac{2\alpha}{p+2\alpha}}$ (ignoring $\log n$). It entails that RWN is still optimal in the sense of a non-asymptotic convergence rate.

3.3 Circumventing the curse of dimensionality

The convergence rate of $\hat{f}_{\tau,n}$ can be extremely slow if p is large, a phenomenon often called the curse of dimensionality. But in practice, it is often found that the support of X is concentrated on a much lower dimensional manifold embedded in the high-dimensional space. This insight offers hope for lessening the curse of dimensionality. Jiao et al. (2023) assumed

(A5) The predictor X is supported on $\mathcal{M} \subset [0,1]^p$, where $\mathcal{M}$ is a compact $p_{\mathcal{M}}$-dimensional Riemannian manifold isometrically embedded in $\mathbb{R}^p$ with condition number $(1/r)$ and area of surface $S_{\mathcal{M}}$.

and gave a complete answer to the approximation error of deep ReLU neural networks under this low-dimensional data structure. The condition number and the surface area characterize the geometric properties of manifolds. More details and references can be found in Jiao et al. (2023).

Based on the assumption (A5), the convergence rate of $\hat{f}_{\tau,n}$ can be controlled by the intrinsic dimension $p_{\mathcal{M}}$ of the manifold, not just by the ambient dimension p. This highlights the capability of RWN to adapt to low-dimensional geometric patterns in data.

Theorem 4 For $0 < \tau < 1$, assume (A1)–(A5) hold true. Let $\pi > 0$. For any $N \in \mathbb{N}^+$, consider the ReLU multi-layer perceptrons $\mathcal{F}_n$ with width $\mathcal{W} = 266\lceil S_{\mathcal{M}}(6/r)^{p_{\mathcal{M}}}\rceil p_{\mathcal{M}}^2 N\lceil\log(8N)\rceil$ and depth $\mathcal{D} = 21\left\lceil n^{\frac{p_{\mathcal{M}}}{2p_{\mathcal{M}}+4\alpha}}\log(8n^{\frac{p_{\mathcal{M}}}{2p_{\mathcal{M}}+4\alpha}})\right\rceil + 2p_{\mathcal{M}} + 2$. Let $V = \mathcal{W}^2\mathcal{D}^2\log(\mathcal{W}^2\mathcal{D})$. For $n \gtrsim V$, if $\lambda \gtrsim n^{\frac{2\alpha}{(p+2\alpha)(p_{\mathcal{M}}+2\alpha)}}$, then with probability at least $1-\pi$, the excess risk of $\hat{f}_{\tau,n}$ from (4) satisfies

$$\mathbb{E}_X[\{\hat{f}_{\tau,n}(X) - f^*(X)\}^2] \le C_6\left(n^{-\frac{2\alpha}{p_{\mathcal{M}}+2\alpha}}(\log n)^3\{1+\log(1/\pi)\} + \sqrt{\frac{\log(n(n-1)/\pi)}{M}}\right);$$

furthermore, if $M \gtrsim n^{\frac{4\alpha}{p_{\mathcal{M}}+2\alpha}}$, the prediction error of $\hat{f}_{\tau,n}$ satisfies

$$\mathbb{E}[\{\hat{f}_{\tau,n}(X) - f^*(X)\}^2] \le C_6' n^{-\frac{2\alpha}{p_{\mathcal{M}}+2\alpha}}(\log n)^3.$$

The above C_6, C_6' are constants which may depend on $\tau, B, \gamma, \gamma', p_{\mathcal{M}}, p, \overline{p}, N, \alpha$.

4 More extensions

4.1 *V*-statistic type estimator

The parameter τ in (4) needs to be selected by cross-validation (CV), which is a time-consuming step. With the increase in sample size, the lifetime parameter λ of the Mondrian process will increase, resulting in the tree growing deeper. In this case, the similarity of samples in the same leaf becomes higher, and the local enhancement effect should be strengthened. A natural idea is to take $\tau = 1/n$ without a cumbersome cross-validation process. Under this setting,

$$\hat{f}_n = \underset{f\in\mathcal{F}_n}{\operatorname{argmin}} \frac{1}{n^2}\sum_{k=1}^{n}\{Y_k - f(X_k)\}^2 + \frac{1}{n^2}\sum_{i\neq j} h_f(Z_i, Z_j).$$

Note that $w_\lambda(x, x) = 1$, $\hat{f}_n$ is actually an estimator of the *V*-statistic type

$$\hat{f}_n = \underset{f\in\mathcal{F}_n}{\operatorname{argmin}} \frac{1}{n^2}\sum_{i=1}^{n}\sum_{j=1}^{n}\{Y_j - f(X_i)\}^2 w_\lambda(X_i, X_j).$$

