Near-optimal Sparse Neural Trees

Tanujit Chakraborty

Abstract Decision tree algorithms have been among the most popular algorithms for interpretable (transparent) machine learning since the early 1980s. On the other hand, deep learning methods have boosted the capacity of machine learning algorithms and are now being used for non-trivial applications in various applied domains. But training a fully-connected deep feed-forward network by gradient-descent backpropagation is slow and requires arbitrary choices regarding the number of hidden units and layers. In this paper, we propose near-optimal neural regression trees, intending to make it much faster than deep feed-forward networks and for which it is not essential to specify the number of hidden units in the hidden layers of the neural network in advance. The key idea is to construct a decision tree and then simulate the decision tree with a neural network. This work aims to build a mathematical formulation of neural trees and gain the complementary benefits of both sparse optimal decision trees and neural trees. We propose near-optimal sparse neural trees (NSNT) that is shown to be asymptotically consistent and robust in nature. Additionally, the proposed NSNT model obtain a fast rate of convergence which is near-optimal upto some logarithmic factor. We comprehensively benchmark the proposed method on a sample of 80 datasets (40 classification datasets and 40 regression datasets) from the UCI machine learning repository. We establish that the proposed method is likely to outperform the current state-of-the-art methods (random forest, XGBoost, optimal classification tree, and near-optimal nonlinear trees) for the majority of the datasets.

Keywords Decision trees · Deep feed-forward network · Neural trees · Consistency · Optimal rate of convergence

Corresponding Author: Tanujit Chakraborty
International Institute of Information Technology, Bangalore, India.
E-mail: tanujit_r@isical.ac.in

1 Introduction

Decision trees [8] and deep feed-forward neural networks [23] are very popular nonparametric prediction models due to their elegant mathematical basis and ability to model both linear and non-linear decision boundaries. Since decision trees are rule-based, when small-sized, they are deemed to be leaders in terms of interpretability whereas neural networks perform superior in terms of out-of-sample predictability, but sometimes lack mathematical interpretability due to having multiple hidden layers (usually unknown). Generally, these networks have no built-in hierarchy and consequently are fully connected, viz. all neurons in a layer are connected to all the neurons in the adjacent layers [34]. Since the number of neurons in the hidden layers is not known apriori, one designs a network with a varying number of neurons in different layers to determine the best suitable architecture for a given problem. As a result of these limitations, a considerable amount of design and training time is needed in many situations, and even then one is not sure that the *'optimal'* design has been achieved. It is also well-known that the problem of building optimal neural networks is NP-complete [7].

To overcome these drawbacks, a tree-structured hybrid representation of the neural network was proposed in the previous literature [48][41][44]. The main idea behind neural trees is to construct a decision tree and then simulate the decision tree with a neural net [10]. Neural trees are composed of three basic steps – (a) converting a tree into rules, (b) constructing a neural network from the rules, and (c) training the neural net. The main motivation behind converting the tree into a rule set is that it allows distinguishing among different contexts in which a decision node is used. The significance level of each node is determined in terms of weights trained by the neural network model as follows. The antecedents of a rule are used as input features that are linked to the hidden unit(s) which represent the rule. Thus, the number of hidden units in the network is the same as the total number of leaf nodes obtained from the tree-based algorithm. All the hidden units and output units include bias weights and network weights. These weights are further trained using gradient backpropagation algorithm [39]. Training multilayer neural networks by backpropagation is slow and requires arbitrary choices about the number of hidden units and layers. Neural trees are much faster and for which it is not necessary to specify the number of hidden units in advance [42]. To design neural trees for a given problem, one first develops a decision tree which is then transformed into a three-layered structure following a set of simple rules. Decision trees can be automatically designed using a set of input-output mapping pairs. Thus, the model can self-configure the architecture for a given problem. This is very important as using the proper number of neurons in the hidden layers affects the training time and the prediction performance [42][13]. The model has been applied to solve classification and regression problems [44][47][38], and various modification of the algorithm can be found in previous literature [40][43][50][16][19][25][46]. However, most of these methods are non-scalable, and their mathematical formulations are not aptly done. From a theoretical point of view, the literature on neural trees is less conclusive. Regardless of the use of the model in applied problems of classification and regression, asymptotic results are yet to be proved. This creates a gap between theory and practice. To the best of our knowledge, there is no optimal (near or sub) set-up available for the neural tree. However, in the current literature of decision trees, there are recent works that introduced optimal classification trees [3][49][4] and sparse optimal decision trees [24][30][6] which are necessarily a strong motivation behind the current work.

The aim of this work is to *introduce a mathematical formulation of near-optimal neural trees* and gain the complementary benefits of both sparse optimal decision trees and neural trees. To this end, we propose *near-optimal sparse neural trees* (NSNT), which generalize and address the limitations of the previous works [44][41][19][6] that attempted the same unification. We aim to make the proposed model *scalable* (the size of the data does not pose a problem), *robust* (work well in a wide variety of problems in the presence of noisy samples), *accurate* (achieve higher predictive accuracy), *statistically sound* (have desired asymptotic properties), and *interpretable* for its effective implementation in real-world classification and regression problems. We strongly desire that, whilst achieving competitive performance on real-world datasets, NSNT would benefit from (i) lightweight inference via conditional computation, (ii) hierarchical separation of features useful to the neural network building stage, and (iii) a mechanism to adapt the architecture to the size and complexity of the training dataset. We further investigate *asymptotic consistency* and *rate of convergence* for the theoretical robustness of the proposed NSNT model.

The most celebrated theoretical results in the field of decision trees and neural networks have given the general sufficient conditions for almost-sure L_2-consistency of data-driven density estimates [31] and consistency for feed-forward neural network estimates [32], respectively. Universal approximation properties for two hidden layered neural networks with a bounded number of neurons in hidden layers are proved [26][20]. In recent work, least-square estimates based on deep feed-forward neural networks are shown to circumvent the curse of dimensionality in nonparametric regression [2]. Motivated by these works, our current study proves the strong consistency of NSNT model, which gives a basic theoretical guarantee for its effectiveness in practical studies. The approach depends on the choice of the total number of leaves and certain restrictions imposed on neural network hyper-parameters to ensure the consistency of the model. We discuss an analysis of the algorithmic complexity of the model along with the fast rate of convergence of the model. More interestingly, a general bound on the expected L_2 error of adaptive least square estimates is established and applied to the regression function estimates of the proposed model to obtain the rate of convergence in the case of additive regression functions. The rate of convergence for the model is shown to be near-optimal up to some logarithmic factor according to [45].

To show the practical utility of the proposal to machine learning practitioners, we comprehensively benchmark NSNT against the state-of-the-art models on a sample of 80 datasets from the UCI machine learning repository.