4.2 Other random forest kernel

In addition to the random forest kernel function (3), another kernel function based on random forests is

$$\tilde{w}_\lambda(x, x') = \frac{1}{M}\sum_{j=1}^{M}\frac{\mathbb{I}\{x' \in A(x, \Pi_\lambda^{(j)})\}}{N_n(x, \Pi_\lambda^{(j)})}, \tag{7}$$

where $N_n(x, \Pi_\lambda^{(j)})$ is the number of training samples falling into the leaf node $A(x, \Pi_\lambda^{(j)})$. To distinguish the two weights, we refer to (3) as the unnormalized kernel and (7) as the normalized kernel. Actually, the unnormalized kernel can be a Mercer kernel function induced by a feature map

$$\varphi(x) = \frac{1}{\sqrt{M}}\left(\varphi^{(1)}(x)^\top, \varphi^{(2)}(x)^\top, \ldots, \varphi^{(M)}(x)^\top\right)^\top,$$

where $\varphi^{(i)}(x)$ denotes a vector with the jth element being 1 and the other elements being 0 if x falls into the jth leaf of the ith Mondrian tree (assuming the leaves of the tree have been indexed). The normalized kernel possesses the expected property

$$\sum_{i=1}^{n} \tilde{w}_\lambda(x, X_i) = 1, \text{ for any given } x \in [0, 1]^p.$$

Of course, we can also use other forests such as Breiman's random forests instead of Mondrian forests to generate the aforementioned two kennel functions.

4.3 Adaptation for classification

The idea of utilizing random forests to provide local enhancement effects is not restricted to regression problems but can also be generalized to classification problems. We only need to replace the squared loss function with an appropriate loss function for classification. As an illustration, let us consider a multiclassification task with the predictor $X \in \mathbb{R}^p$ and label $Y \in \{1, 2, \ldots, K\}$. Then, the V-statistic type target function is

$$\hat{h}_n = \underset{h \in \mathcal{H}_n}{\operatorname{argmin}} \frac{1}{n^2} \sum_{i=1}^{n} \sum_{j=1}^{n} -w(X_i, X_j) \log \frac{\sum_{k=1}^{K} \exp\{h_{(k)}(X_i)\} \mathbb{I}\{Y_j = k\}}{\sum_{k=1}^{K} \exp\{h_{(k)}(X_i)\}}, \tag{8}$$

where $w(X_i, X_j)$ is a kind of random forest kernel and $\mathcal{H}_n$ is a function class of neural networks with K-dimensional output, denoted by $h = (h_{(1)}, h_{(2)}, \ldots, h_{(K)})^\top$.

4.4 Gradient learning

Gradient learning problems have received increasing attention in the statistical field. It provides valuable information about the local variation of regression function in multidimensional space. In this part, we propose two global estimation methods of the gradient function by neural networks with the help of random forests. They remain valid even when the dimension of X is relatively high. When X_j is close to X_i, the first-order Taylor expansion gives

$$f^*(X_j) \approx f^*(X_i) + \nabla f^*(X_i)^\top (X_j - X_i).$$

By this approximation, we consider the simultaneous regression and gradient estimation as

$$(\hat{f}_n, \hat{g}_n) = \underset{f \in \mathcal{F}_n, g \in \mathcal{G}_n}{\operatorname{argmin}} \frac{1}{n^2} \sum_{i=1}^{n} \sum_{j=1}^{n} \{Y_j - f(X_i) - g(X_i)^\top (X_j - X_i)\}^2 w(X_i, X_j), \tag{9}$$

where the neural network f is used to estimate the regression function f^* and g is used to estimate its gradient ∇f^*. The random forest kernel $w(X_i, X_j)$ is to weight the contributions of each datum (X_j, Y_j) to the estimation of $f^*(X_i)$ according to the proximity of X_j to X_i.

If we have acquired a well-estimated regression function $\hat{f}_{\tau,n}$ from (4), we have another option called two-stage estimation.

$$\hat{g}_n = \underset{g \in \mathcal{G}_n}{\operatorname{argmin}} \frac{1}{n(n-1)} \sum_{i \neq j} \{Y_j - \hat{f}_{\tau,n}(X_i) - g(X_i)^\top (X_j - X_i)\}^2 w(X_i, X_j). \tag{10}$$

Two valuable applications of gradient learning for sufficient dimension reduction and variable importance are consolidated in the online supplementary material.

Table 1. Notations of all variants

Method	Forest	τ	Kernel
RWN-BUU	Breiman's random forest	CV tuned value	Unnormalized
RWN-BUN	Breiman's random forest	CV tuned value	Normalized
RWN-BVU	Breiman's random forest	n^{-1}	Unnormalized
RWN-BVN	Breiman's random forest	n^{-1}	Normalized
RWN-MUU	Mondrian forest	CV tuned value	Unnormalized
RWN-MUN	Mondrian forest	CV tuned value	Normalized
RWN-MVU	Mondrian forest	n^{-1}	Unnormalized
RWN-MVN	Mondrian forest	n^{-1}	Normalized

5 Simulation

In this section, we will show the performance of our method on artificial data compared with the classical Breiman's random forest (BRF for short), Mondrian forest (MF for short), ordinary neural network (NN for short), and local linear forest (LLF for short) (Friedberg et al., 2020). As illustrated in Section 4, there are a bunch of variants of our method. For the sake of convenience, we employ the notations specified in Table 1. Taking RWN-MUU as an example, it denotes a random forest weighted neural network approach with Mondrian forest, $\tau \in (0, 1]$ (U-statistic type) and unnormalized kernel. Actually, RWN-MUU is (4).