We show that across this sample, proposed NSNT perform consistently well for datasets with sizes in the thousands and yield higher out-of-sample accuracy than the random forest [9], XGBoost [14], optimal classification tree [3] and near-optimal nonlinear trees [4] on an average. The application of the proposal is demonstrated with several standard classification and regression data sets of various sizes. An implementation of the proposed NSNT model is made available for public use at `https://github.com/tanujit123/NSNT`.

2 Formulation of Proposed NSNT Model

In this section, we describe how a pre-trained decision tree can be reformulated as a two hidden-layered (2HL) deep feed-forward neural network with similar types of predictive behavior [48][41][10][11]. Suppose we are given a training sample $D_n = \{(X_1, Y_1), (X_2, Y_2), ..., (X_n, Y_n)\}$ with n observations on p independent variables. Consider a nonparametric regression framework in which p input features $X \in C^p = [0,1]^p$ are observed and we predict a square integrable output function $Y \in R$. A decision tree can be viewed as a regression function estimate using hierarchical axes-parallel splits of the feature space. Each tree node must correspond to one of the segmentation subsets available in C^p. For simplicity and easy interpretability, let us consider ordinary binary decision tree where a node has exactly two child nodes or zero child nodes (leaf nodes). Tree consists of split nodes (for example, $x^{(i)} \geq \alpha$ for some $i \in \{1, 2, ..., p\}$ and some $\alpha \in C$) and leaf nodes. The feature space C^p is partitioned into axes-parallel hyper-rectangles. The standard splitting criteria, MSE (mean squared error) is used to create the decision tree. While making prediction, the input vector is passed into root node of the decision tree and iteratively transmitted further to the leaf node which belongs to the subspace where the input is located; this is repeated until a leaf node is finally reached. If we suppose that a leaf represents region S ($S \subseteq C^p$), then the natural regression function estimate can be written in the following mathematical form:

$$t_n(x) = \frac{\sum_{i=1}^{n} Y_i \cdot \mathbb{1}_{X_i \in S}}{N_n(S)}, \; x \in S$$

where $N_n(S)$ is defined as the number of observations in cell S (by convention, we assume $\frac{0}{0} = 0$). We obtain the predicted result for a query point x in leaf node S as an average of the Y_i's of all training samples which fall into the region S. Below we present a formal description of the decision tree to be used in the proposed model.

2.1 Background: Regression Trees

To be specific, we consider the decision tree by [8], also named as *classification and regression tree* (CART) in our case. The main idea is to form a tree having k leaf regions (k depends on n) defined by a partition of the p-dimensional feature space with n observations. In the construction of the tree, the so-called CART-split criterion (MSE for regression set up) is applied recursively. This criterion helps in determining the input direction for the split and also for finding the appropriate cut. A formal mathematical expression for the decision tree algorithm based on [8] is as follows.

We assume S to be a generic cell and $N_n(S)$ to be the number of observations falling in S. Then a cut in S is a pair (i, α), where $i \in \{1, 2, ..., p\}$ and $\alpha \in [0, 1]$ is the position of the cut along the i-th coordinate, within the limits of S. Let $\mathcal{P}_S$ be the set of all such possible cuts in S. Then, with the notation $X_j = \left(X_j{}^{(1)}, \cdots, X_j{}^{(p)}\right)$, the CART-split criterion for the decision tree takes the following form,

$$\begin{aligned} L_n(i,\alpha) = & \frac{1}{N_n(S)} \sum_{j=1}^{n} \left(Y_j - \bar{Y}_S\right)^2 \mathbb{1}_{X_j \in S} \qquad (1) \\ & - \frac{1}{N_n(S)} \sum_{j=1}^{n} \left(Y_j - \bar{Y}_{S_L} \mathbb{1}_{X_j{}^{(i)} < \alpha} - \bar{Y}_{S_R} \mathbb{1}_{X_j{}^{(i)} \geq \alpha}\right)^2 \mathbb{1}_{X_j \in S}, \end{aligned}$$

for any $(i, \alpha) \in \mathcal{P}_S$, where $S_L = \{x \in S, x^{(i)} < \alpha\}$, $S_R = \{x \in S, x^{(i)} \geq \alpha\}$, and $\bar{Y}_S$ represents the average of the Y_j belonging to S with the convention $\frac{0}{0} = 0$. Intuitively, $L_n(i, \alpha)$ measures the (re-normalized) difference between the empirical variance in the node before and after a cut is performed. Specifically, the best cut (i_n^*, α_n^*) for each cell S is selected by maximizing $L_n(i, \alpha)$ over $\mathcal{P}_S$, viz.,

$$(i_n^*, \alpha_n^*) \in \underset{(i,\alpha)\in\mathcal{P}_S}{\operatorname{argmax}} L_n(i, \alpha).$$

At each cell, the decision tree model evaluates the criterion (1) over all possible cuts in the p directions and returns the best possible cut. In case of ties, the best cut is usually selected in the middle of the two consecutive data points. This process is recursively continued until the tree exactly contains k terminal nodes, where $k \geq 2$ is an integer eventually depends on n (k and k_n have the same meaning).

2.2 Near-optimal Sparse Neural Trees

Assuming that we have a decision tree t_n (whose construction eventually depends upon the data D_n) at hand, which takes constant values on each of $k \geq 2$ terminal nodes. It turns out that these estimate can be reinterpreted as two hidden-layer neural networks, as summarized below. Let $HL_1 = \{H_1, \ldots, H_{k-1}\}$ denote the collection of all hyperplanes participating in the construction of t_n. Each $H_{k'} \in HL_1$ is of the form $H_{k'} = \{x \in C^p : h_{k'}(x) = 0\}$, where $h_{k'}(x) = x^{(i_{k'})} - \alpha_{i_{k'}}$ for some (eventually data-dependent) $i_{k'} \in \{1, 2, ..., p\}$ and $\alpha_{i_{k'}} \in C$. To reach the leaf of the query point x, one finds the side on which x falls ($+1$ for right and -1 for left) for each hyperplane $H_{k'}$. Using the above notations, the tree estimate t_n is mapped to the neural network as discussed below.