To train two kinds of random forests, we utilized the `RandomForestRegressor` and `MondrianForestRegressor` functions from the scikit-learn library in Python. For both BRF and MF, we set the number of trees to 100, a common practice in the random forest literature (Breiman, 2001). To prevent overfitting, we tuned the `min_samples_split` parameter (representing the minimum number of samples required to split an internal node, analogous to the lifetime parameter λ for MF) within a range of 2 to 7. In particular, the BRF benefits from reducing the correlation between trees by only using a random subset of features for each node split. Therefore, we additionally tuned the `max_features` parameter in BRF using a range from 1/6 to 1, incrementing by 1/6. The best MF models obtained from the tuning process were directly employed to generate local weights for RWN variants powered by MF. But for RWN variants powered by BRF, we established separate BRF with 100 trees, `min_samples_split` set to 5, and `max_features` being 1/3. We implemented LLF by using `ll_regression_forest` function from the grf package in R. The hyperparameters of BRF used in LLF were identical to those utilized in RWN for fairness purposes. The remaining hyperparameters within the `ll_regression_forest` function remained consistent with their default values.

NNs were implemented using the Pytorch framework. We designed the number of neurons in hidden layers as $h - h/2 - h/4 - h/2$ for the fundamental neural network structure, where h is a positive integer. When the sample size is not large, choosing a relatively shallow network structure is sufficient, and the network width can be adjusted to compensate for the lack of depth. For the NN method, we tuned h from the set {64, 128, 256, 512, 1,024} (referred to as NN-OPT). In our method RWN, h is set to be 256 for training sets with fewer than 500 samples and to 512 otherwise. Furthermore, to directly reflect the performance enhancement mediated by random forest kernels, we also implemented the NN with h matching that of RWN (referred to as NN-FIX). Our training process did not incorporate regularization, instead relying on the efficient Adam optimization algorithm (i.e. `torch.optim.Adam`) with customized learning rate parameters set to 10^{-3}, while leaving other parameters at their default values. In every training iteration, a batch of 100 observations was extracted randomly from the entire training set. When observing that the total loss (5) for RWN or (1) for NN changed by less than 10^{-5} within a single batch, we determined that training shall cease; the maximum number of possible iterations was set to be 2,000. For ease of implementation, RWN selected a value for τ from $\{1, 1/(4n), 1/(2n), 1/n, 2/n, 4/n\}$ when τ needed to be tuned, where n represents the size of the training sample.

5.1 Nonparametric regression

Let us begin with regression tasks where our goal is to fit the functional relationship between input $X = (X_{(1)}, \ldots, X_{(p)})^\top \in \mathbb{R}^p$ and response $Y \in \mathbb{R}$, and to predict the response when a new input comes. Here, we focus on four typical scenarios, divided into sparse or nonsparse cases, and with or without local structure. We can obtain some insights about the advantages of our method.

- Setting 1: nonsparse case without local structure

$$Y = \frac{\log(\beta_1^\top X)}{1.5 + \sin(\pi\beta_2^\top X)} - (\beta_3^\top X)^{1/3} + \sigma\varepsilon,$$

  where $\beta_1 = (1, 1, \ldots, 1)^\top$; $\beta_2 = (1, -1, 1, -1, \ldots, 1, -1)^\top$; $\beta_3 = (2, 0, 4, 0, \ldots, p, 0)^\top$ with p being an even number.
- Setting 2: sparse case without local structure

$$Y = 2X_{(1)}X_{(2)}\sqrt{X_{(3)}^2 + X_{(4)}} - X_{(5)}e^{X_{(6)}} + \sigma\varepsilon.$$

- Setting 3: nonsparse case with local structure

$$Y = \sum_{j=1}^{p}\left(\sqrt{X_{(j)}(1 - X_{(j)})} - \mathbb{I}\{X_{(j)} < 0.5\}\right) + \sigma\varepsilon.$$

- Setting 4: sparse case with local structure

$$Y = (1 + X_{(1)} - X_{(2)} - 2X_{(3)})^3 \cdot \mathbb{I}\{X_{(1)} - X_{(2)} > 0\} + \sigma\varepsilon.$$

Here, $X \sim \text{Uniform}([0, 1]^p)$ and $\varepsilon \sim \text{Normal}(0, 1)$. It is worth noting that the value of σ was determined based on the signal-to-noise ratio (SNR), which quantifies the strength of a regular signal (i.e. the conditional mean) relative to the irregular noise in a model. Generally, models with higher SNR values are deemed to be more manageable. To be more specific, we adopted the approach of Cheng and Wu (2013) to define SNR as follows

$$\text{snrdb} = 10\log_{10}\left(\frac{\text{Var}(Y)}{\text{Var}(\varepsilon)}\right).$$

For all aforementioned settings, we set snrdb = 5, which corresponds to a moderate level of SNR. Next, we utilized bootstrap sampling to estimate $\text{Var}(Y)$ and inferred the value of σ according to the definition of snrdb. The training sample size n varied from 200 to 1,000 with different choices of p. In addition, we drew 1,000 sample examples for testing and calculated the mean squared error (MSE for short) between true response values (without noise) and predicted values. The procedure was then run 100 times to reduce randomness.

Figure 1 displays the results of MSE for each method under $n = 200$ and $p = 10$. For our random forest weighted approach (RWN), we report the results of RWN-MUU, alongside the best performance among all the eight variants (referred to as RWN-OPT). It can be observed that when

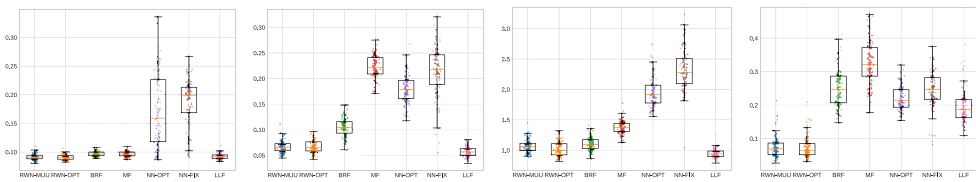

Figure 1. The MSEs of different methods in Settings 1–4 (from left to right) under $n = 200$ and $p = 10$.

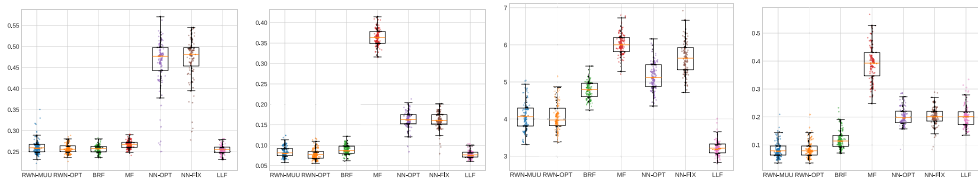

Figure 2. The MSEs of different methods in Settings 1–4 (from left to right) under $n = 500$ and $p = 30$.

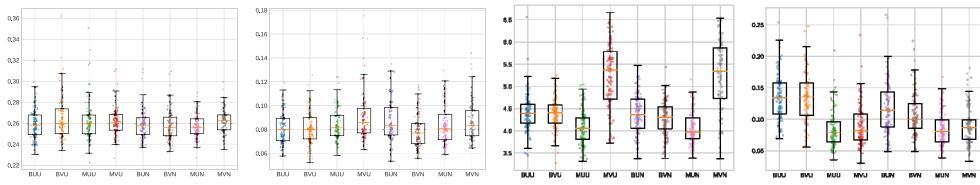

Figure 3. The MSEs of different variants of RWN in Settings 1–4 (from left to right) under $n = 500$ and $p = 30$.

compared to NN-FIX, RWNs (RWN-MUU and RWN-OPT) are more accurate and stable due to the much lower variance, thus demonstrating the benefits of incorporating weights generated by random forest. Even compared to the well-tuned neural network (NN-OPT) and two kinds of forests (BRF and MF), RWNs show notable advantages. We also note that the performance of RWN-MUU closely resembles that of RWN-OPT, indicating that RWN-MUU is a good candidate variant in practice. In both Setting 1 and Setting 2, RWNs and LLF exhibit similar performance and outperform the other methods. Moreover, LLF demonstrates superior capability in handling the task in Setting 3, while RWNs showcase their dominant strength in Setting 4. LLF, as a local regression method, employs random forests to address the dimensionality limitation problem of local linear regression. In contrast, our RWN method is a global regression method that utilizes random forests to enhance the accuracy and stability of neural networks, especially in the case of limited samples. Despite their distinct initial intentions, the result showcases the effectiveness of leveraging random forest weights to improve both methods. Similar patterns are observed under $n = 500$ and $p = 30$ (see Figure 2). The comprehensive results can be found in online supplementary material Table 1.

As a next step, we proceeded to examine the variation among eight variants of RWN. Under $n = 500$ and $p = 30$, Figure 3 visually demonstrates the proximity of results among eight variants in Setting 1 and Setting 2, whereas RWN-MUU and RWN-MUN, corresponding to Mondrian forest weights of primary interest in this article, exhibit superior performance in Setting 3 and Setting 4. While the choice of variants is highly flexible, determining the optimal variant in practice may necessitate testing all combinations, a strategy sometimes not easily embraced. To gain more insight, we computed the rankings of all methods (including all variants of RWN) in each setting (see online supplementary material Table 2). The resulting average rankings indicate that RWN-MUN emerges as a potentially optimal choice. Moreover, the theoretically covered variant, RWN-MUU, also proves to be a strong contender, outperforming individual RF and NN in these numerical simulations, aligning with our expectations. Hence, we recommend using RWN-MUU or other better variants for practical purposes. Fréchet

In the former study, we investigated the performances of RWN under various settings across local structures, the number of observations and the number of features. It can be found that RWN outperformed NN obviously. Another factor influencing the usability of a method is the level of noise. As Mentch and Zhou (2020) claimed, random forests are particularly applicable in moderate and low SNR cases due to their implicit regularization. Therefore, it is of interest to investigate the relative performances of RWN across various SNRs. Recall that a higher SNR indicates weaker perturbation. We modified the aforementioned Setting 4 by specifying three cases of snrdb =1, 5 and 10. The results are displayed in Figure 4 (see Table 3 in online supplementary material for numerical results). In Setting 4, under moderate and high SNRs, RWNs