Designing the first hidden layer (HL_1): The input layer supplies the features to the first hidden layer of neurons which corresponds to $k-1$ perceptrons, with the threshold activation function defined as $\tau(h_{k'}(x)) = \tau(x^{(i_{k'})} - \alpha_{i_{k'}})$, where $\tau(u) = 2\mathbb{1}_{u\geq 0} - 1$. Therefore, for each split in the tree, there is a neuron in HL_1 whose activity encodes the relative position of an input x with respect to the concerned split. The output of the first layer is ± 1-vector $(\tau(h_1(x)), \ldots, \tau(h_{k-1}(x)))$, which describes all decisions of the inner tree nodes (it also includes the nodes off the tree path of x). Note that $\tau(h_{k'}(x))$ takes the value $+1$ if x is on one side of the hyperplane $H_{k'}$ and -1 if x is on the other side of $H_{k'}$ (where, by convention, $+1$ if $x \in H_{k'}$). Also, each neuron $k^{'}$ of the first hidden layer is connected to one and only one input $x^{(i_{k'})}$ and the connection has weight $+1$ and offset $-\alpha_{i_{k'}}$.

Designing the second hidden layer (HL_2): HL_1 outputs a $(k-1)$-dimensional vector of ± 1-bits that encodes the exact position of x in the leaves of the tree. The leaf node identity of x can be extracted from the above-mentioned vector using a weighted combination of the bits along with an appropriate threshold function. Second hidden layer consists of k neurons, one for each leaf, and assigns a terminal cell to x. Let $HL_2 = \{L_1, \ldots, L_k\}$ denote the collection of all the leaf nodes of the tree, and let $L(x)$ be the leaf that contains x. We connect a unit $k^{'}$ from HL_1 to a unit k'' from HL_2 if and only if the hyperplane $H_{k'}$ is involved in the sequence of splits forming the path from the root to the leaf $L_{k''}$. The connection has weight $+1$ if the split by $H_{k'}$ is from a node to a right child in that path and -1 otherwise. Suppose we have $(u_1(x), \ldots, u_{k-1}(x))$ as the vector of ± 1-bits seen at the output of HL_1. Then the output $v_{k''}(x) \in \{-1, 1\}$ of neuron k'' is $\tau(\sum_{k'\to k''} b_{k',k''} u_{k'}(x) + b^0_{k''})$, where notation $k' \to k''$ means that k' is connected to k'', and $b_{k',k''} = \pm 1$ is the corresponding weight. The offset $b^0_{k''}$ is set to

$$b^0_{k''} = -l(k'') + \frac{1}{2} \tag{2}$$

where $l(k'')$ is the length of the path from the root to $L_{k''}$. The rationale behind the choice (2) is that there are exactly $l(k'')$ connections starting from the first layer and pointing to k'' and that

$$\begin{cases} \sum_{k'\to k''} b_{k',k''} u_{k'}(x) - l(k'') + 1/2 = 1/2 & \text{if } x \in L_{k''} \\ \sum_{k'\to k''} b_{k',k''} u_{k'}(x) - l(k'') + 1/2 \leq -1/2 & \text{if } x \notin L_{k''} \end{cases} \tag{3}$$

Using (2), the argument of the threshold function is $\frac{1}{2}$ if $x \in L_{k''}$, and is smaller than $\frac{-1}{2}$, otherwise. Thus, $v_{k''}(x) = 1$ if and only if the terminal cell of x is $L_{k''}$. To summarize, HL_2 outputs a vector of ± 1-bits $(v_1(x), \ldots, v_k(x))$ whose components equal -1 except the one corresponding to the leaf $L(x)$, which is +1.

Output layer (*OL*)**:** Let $(v_1(x), \ldots, v_k(x))$ be the output of the HL_2. If $v_{k''}(x) = 1$, then the output layer calculates the average $\bar{Y}_{k''}$ of the Y_i corresponding to X_i falling in $L_{k''}$ as follows:

$$t_n(x) = \sum_{k''=1}^{k} w_{k''} v_{k''}(x) + b_{\text{out}} \tag{4}$$

where $w_{k''} = \frac{\bar{Y}_{k''}}{2}$ and $b_{\text{out}} = \frac{1}{2} \sum_{k''=1}^{k} \bar{Y}_{k''}$ for all $k'' \in \{1, \ldots, k\}$.

In order to increase the generalization capabilities of the proposed NSNT model, we replace the original relay-type activation function $\tau(u)$ with a hyperbolic tangent activation function $\sigma(u) := \tanh(u)$ that has ranges between -1 to 1. Specifically, we use $\sigma_1(u) = \sigma(\beta_1 u)$ at every neuron of the first hidden layer and $\sigma_2(u) = \sigma(\beta_2 u)$ at every neuron of the second hidden layer. Here, β_1 and β_2 are positive hyper-parameters that determine the contrast of the hyperbolic tangent activation; larger the parameters β_1 and β_2, sharper is the transition from -1 to 1. As β_1 and β_2 approach to infinity, the continuous functions σ_1 and σ_2 converge to the threshold function. The hyperbolic tangent activation functions allow operations with a smooth approximation of the discontinuous step activation function. Having a differentiable loss function with respect to the parameters almost everywhere in the network helps the gradients to be backpropagated while training the model. The probabilistic interpretation of the network output can be obtained by interpreting the activation functions in the hidden layers.

Remark 1 The tree estimate t_n, depending on D_n, can be interpreted as a neural network estimate. The architecture of this network is kept fixed and so are the weights and offsets of the network layers. The idea of building a near-optimal neural tree with sparse connections is to keep the structure of the network intact (as discussed above) and let the parameters vary in a subsequent network training procedure with backpropagation algorithm. Thus, once we design the connections between the neurons of the NSNT model, we can learn network parameters in a better way by minimizing the empirical MSE for this network over the sample D_n. Fitting a fully-connected deep feed-forward neural network model with two hidden layers (p input features, $k_n - 1$ neurons in the HL_1, and k_n neurons in the HL_2) requires $o(pk_n + k_n^2)$ parameters to fit whereas for the proposed NSNT model, it is $o(k_n log k_n)$ (assuming that decision tree generates roughly balanced trees). We show the near-optimal rate of convergence of the proposed frameworks for nonparametric regression set-up in Section 4. Since the value of k depends on n, we can use k_n instead of k in the paper with the same meaning.

In the next section, we show the asymptotic consistency (local and strong) of the proposed NSNT model in the context of nonparametric regression using empirical risk minimization.