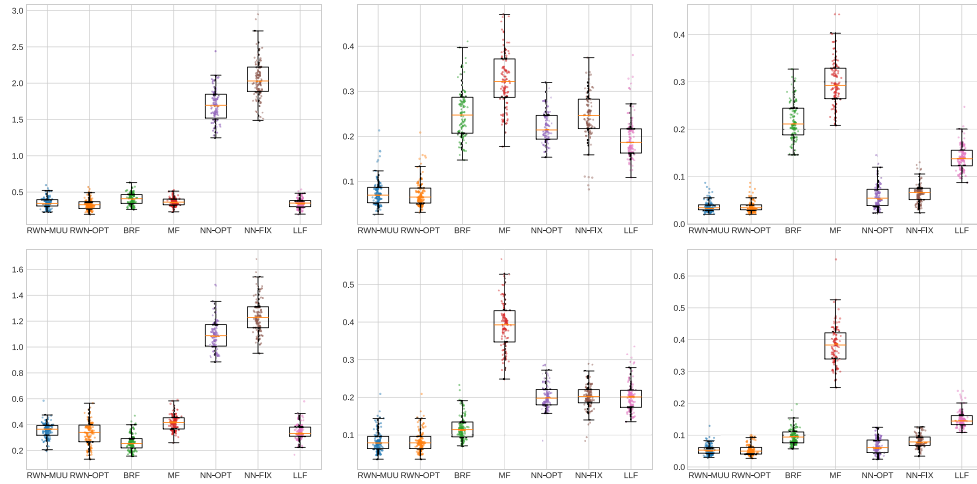

Figure 4. The MSEs of different methods across snrdb =1, 5, 10 (from left to right) in setting 4 with $n = 200$, $p = 10$ (top three panels) and $n = 500$, $p = 30$ (bottom three panels).

(RWN-MUU and RWN-OPT) demonstrate exceptional performances, with NNs (NN-OPT and NN-FIX) surpassing random forests merely in the high SNR cases. When adding extreme noise (i.e. snrdb =1), random forest based methods show their superiority. This shift highlights the robustness of random forests to noise. In particular, RWNs, leveraging local information from the random forest, are more competitive than NNs in this challenging scenario. It indicates that RWNs have acquired some noise resistance capability from RF. LLF yields comparable results with RWNs when snrdb =1 while its performance is not satisfied at moderate and high values of snrdb. In summary, the method RWN is fairly robust to noise, which positions it as a versatile choice across a wider range of noise scenarios.

5.2 Gradient estimation

We then turn to gradient estimation with simulations. We choose to estimate the target function together with its gradients simultaneously (see (9)), or in a two-stage manner (see (10)). The following settings are considered.

- Setting 5: discontinuous case

$$Y = (1 + X_{(1)} - X_{(2)} - 2X_{(3)})^3 \cdot \mathbb{I}\{X_{(1)} > X_{(2)}\} + \sigma\varepsilon.$$

- Setting 6: continuous case

$$Y = (1 + X_{(1)} - X_{(2)} - 2X_{(3)})^2 X_{(2)} - 2\sin(\pi X_{(1)}) e^{-X_{(4)} - X_{(5)}} + \sigma\varepsilon.$$

Here, $X \sim \text{Uniform}([0, 1]^p)$, $\sigma = 0.1$, $\varepsilon \sim \text{Normal}(0, 1)$, $n = 500$ and $p = 20$.

To generate random forest weights, we configured Breiman's random forest with 100 trees, setting `min_sample_split` to 5 and `max_features` to 1; the hyperparameters in Mondrain forest were tuned through cross-validation, same as the regression simulation part. We terminated the training process when the total loss change was less than 10^{-8}. Here, we merely implemented the V-statistic type variants for simplicity. From Figure 5, it is clear that Breiman's random forest weighted variants could precisely learn the derivatives, while Mondrian forests failed. Additionally, all of Breiman's random forest weighted variants produced similar performances. This reveals that the CART splitting criterion that utilizes information of response variables can substantially improve the accuracy of derivative estimation. To get a more

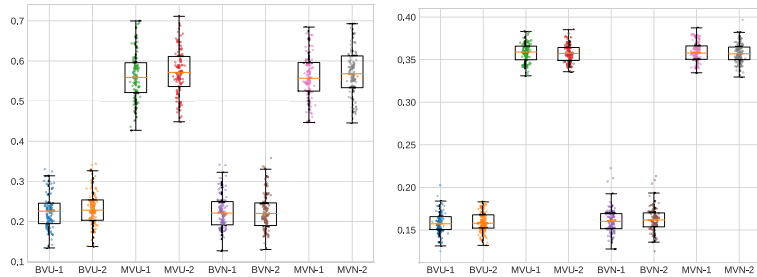

Figure 5. The MSEs between true derivatives and predicted derivatives of different variants. BVU-1 means that we learn f and g simultaneously via RWN-BVU; BVU-2 means that we learn f via RWN-BVU and then learn g with $\hat{f}_n$ plugged in, hereafter. Left panel: Setting 5; right panel: Setting 6.

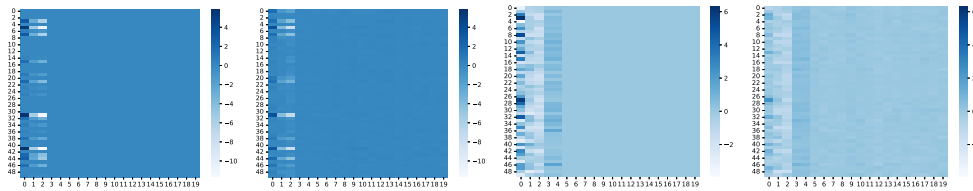

Figure 6. The heatmaps of true derivatives and predicted derivatives on 50 random samples. From left to right are the true values in Setting 5, predicted values in Setting 5, true values in Setting 6, and predicted values in Setting 6.

intuitive impression, we randomly generated 50 testing samples for two settings and plotted the heatmaps of the true derivatives with the predicted derivatives based on the best variant. Figure 6 shows that true values and predicted values were in the same patterns. Notably, the first three variables were most important for setting 5, and the first five variables greatly accounted for the association between X and Y in Setting 6, just as specified in the settings. Such significance can further be quantified through the variable importance score introduced in online supplementary material.

6 Real data

In this section, we compare the RWN method with the ordinary neural network, two kinds of random forests, neural random forest (NRF for short) (Biau et al., 2019) and local linear forest (LLF for short) (Friedberg et al., 2020). The deep forest (Zhou & Feng, 2019), designed primarily for complex classification tasks such as face recognition and music classification, is not included in the comparison here. We collected 10 real datasets from the UCI Machine Learning Repository and other sources. The datasets involved are aquatic toxicity, Boston housing prices, forest fires, concrete compressive strength, relative location of CT slices on axial axis for regression benchmark; seismic bumps, spambase, diabetic retinopathy debrecen, ozone level detection, and insurance company for extensional classification benchmark. Their sample sizes vary from 500 to 50,000, and predictor dimensions increase from 8 to nearly 400.

For each dataset, rows with missing values were removed for convenience. The whole dataset was split into the training set (70%) and the testing set (30%). Then, we normalized each feature among the training set and the same processing was conducted on the testing set (using function `StandardScaler` in scikit-learn). We trained all methods on the training set and evaluated testing performances on the testing set. The hyper-parameters were selected via cross-validation. To obtain reliable results, the training and evaluation procedures were replicated 100 times, except for the CT dataset with 10 repetitions, as the neural random forest took a long training time for this large volume of data. Note that neural random forest and local linear forest are designed for regression, and are thus not included in classification tasks.

From Table 2, it is observable that RWN (RWN-MUU and RWN-OPT) achieved sound performance. For the toxicity and forest fires datasets, LLF performed the best, and RWN produced

Table 2. The averages (standard deviations) of mean squared errors (regression) or accuracy (classification) for each method for real data. Bold-faced numbers indicate the best performers.

Dataset	n	p	Task	RWN-MUU	RWN-OPT	BRF	MF	NN-OPT	NN-FIX	NRF	LLF
Toxicity	546	8	Regression	1.374 (0.244)	1.345 (0.191)	1.399 (0.185)	1.260 (0.159)	1.758 (0.316)	1.867 (0.331)	1.344 (0.286)	**1.234** (0.154)
Housing	506	13	Regression	11.490 (3.538)	**11.189** (3.796)	12.690 (4.290)	14.999 (4.925)	11.409 (3.175)	12.487 (4.103)	11.999 (4.888)	11.922 (3.238)
Fires	517	43	Regression	1.955 (0.221)	1.955 (0.221)	2.057 (0.227)	1.950 (0.201)	3.292 (0.356)	3.476 (0.442)	2.027 (0.289)	**1.934** (0.214)
Concrete	1,030	8	Regression	25.916 (3.905)	25.756 (3.944)	26.030 (3.397)	64.825 (6.796)	26.992 (3.673)	26.006 (4.119)	**21.636** (4.420)	33.358 (4.216)
CT	53,500	384	Regression	**0.504** (0.189)	**0.504** (0.189)	2.479 (0.328)	2.820 (0.185)	0.573 (0.221)	0.785 (0.357)	18.127 (1.300)	0.531 (0.049)
Diabetic	1,151	19	Classification	–	**70.5%** (0.036)	67.7% (0.023)	67.9% (0.021)	70.4% (0.024)	69.7% (0.022)	–	–
Seismic	2,584	24	Classification	–	**93.5%** (0.007)	93.0% (0.008)	92.6% (0.008)	91.3% (0.012)	90.6% (0.010)	–	–
Spambase	4,601	57	Classification	–	93.9% (0.007)	**95.2%** (0.005)	94.5% (0.005)	94.1% (0.007)	94.1% (0.006)	–	–
Ozone	1,848	72	Classification	–	**96.8%** (0.006)	96.7% (0.006)	**96.8%** (0.006)	96.0% (0.007)	96.1% (0.007)	–	–
Insurance	5,822	85	Classification	–	**94.0%** (0.001)	93.1% (0.002)	92.6% (0.001)	91.9% (0.008)	91.2% (0.009)	–	–

a performance on par with it. For the housing dataset, the performance of LLF was modest, NN-OPT showed its advantage, and RWN-OPT provided even better precision. For the concrete dataset, NRF and RWN are prominent contenders. For the CT dataset, RWN showed its superiority, with LLF and NN-OPT slightly falling behind. Additionally, for classification tasks, we only consider the *V*-statistic type of RWN (see (8); thus RWN-MUU has no output results). RWN-OPT also demonstrates its competitiveness compared to random forests and neural networks. These results reflect that RWN simultaneously combines the superiorities of random forests and neural networks. RWN is more likely an improvement of neural networks, while it can utilize the information of random forest weights to strengthen its power. LLF is a strong contender, and it is the only local regression method. The method NRF also produced impressive results better than BRF on most datasets, except for the last one in the regression scenario. For large datasets, the random forest used for neural network initialization in NRF could not grow deep enough due to our limitation of computer memory. It is potential that insufficient initialization along with the shallow structure of NRF itself is the main cause for this exception. Moreover, we found in the experiments that NRF was rather time-consuming compared to the other methods.