3 Strong Consistency of the Proposed NSNT Model

Consider the tree structure and denote it by $\mathscr{G}_1 \equiv \mathscr{G}_1(D_n)$, the bipartite graph which creates the connections between the input vectors $x = (x^{(1)}, \ldots, x^{(p)})$ and the $k_n - 1$ hidden neurons of HL_1. Similarly, let $\mathscr{G}_2 \equiv \mathscr{G}_2(D_n)$ be the bipartite graph that represents the connections between the first hidden layer and the k_n hidden neurons of HL_2. Let $M(\mathscr{G}_1)$ be the set of $p \times (k_n - 1)$ matrices $A = (a_{ij})$ such that $a_{ij} = 0$ if $(i,j) \notin \mathscr{G}_1$. Also let $M(\mathscr{G}_2)$ be the $(k_n - 1) \times k_n$ matrices $B = (b_{ij})$ such that $b_{ij} = 0$ if $(i,j) \notin \mathscr{G}_2$. The parameters that specify the first hidden units are contained in a matrix A of $M(\mathscr{G}_1)$ identified by the weights over the edges of $\mathscr{G}_1$ and a column vector of biases b_1, of size $k_n - 1$. Similarly, the parameters of the second hidden units are represented by a matrix B of $M(\mathscr{G}_2)$ of weights over $\mathscr{G}_2$ and by a column vector b_2 of offsets having size k_n. Let us take the output weights and offset to be $W_{\text{out}} = (w_1, \ldots, w_{k_n})^\top \in R^{k_n}$ and $b_{\text{out}} \in R$, respectively. And the parameters that specify the model are represented by a vector:

$$\lambda = (A, b_1, B, b_2, W_{\text{out}}, b_{\text{out}}) \in M(\mathscr{G}_1) \times R^{k_n - 1} \times M(\mathscr{G}_2) \times R^{k_n} \times R^{k_n} \times R.$$

We further assume that there exists a positive constant c_1 such that

$$\|B\|_\infty + \|b_2\|_\infty + \|W_{\text{out}}\|_1 + |b_{\text{out}}| \leq c_1 k_n, \tag{5}$$

where $\|\cdot\|_\infty$ is the supremum norm of a matrix, and $\|\cdot\|_1$ is the L_1-norm of a vector. The rationale behind this assumption (5) is that the weights and offsets are taken by the computation units of the second hidden layer and the output layer. It can be easily verified that the condition is satisfied by the original random tree estimates when Y is assumed to be bounded.

Thus, we can assume that the absolute value of $\|Y\|_\infty \leq L < \infty$ almost surely, for some L. Therefore, letting $\Lambda_n = \big\{\lambda = (A, b_1, B, b_2, W_{\text{out}}, b_{\text{out}}) :$ (5) is satisfied$\big\}$, we see that the neural network implements functions of this specific form

$$f_\lambda(x) = W_{\text{out}}^\top \sigma_2\Big(B^\top \sigma_1(A^\top x + b_1) + b_2\Big) + b_{\text{out}}, \quad x \in \mathbb{R}^p,$$

where $\lambda \in \Lambda_n$. We aim to tune the parameters λ using the data D_n such that the function realized by the obtained network becomes a 'good' estimate that can minimize the empirical L_2 error. Let $\mathscr{F}_{n,k_n} = \big\{f_\lambda : \lambda \in \Lambda_n\big\}$ be the class of neural networks and m_n be the network that minimizes the empirical L_2 error which is defined as,

$$J_n(f) = \frac{1}{n}\sum_{i=1}^{n} \big|Y_i - f(X_i)\big|^2$$

over all functions $f \in \mathscr{F}_{n,k_n}$, i.e., $J_n(m_n) \leq J_n(f)$, where $\mathscr{F}_n$ is a rich class of functions, including additive functions, polynomial functions having coefficients of the same sign, products of continuous functions and etc. In order to establish the consistency of the regression function estimates m_n, we specify some specific class F_n of functions over C^p. Consider a hyperplane $S = [a_1, b_1] \times [a_2, b_2] \times \cdots \times [a_p, b_p] \subset C^p$, we are given a measurable function $f : C^p \to \mathbb{R}$ together with $S \subset C^p$. We consider the following two statements:

(a) For any $i \in \{1, 2, \cdots, p\}$, the function

$$x_i \mapsto \int_{\prod\limits_{j\neq i}[a_i, b_i]} f(x)dx_1 \cdots dx_{i-1}dx_{i+1} \cdots dx_p$$

is constant on $[a_i, b_i]$.

(b) the function f is constant on S.

Definition 1 If F_n be the class of continuous real function on C^p such that, for any $S = [a_1, b_1] \times [a_2, b_2] \times \cdots \times [a_p, b_p] \subset C^p$, then (a) implies (b).

For example, the additive functions of the following form $f(x) = \sum_{i=1}^{p} f_i(x^{(i)})$, where each f_i is continuous, belong to F_n. Also, the products of continuous functions of the following form $f(x) = \prod_{i=1}^{p} f_i(x^{(i)})$, where $[f_i > 0$ or $f_i < 0]$ are included in F_n, for all $i \in \{1, 2, \cdots, p\}$. Also, this will be true for polynomial function whose coefficients have the same sign.
Thus, we aim to find an estimate $m_n : C^p \to R$ of the regression function $m(X) = E[Y|X = x]$. We say m_n is consistent if (5) tends to 0 as $n \to \infty$. We can write using Lemma 10.1 of [21]

$$\int \left|m_n(X) - m(X)\right|^2 \mu(\mathrm{dx}) \leq 2 \sup_{f\in\mathscr{F}_{n,k_n}} \left|\frac{1}{n}\sum_{i=1}^{n} \left|Y_i - f(X_i)\right|^2 - E\left|Y - f(X)\right|^2\right| + \inf_{f\in\mathscr{F}_{n,k_n}} \int_{C^p} \left|f(x) - m(x)\right|^2 \mu(\mathrm{dx}), \quad (6)$$

where μ denotes the distribution of X. For the strong consistency of the near-optimal sparse neural regression trees model, we show that the *estimation error* (first term in the R.H.S. of Eqn. 6) and the *approximation error* (second term in the R.H.S. of Eqn. 6) tend to 0. The former can be proved by using non-asymptotic uniform deviation inequalities and covering numbers corresponding to $\mathscr{F}_n$, to be shown in Theorem 1. Approximation error can be handled using a pseudo-estimate similar to decision tree generated function t_n and application of Lipschitz property on the activation function of the model, as shown in Theorem 2. We further assume that X is uniformly distributed in C^p and $\|Y\|_\infty \leq L < \infty$ almost surely, for some L in the proof of Theorem 1 and 2.

The next two theorems state that with certain restrictions imposed on the number k_n of terminal nodes and with the parameters, β_1 and β_2 being properly regulated as functions of n, the empirical L_2 risk-minimization provides the strong consistency of the proposed NSNT model.

Theorem 1 (Estimation error) *Assume that X is uniformly distributed in C^p and $\|Y\|_\infty \leq L < \infty$. If k_n, β_2 satisfy*

$$k_n \to \infty, \quad \beta_2 \to \infty, \quad \frac{k_n^6 log(\beta_2 k_n^5)}{n} \to 0, \quad \textit{and there exists} \quad \delta > 0 \quad \textit{such that} \quad \frac{k_n^4}{n^{1-\delta}} \to 0,$$

then estimation error tends to zero ($n \to \infty$).

Proof Let $\mathscr{F}_n$ be the set containing all neural networks constrained by Eqn. (4) with inputs in $\mathbb{R}^p$ having two hidden layers of respective size $k_n - 1$ and k_n, and one output unit. We have assumed that for each $f \in \mathscr{F}_{n,k_n}$, f satisfies $\|f\|_\infty \leq c_1 k_n$ and Y is also bounded ($\|Y\|_\infty \leq L < \infty$). Let $z_1^n = (z_1, \ldots, z_n)$ be a vector of n fixed points in R^p and let $\mathscr{H}$ be a set of functions from $R^p \to R$. For every $\varepsilon > 0$, we let $\mathscr{N}_1(\varepsilon, \mathscr{H}, z_1^n)$ be the L_1 ε-covering number of $\mathscr{H}$ with respect to $z_1, \ldots, z_n$. $\mathscr{N}_1(\varepsilon, \mathscr{H}, z_1^n)$ is defined as the smallest integer N such that there exist functions $h_1, \ldots, h_N : R^p \to R$ with the property that for every $h \in \mathscr{H}$, there is a $j \in \{1, \ldots, N\}$ such that

$$\frac{1}{n}\sum_{i=1}^{n} \left|h(z_i) - h_j(z_i)\right| < \varepsilon.$$

Note that if $Z_1^n = (Z_1, \ldots, Z_n)$ is a sequence of i.i.d. random variables, then $\mathscr{N}_1(\varepsilon, \mathscr{H}, Z_1^n)$ is a random variable as well. Now, let $Z = (X, Y)$, $Z_1 = (X_1, Y_1), \ldots, Z_n = (X_n, Y_n)$, and $C^p = [0,1]^p$, we write

$$\mathscr{H}_n = \left\{ h(x,y) := |y - f(x)|^2 : (x,y) \in C^p \times [-L, L] \text{ and } f \in \mathscr{F}_n \right\}.$$

The functions in $\mathscr{H}_n$ will satisfy the following: $0 \leq h(x,y) \leq 2c_1^2 k_n^2 + 2L^2 \leq 4c_1^2 k_n^2$. This assumption holds for large n such that $c_1 k_n \geq L$ is satisfied. Using Pollard's inequality [21], we have, for arbitrary $\varepsilon > 0$,

$$\begin{aligned} &P\left\{ \sup_{f \in \mathscr{F}_n} \left| \frac{1}{n}\sum_{i=1}^{n} \left|Y_i - f(X_i)\right|^2 - E\left|Y - f(X)\right|^2 \right| > \varepsilon \right\} \\ &= P\left\{ \sup_{h \in \mathscr{H}_n} \left| \frac{1}{n}\sum_{i=1}^{n} h(Z_i) - E(h(Z)) \right| > \varepsilon \right\} \\ &\leq 8E\left[\mathscr{N}_1\left(\frac{\varepsilon}{8}, \mathscr{H}_n, Z_1^n\right)\right] \exp\left(-\frac{n\varepsilon^2}{128(4c_1^2 k_n^2)^2}\right). \end{aligned} \tag{7}$$

Next we bound the covering number $\left(\mathscr{N}_1(\frac{\varepsilon}{8}, \mathscr{H}_n, Z_1^n)\right)$. Let us consider two functions $h_i(x,y) = |y - f_i(x)|^2$ of $\mathscr{H}_n$ for some $f_i \in \mathscr{F}_n$ and $i = 1, 2$. We get

$$\begin{aligned}
&\frac{1}{n}\sum_{i=1}^{n}\left|h_1(Z_i) - h_2(Z_i)\right| \\
&= \frac{1}{n}\sum_{i=1}^{n}\left|\left|Y_i - f_1(X_i)\right|^2 - \left|Y_i - f_2(X_i)\right|^2\right| \\
&= \frac{1}{n}\sum_{i=1}^{n}\left|f_1(X_i) - f_2(X_i)\right| \times \left|f_1(X_i) - Y_i + f_2(X_i) - Y_i\right| \\
&\le \frac{4c_1k_n}{n}\sum_{i=1}^{n}\left|f_1(X_i) - f_2(X_i)\right|.
\end{aligned}$$

Thus, if $\{h_1, h_2, ..., h_l\}$ is an $\varepsilon/8$ packing of $\mathscr{H}_n$ on Z_1^n, then $\{f_1, f_2, ..., f_l\}$ is an $\varepsilon/64c_1k_n$ packing of $\mathscr{F}_n$.

$$\text{Thus,} \quad \mathscr{N}_1\left(\frac{\varepsilon}{8}, \mathscr{H}_n, Z_1^n\right) \le \mathscr{N}_1\left(\frac{\varepsilon}{64c_1k_n}, \mathscr{F}_n, X_1^n\right). \tag{8}$$

The covering number $\mathscr{N}_1(\frac{\varepsilon}{64c_1k_n}, \mathscr{F}_n, X_1^n)$ can be upper bounded independently of X_1^n by extending the arguments of Theorem 16.1 of [21,] from a network with one hidden layer to a network with two hidden layers. We will now apply Theorem 9.4, Lemma 16.4, and Lemma 16.5 repeatedly in the rest of the proof [21,]. Let the neurons of the first hidden layer output belong to the class

$$G_1 = \{\sigma_1(a^\top x + a_0) : a \in R^p, a_0 \in R\}.$$

For $0 < \varepsilon < 1/4$,

$$\begin{aligned}
\mathscr{N}_1(\varepsilon, G_1, X_1^n) &\le 3\left(\frac{2e}{\varepsilon}\log\frac{3e}{\varepsilon}\right)^{p+2} \\
&= 3\left(\frac{3e}{\varepsilon}\right)^{2p+4}.
\end{aligned}$$

Next, letting $G_2 = \{bg : g \in G_1, b \in [-c_1k_n, c_1k_n]\}$ we get

$$\begin{aligned}
\mathscr{N}_1(\varepsilon, G_2, X_1^n) &\le \frac{4c_1k_n}{\varepsilon}\mathscr{N}_1\left(\frac{\varepsilon}{2c_1k_n}, G_1, X_1^n\right) \\
&= \frac{3 \times 4c_1k_n}{\varepsilon}\left(\frac{3e \times 2c_1k_n}{\varepsilon}\right)^{2p+4} \\
&\le \left(\frac{12ec_1k_n}{\varepsilon}\right)^{2p+5}.
\end{aligned}$$