7 Discussion

The neural network is a popular and promising nonparametric method, but it also has some shortcomings, such as moderate performance in small sample sizes, mediocre interference immunity in noisy environments and inferior ability to capture local information buried in the data. On the contrary, the traditional machine learning method, random forest, can compensate for these weaknesses effectively. And the superior property of random forests being suitable for relatively high-dimensional inputs allows them to cooperate with neural networks. In this article, we propose to enhance neural networks with random forests. Combining global and local methods enables us to extract information from the data in a more comprehensive way and facilitates statistical learning.

The main focus of this article is the statistical analysis of tabular data. It would be interesting to extend our approach to cases where X is a matrix or tensor. Constructing a random forest that can adapt to these types of inputs is critical, but it is beyond the scope of this article. We leave this for future research.

Acknowledgments

The authors are grateful to the Editor, the Associate Editor, and the reviewers for their valuable discussions and comments on earlier versions of this article. The research is supported by the National Key R&D Program of China (Grant No. 2023YFA1008700 and 2023YFA1008703), the National Natural Science Foundation of China (Grant No. 12371289), and the Shanghai Pilot Program for Basic Research (Grant No. TQ20220105).

Conflict of interest: None declared.

Data availability

The aquatic toxicity dataset, forest fires dataset, concrete compressive strength dataset, relative location of CT slices dataset, diabetic retinopathy debrecen dataset, seismic bumps dataset, spambase dataset, ozone level detection dataset, and insurance company dataset are available in the UCI Machine Learning Repository, at https://archive.ics.uci.edu/datasets; the Boston housing prices dataset is available in the StatLib Datasets Archive, at https://lib.stat.cmu.edu/datasets/boston. Source code of our method is available at https://github.com/oaksword/RWN.

Supplementary material

Supplementary material is available online at *Journal of the Royal Statistical Society: Series B*.

References

Arnould L., Boyer C., & Scornet E. (2021). Analyzing the tree-layer structure of deep forests. In *Proceedings of the 37th International Conference on Machine Learning* (pp. 342–350). PMLR.

Bartlett P. L., Bousquet O., & Mendelson S. (2005). Local Rademacher complexities. *Annals of Statistics*, *33*(4), 1497–1537. https://doi.org/10.1214/009053605000000282

Bartlett P. L., Harvey N., Liaw C., & Mehrabian A. (2019). Nearly-tight VC-dimension and pseudodimension bounds for piecewise linear neural networks. *Journal of Machine Learning Research*, *20*(63), 1–17. https://doi.org/10.48550/arXiv.1703.02930

Bauer B., & Kohler M. (2019). On deep learning as a remedy for the curse of dimensionality in nonparametric regression. *Annals of Statistics*, *47*(4), 2261–2285. https://doi.org/10.1214/18-AOS1747

Biau G. (2012). Analysis of a random forests model. *Journal of Machine Learning Research*, *13*(1), 1063–1095. https://doi.org/10.48550/arXiv.1005.0208

Biau G., Devroye L., & Lugosi G. (2008). Consistency of random forests and other averaging classifiers. *Journal of Machine Learning Research*, *9*(9), 2015–2033. https://doi.org/10.5555/1390681.1442799

Biau G., Scornet E., & Welbl J. (2019). Neural random forests. *Sankhya A*, *81*(2), 347–386. https://doi.org/10.1007/s13171-018-0133-y

Breiman L. (2001). Random forests. *Machine Learning*, *45*(1), 5–32. https://doi.org/10.1023/A:1010933404324

Chen M., Jiang H., Liao W., & Zhao T. (2022). Nonparametric regression on low-dimensional manifolds using deep ReLU networks: Function approximation and statistical recovery. *Information and Inference: A Journal of the IMA*, *11*(4), 1203–1253. https://doi.org/10.1093/imaiai/iaac001

Chen X., & Ishwaran H. (2012). Random forests for genomic data analysis. *Genomics*, *99*(6), 323–329. https://doi.org/10.1016/j.ygeno.2012.04.003

Cheng M.-Y., & Wu H.-T. (2013). Local linear regression on manifolds and its geometric interpretation. *Journal of the American Statistical Association*, *108*(504), 1421–1434. https://doi.org/10.1080/01621459.2013.827984

Chi C.-M., Vossler P., Fan Y., & Lv J. (2022). Asymptotic properties of high-dimensional random forests. *Annals of Statistics*, *50*(6), 3415–3438. https://doi.org/10.1214/22-AOS2234

Denil M., Matheson D., & De Freitas N. (2014). Narrowing the gap: Random forests in theory and in practice. In *Proceedings of the 31st International Conference on Machine Learning* (pp. 665–673). PMLR.

Fan J., & Gijbels I. (2018). *Local polynomial modelling and its applications*. Routledge.

Farrell M. H., Liang T., & Misra S. (2021). Deep neural networks for estimation and inference. *Econometrica*, *89*(1), 181–213. https://doi.org/10.3982/ECTA16901

Friedberg R., Tibshirani J., Athey S., & Wager S. (2020). Local linear forests. *Journal of Computational and Graphical Statistics*, *30*(2), 503–517. https://doi.org/10.1080/10618600.2020.1831930

Gao W., & Zhou Z.-H. (2020). Towards convergence rate analysis of random forests for classification. In *Proceedings of the 34th International Conference on Neural Information Processing Systems* (pp. 9300–9311). Curran Associates, Inc.