The second units compute the functions of the collection

$$G_3 = \left\{\sigma_2\left(\sum_{i=1}^{k_n-1} g_i + b_0\right) : g_i \in G_2, b_0 \in [-c_1k_n, c_1k_n]\right\}.$$

Note that σ_2 satisfies the Lipschitz property $|\sigma_2(u) - \sigma_2(v)| \leq \beta_2|u - v|$ for all $(u, v) \in R^2$. Thus,

$$\begin{aligned}\mathscr{N}_1(\varepsilon, G_3, X_1^n) &\leq \frac{2c_1\beta_2 k_n^2}{\varepsilon}\mathscr{N}_1\Big(\frac{\varepsilon}{2\beta_2 k_n}, G_2, X_1^n\Big)^{k_n - 1} \\ &= \frac{2c_1\beta_2 k_n^2}{\varepsilon}\left[\Big(\frac{12ec_1 k_n \times 2\beta_2 k_n}{\varepsilon}\Big)^{2p+5}\right]^{k_n-1} \\ &\leq \Big(\frac{24ec_1\beta_2 k_n^2}{\varepsilon}\Big)^{(2p+5)k_n}.\end{aligned}$$

Also, letting

$$G_4 = \{wg : g \in G_3, w \in [-c_1k_n, c_1k_n]\},$$

By assuming without loss of generality $c_1, \beta_2 \geq 1$, we get

$$\begin{aligned}\mathscr{N}_1(\varepsilon, G_4, X_1^n) &\leq \frac{4c_1k_n}{\varepsilon}\mathscr{N}_1\Big(\frac{\varepsilon}{2c_1k_n}, G_3, X_1^n\Big) \\ &= \frac{4c_1k_n}{\varepsilon}\Big(\frac{48ec_1^2\beta_2k_n^3}{\varepsilon}\Big)^{(2p+5)k_n} \\ &\leq \Big(\frac{48ec_1^2\beta_2k_n^3}{\varepsilon}\Big)^{(2p+5)k_n+1}.\end{aligned}$$

Finally, we can write

$$\mathscr{F}_n = \Big\{\sum_{i=1}^{k_n} g_i + b_{\text{out}} : g_i \in G_4, b_{\text{out}} \in [-c_1k_n, -c_1k_n]\Big\}.$$

We conclude

$$\begin{aligned}\mathscr{N}_1(\varepsilon, \mathscr{F}_n, X_1^n) &\leq \frac{2c_1k_n(k_n+1)}{\varepsilon}\left[\mathscr{N}_1\Big(\frac{\varepsilon}{k_n+1}, G_4, X_1^n\Big)\right]^{k_n} \\ &\leq \Big(\frac{48ec_1^2\beta_2(k_n+1)^4}{\varepsilon}\Big)^{(2p+5)k_n^2+k_n+1}.\end{aligned} \tag{9}$$

Combining (7)-(9) together, we obtain

$$\begin{aligned}&P\Big\{\sup_{f\in\mathscr{F}_n}\Big|\frac{1}{n}\sum_{i=1}^{n}\big|Y_i - f(X_i)\big|^2 - E\big|Y - f(X)\big|^2\Big| > \varepsilon\Big\} \\ &\leq 8\Big(\frac{3072ec_1^3\beta_2(k_n+1)^5}{\varepsilon}\Big)^{(2p+5)k_n^2+k_n+1} e^{\left(-\frac{n\varepsilon^2}{2048c_1^4k_n^4}\right)}.\end{aligned}$$

Now if the conditions of Theorem 1 holds, then

$$\sum_{n=1}^{\infty} P\Big\{ \sup_{f\in\mathscr{F}_n} \Big|\frac{1}{n}\sum_{i=1}^{n}\big|Y_i - f(X_i)\big|^2 - E\big|Y-f(X)\big|^2\Big| > \varepsilon\Big\}$$

$$\leq \sum_{n=1}^{\infty} 8\exp\left[\Big((2p+5)k_n^2 + k_n + 1\Big)\log\Big(\frac{3072ec_1^3\beta_2(k_n+1)^5}{\epsilon}\Big) - \frac{n\epsilon^2}{2048c_1^4k_n^4}\right]$$

$$\leq \sum_{n=1}^{\infty} 8\left[-n^{\delta}.\frac{n^{1-\delta}}{k_n^4}\left(\frac{\epsilon^2}{2048c_1^4} - \frac{\big((2p+5)k_n^2+k_n+1\big)k_n^4\log\Big(\frac{3072ec_1^3\beta_2(k_n+1)^5}{\epsilon}\Big)}{n}\right)\right]$$

$$< \infty$$

This together with Borel-Cantelli lemma, gives

$$\lim_{n\to\infty}\sup_{f\in\mathscr{F}_n}\Big|\frac{1}{n}\sum_{i=1}^{n}\big|Y_i-f(X_i)\big|^2 - E\big|Y-f(X)\big|^2\Big| = 0.$$

Remark 2 The above theorem assumes

$$\frac{k_n^6 log(\beta_2 k_n^5)}{n} \to 0, \quad \text{and there exists} \quad \delta > 0 \quad \text{such that} \quad \frac{k_n^4}{n^{1-\delta}} \to 0,$$

for strong consistency. For weak consistency, we need a more relaxed condition to be satisfied [5]. Let $\widetilde{Z}$ be a non-negative random variable. Using this for any $0 < \varepsilon < 1/4$ and large n, we can write

$$E[\widetilde{Z}] = \int_0^{\infty} P[\widetilde{Z} > t]dt \leq \varepsilon + \int_{\varepsilon}^{\infty} P[\widetilde{Z} > t]dt. \tag{10}$$

Using (10), we can write

$$E\Big[\sup_{f\in\mathscr{F}_n}\Big|\frac{1}{n}\sum_{i=1}^{n}\big|Y_i-f(X_i)\big|^2 - E\big|Y-f(X)\big|^2\Big|\Big]$$

$$\leq \varepsilon + 8\int_{\varepsilon}^{\infty}\Big(\frac{3072ec_1^3\beta_2(k_n+1)^5}{t}\Big)^{(2p+5)k_n^2+k_n+1}\exp\Big(-\frac{nt^2}{2048c_1^4k_n^4}\Big)\,\mathrm{d}t$$

$$\leq \varepsilon + 8\Big(\frac{3072ec_1^3\beta_2(k_n+1)^5}{\varepsilon}\Big)^{(2p+5)k_n^2+k_n+1} \times \Big[-\frac{2048c_1^4k_n^4}{n\varepsilon}e^{-\frac{n\varepsilon t}{2048c_1^4k_n^4}}\Big]_{t=\varepsilon}^{\infty}$$

$$\leq \varepsilon + 8\Big(\frac{3072ec_1^3\beta_2(k_n+1)^5}{\varepsilon}\Big)^{(2p+5)k_n^2+k_n+1} \times \Big(\frac{2048c_1^4k_n^4}{n\varepsilon}\Big)e^{-\frac{n\varepsilon^2}{2048c_1^4k_n^4}}$$

$$\leq \varepsilon + 8\Big(\frac{2048c_1^4k_n^4}{n\varepsilon}\Big)\exp\left[\big((2p+5)k_n^2+k_n+1\big)\log\Big(\frac{3072ec_1^3\beta_2(k_n+1)^5}{\varepsilon}\Big) - \frac{n\varepsilon^2}{2048c_1^4k_n^4}\right].$$

As $k_n, \beta_2 \to \infty$, $\frac{k_n^6 log(\beta_2 k_n^5)}{n} \to 0$, for large n, we have,

$$\lim_{n\to\infty} E\Big[\sup_{f\in\mathscr{F}_n}\Big|\frac{1}{n}\sum_{i=1}^{n}\big|Y_i - f(X_i)\big|^2 - E\big|Y - f(X)\big|^2\Big|\Big] = 0.$$

Theorem 2 (Approximation error) *Assume that X is uniformly distributed in C^p, $\|Y\|_\infty \le L < \infty$, and $m \in \mathscr{F}_n$. If k_n, β_1, β_2 satisfy*

$$k_n \to \infty,\ \beta_1 \to \infty,\ \beta_2 \to \infty,\ k_n^2 e^{-2\beta_2} \to 0,\ \frac{\beta_2^2 k_n^4}{\beta_1} \to 0, \quad \textit{and}\ \frac{k_n log(n)}{n} \to 0,$$

then approximation error tends to zero ($n \to \infty$).

Proof Let us consider a piece-wise constant function (pseudo-estimate) similar to the tree estimate t_n, with only one difference: the function computes the true conditional expectation $E[Y|X \in L_{k''}]$ in each leaf $L_{k''}$, but not the empirical one, viz. $\bar{Y}_{k''}$. In another way, we can write $(W^\star_{\text{out}})_{k''} = \mathbb{E}[Y|X \in L_{k''}]/2$ and $b^\star_{\text{out}} = \sum_{k''=1}^{k_n} E[Y|X \in L_{k''}]/2$ in Eqn. (3) of Section 3. This tree-type pseudo-estimate has the form

$$t_{\lambda^\star}(x) = W^{\star\top}_{\text{out}}\tau\Big(B^{\star\top}\tau(A^{\star\top}x + b_1^\star) + b_2^\star\Big) + b^\star_{\text{out}}, \quad x \in \mathbb{R}^p,$$

for some $\lambda^\star = (A^\star, b_1^\star, B^\star, b_2^\star, W^\star_{\text{out}}, b^\star_{\text{out}})$. We can write the expression for approximation error as

$$\inf_{f\in\mathscr{F}_n}\int_{C^p}\big|f(x) - m(x)\big|^2\mu(dx) \le 2\Big[\int_{C^p}\big|f_{\lambda^\star}(x) - t_{\lambda^\star}(x)\big|^2\mu(dx) + \int_{C^p}\big|t_{\lambda^\star}(x) - m(x)\big|^2\mu(dx)\Big]. \tag{11}$$

Now, we show that the two terms in the R.H.S. of (11) tends to zero under certain restrictions on k_n, β_1 and β_2. The second term requires a careful analysis of the asymptotic behavior of the cells for the tree pseudo-estimate $t_{\lambda^\star}(x)$. To deal with the first term we consider the following two expressions:

$$t_{\lambda^\star}(x) = W^{\star\top}_{\text{out}}\Big[\tau\Big(B^{\star\top}\tau(A^{\star\top}x + b_1^\star) + b_2^\star\Big)\Big] + b^\star_{\text{out}},$$

$$f_{\lambda^\star}(x) = W^{\star\top}_{\text{out}}\Big[\sigma_2\Big(B^{\star\top}\sigma_1(A^{\star\top}x + b_1^\star) + b_2^\star\Big)\Big] + b^\star_{\text{out}}.$$

Using Cauchy-Schwarz inequality and triangle inequality; we can write for all $x \in \mathbb{R}^p$

$$\begin{aligned}
&\big|f_{\lambda^\star}(x) - t_{\lambda^\star}(x)\big|^2 \\
&\le \frac{k_n L^2}{4} \times \Big\|\sigma_2\Big(B^{\star\top}\sigma_1(A^{\star\top}x + b_1^\star) + b_2^\star\Big) - \tau\Big(B^{\star\top}\tau(A^{\star\top}x + b_1^\star) + b_2^\star\Big)\Big\|^2 \\
&\le \frac{2k_n L^2}{4}\Bigg[\Big\|\sigma_2\Big(B^{\star\top}\tau(A^{\star\top}x + b_1^\star) + b_2^\star\Big) - \tau\Big(B^{\star\top}\tau(A^{\star\top}x + b_1^\star) + b_2^\star\Big)\Big\|^2 \\
&\qquad + \Big\|\sigma_2\Big(B^{\star\top}\sigma_1(A^{\star\top}x + b_1^\star) + b_2^\star\Big) - \sigma_2\Big(B^{\star\top}\tau(A^{\star\top}x + b_1^\star) + b_2^\star\Big)\Big\|^2\Bigg] \\
&= \frac{k_n L^2}{2}\Big[I_1 + I_2\Big]. \qquad (12)
\end{aligned}$$

To find the upper bounds of I_1 and I_2, we will use the following properties of functions:

- Recall that σ_i is tan hyperbolic activation function and τ is a threshold activation function, then we can write for all $u \in R$, $|\sigma_i(u) - \tau(u)| \le 2e^{-2\beta_i|u|}$ for all $i = 1, 2$.
- Also, σ_i satisfies the Lipschitz property for continuous functions $|\sigma_i(u) - \sigma_i(v)| \le \beta_i|u - v|$ for all $(u, v) \in R^2$ and $i = 1, 2$.

Using the above properties of functions, we can write

$$I_1 \le 4\sum_{j=1}^{k_n} \exp\Big[-4\beta_2\big|\big(B^{\star\top}\tau(A^{\star\top}x + b_1^\star) + b_2^\star\big)_j\big|\Big] \le 4k_n e^{-2\beta_2},$$

since for every j, $\big|\big(B^{\star\top}\tau(A^{\star\top}x + b_1^\star) + b_2^\star\big)_j\big| \ge 1/2$.