Hinton G. E., & Salakhutdinov R. R. (2006). Reducing the dimensionality of data with neural networks. *Science*, *313*(5786), 504–507. https://doi.org/10.1126/science.1127647

Hoeffding W. (1948). A class of statistics with asymptotically normal distribution. *Annals of Mathematical Statistics*, *19*(3), 293–325. https://doi.org/10.1214/aoms/1177730196

Jiao Y., Shen G., Lin Y., & Huang J. (2023). Deep nonparametric regression on approximate manifolds: Nonasymptotic error bounds with polynomial prefactors. *Annals of Statistics*, *51*(2), 691–716. https://doi.org/10.1214/23-AOS2266

Klusowski J. (2021). Sharp analysis of a simple model for random forests. In *Proceedings of The 24th International Conference on Artificial Intelligence and Statistics* (pp. 757–765). PMLR.

Klusowski J. M., & Tian P. M. (2024). Large scale prediction with decision trees. *Journal of the American Statistical Association*, *119*(545), 525–537. https://doi.org/10.1080/01621459.2022.2126782

Kohler M., Krzyżak A., & Langer S. (2019). Estimation of a function of low local dimensionality by deep neural networks. *IEEE Transactions on Information Theory*, *68*(6), 4032–4042. https://doi.org/10.1109/TIT.2022.3146620

Kontschieder P., Fiterau M., Criminisi A., & Bulo S. R. (2015). Deep neural decision forests. In *Proceedings of the 2015 IEEE International Conference on Computer Vision* (pp. 1467–1475). AAAI Press.

Lakshminarayanan B., Roy D. M., & Teh Y. W. (2014). Mondrian forests: Efficient online random forests. In *Proceedings of the 27th International Conference on Neural Information Processing Systems* (pp. 3140–3148). MIT Press.

Lin Y., & Jeon Y. (2006). Random forests and adaptive nearest neighbors. *Journal of the American Statistical Association*, *101*(474), 578–590. https://doi.org/10.1198/016214505000001230

Mentch L. & Hooker G. (2016). Quantifying uncertainty in random forests via confidence intervals and hypothesis tests. *Journal of Machine Learning Research*, *17*(1), 841–881. https://doi.org/10.48550/arXiv.1404.6473

Mentch L., & Zhou S. (2020). Randomization as regularization: A degrees of freedom explanation for random forest success. *Journal of Machine Learning Research*, *21*(1), 6918–6953. https://doi.org/10.48550/arXiv.1911.00190

Mourtada J., Gaïffas S., & Scornet E. (2020). Minimax optimal rates for Mondrian trees and forests. *Annals of Statistics*, *48*(4), 2253–2276. https://doi.org/10.1214/19-AOS1886

Peng W., Coleman T., & Mentch L. (2022). Rates of convergence for random forests via generalized U-statistics. *Electronic Journal of Statistics*, *16*(1), 232–292. https://doi.org/10.1214/21-EJS1958

Schmidt-Hieber J. (2020). Nonparametric regression using deep neural networks with ReLU activation function. *Annals of Statistics*, *48*(4), 1875–1897. https://doi.org/10.1214/24-aos2351

Scornet E. (2016). Random forests and kernel methods. *IEEE Transactions on Information Theory*, *62*(3), 1485–1500. https://doi.org/10.1109/TIT.2016.2514489

Scornet E., Biau G., & Vert J.-P. (2015). Consistency of random forests. *Annals of Statistics*, *43*(4), 1716–1741. https://doi.org/10.1214/15-AOS1321

Shen Z., Yang H., & Zhang S. (2019). Nonlinear approximation via compositions. *Neural Networks*, *119*, 74–84. https://doi.org/10.1016/j.neunet.2019.07.011

Shen Z., Yang H., & Zhang S. (2020). Deep network approximation characterized by number of neurons. *Communications in Computational Physics*, *28*(5), 1768–1811. https://doi.org/10.4208/cicp

Shen Z., Yang H., & Zhang S. (2022). Optimal approximation rate of ReLU networks in terms of width and depth. *Journal de Mathématiques Pures et Appliquées*, *157*, 101–135. https://doi.org/10.1016/j.matpur.2021.07.009

Stone C. J. (1982). Optimal global rates of convergence for nonparametric regression. *Annals of Statistics*, *10*(4), 1040–1053. https://doi.org/10.1214/aos/1176345969

Vaart A. W., & Wellner J. A. (1996). *Weak convergence and empirical processes*. Springer.

Wager S., & Athey S. (2018). Estimation and inference of heterogeneous treatment effects using random forests. *Journal of the American Statistical Association*, *113*(523), 1228–1242. https://doi.org/10.1080/01621459.2017.1319839

Yarotsky D. (2017). Error bounds for approximations with deep ReLU networks. *Neural Networks*, *94*, 103–114. https://doi.org/10.1016/j.neunet.2017.07.002

Yarotsky D. (2018). Optimal approximation of continuous functions by very deep ReLU networks. In *Proceedings of the 31st Conference on Learning Theory* (pp. 639–649). PMLR.

Zhou Z.-H., & Feng J. (2019). Deep forest. *National Science Review*, *6*(1), 74–86. https://doi.org/10.1093/nsr/nwy108