Using Lipschitz property and Cauchy-Schwarz inequality, we can find the upper-bound for I_2 as follows:

$$\begin{aligned}
I_2 &\le \sum_{j=1}^{k_n} \beta_2^2\Big|\Big(B^{\star\top}\big(\sigma_1(A^{\star\top}x + b_1^\star) - \tau(A^{\star\top}x + b_1^\star)\big)_j\Big|^2 \\
&\le \beta_2^2 k_n^2\big\|\sigma_1(A^{\star\top}x + b_1^\star) - \tau(A^{\star\top}x + b_1^\star)\big\|^2 \\
&\le 4\beta_2^2 k_n^2 \sum_{j=1}^{k_n - 1} \exp\Big(-4\beta_1|(A^{\star\top}x + b_1^\star)_j|\Big) \\
&\le 4\beta_2^2 k_n^3 \exp\big(-4\beta_1\varepsilon\big) \quad \text{(for some fixed } j \text{ and arbitrary } \varepsilon > 0)
\end{aligned}$$

For all n large enough, choosing $\varepsilon = \frac{\log(\beta_1)}{4\beta_1}$, to get $I_2 \le \frac{4\beta_2^2 k_n^3}{\beta_1}$.

Putting these upper bounds of I_1 and I_2 together and using Eqn. (12) we get the following:

$$\int_{C^p} \big|f_{\lambda^\star}(x) - t_{\lambda^\star}(x)\big|^2 \mu(\mathrm{d}x) \le 2L^2\Bigg[k_n^2 e^{-2\beta_2} + \frac{\beta_2^2 k_n^4}{\beta_1}\Bigg]. \qquad (13)$$

R.H.S. of (13) tends to zero if the conditions $k_n^2 e^{-2\beta_2} \to 0$, and $\frac{\beta_2^2 k_n^4}{\beta_1} \to 0$ hold.
To deal with the second term of Eqn. (11), we consider $m \in F_n$ and X is uniformly distributed in C^p and bounded Y.
Recall that $t_{\lambda^\star}(x) = E\left[Y|X \in S_n(x)\right]$, where $S_n(x)$ be the cell of the tree that contains x. Accordingly,

$$\begin{aligned}\left|t_{\lambda^\star}(x) - m(x)\right|^2 &= \left|E\left[Y|X \in S_n(x)\right] - m(x)\right|^2 \\ &= \left|E\left[m(X)|X \in S_n(x)\right] - m(x)\right|^2 \\ &\leq \sup_{z,z' \in S_n(x)} \left|m(z) - m(z')\right|^2\end{aligned} \tag{14}$$

where m is continuous on C^p. It reduces to the problem of finding empirically optimal regression trees (as in [8]) to yield consistent estimates of $m(\cdot)$. We can directly use the consistency results of the CART model to show that the R.H.S. of (14) tends to zero when $k_n \to \infty$ and $k_n = o(n/logn)$ as $n \to \infty$ [36][12]. The condition suggests that the tree estimates are consistent when the size of the tree grows with sample size n at a controller rate.

Remark 3 The consistency results for near-optimal sparse neural regression trees depend on the choice of the total number of leaves and certain restrictions imposed on neural network hyper-parameters to ensure the theoretical consistency of the model.

4 Analysis of Rate of Convergence

We can find the rate of convergence by using complexity regularization principle [28][22][27]. Using Equation (9), we can penalize the complexity of $\mathscr{F}_{n,k_n}$. For a detailed discussion on penalized risk minimization, one may refer to Chapter 12 of [21]. We balance the approximation error with the bounds on the covering number to get the following theorem for the rate of convergence of the near-optimal sparse neural regression trees model. The similar idea for finding rate of convergence for single and multilayered perceptrons has been used in [21][28][33]. In the sequel, we have assumed that m is Lipschitz (δ, c)-smooth according to the following definition:

Definition 2 Let $\delta \in (0, 1]$ and $c \in R_+$, then a function $m : C^p \to R$ is called Lipschitz (δ, C)-smooth if it satisfies the following equation:

$$|m(x) - m(z)| \leq c\|x - z\|^\delta$$

for all $x, z \in [0, 1]^p$.

Theorem 3 (Rate of convergence) *Assume that X is uniformly distributed in C^p and $\|Y\|_\infty \leq L < \infty$ a.s. and m is Lipschitz (δ, c)-smooth. Let m_n be the* *estimate that minimizes empirical L_2-risk and the network activation function σ_i satisfies Lipschitz property. Then for any $n \geq max\{\beta_2, 2^{p+1}L\}$, we have*

$$E\int_{C^p}\left|m_n(X)-m(X)\right|^2\mu(dx) = O\left(\frac{log(n)^6}{n}\right)^{\frac{2\delta}{2\delta+2p}},$$

where δ characterizes the smoothness of the true function.

Proof Assume for any $n \geq max\{\beta_2, 2^{p+1}L\}$. According to (9) and for any $X_1^n \in R^p$

$$\begin{aligned}\mathscr{N}_1(\frac{1}{n}, \mathscr{F}_{n,k_n}, X_1^n) &\leq \left(\frac{48ec_1^2\beta_2(k_n+1)^4}{1/n}\right)^{(2p+5)k_n^2+k_n+1}\\ &\leq \left(48ec_1^2(n+1)^6\right)^{(2p+5)k_n(k_n+1)}. \qquad (15)\end{aligned}$$

Next we use the complexity regularization principle to choose the parameter k_n of the estimate in a data-dependent way. To do this let

$$\sup_{X_1^n}\mathscr{N}_1\left(\frac{1}{n}, \mathscr{F}_{k_n}, X_1^n\right) \leq \mathscr{N}_1\left(\frac{1}{n}, \mathscr{F}_{k_n}\right)$$

be the upper bound on the covering number of $\mathscr{F}_{k_n}$ and define for $w_{k_n} \geq 0$

$$pen_n(k_n) = \frac{45L^2 log\mathscr{N}_1\left(\frac{1}{n}, \mathscr{F}_{k_n}\right) + w_{k_n}}{n}$$

as a penalty term penalizing the complexity of $\mathscr{F}_{k_n}$ [28]. Thus (15) implies that $pen_n(k_n)$ is of the following form with $w_{k_n} = 1$,

$$pen_n(k_n) = \frac{45L^2(2p+5)k_n(k_n+1)log\left(48ec_1^2(n+1)^6\right)+1}{n} = O\left(\frac{k_n^2 log(n)^6}{n}\right).$$

Our proof for rate of convergence relies on an extension of the proof techniques introduced by [28] and Chapter 12 of [21]. Assuming Y is bounded as in Theorem 1 and 2, we write (6) as

$$E\int_{C^p}\left|m_n(X)-m(X)\right|^2\mu(\mathrm{dx}) \leq 2\min_{k_n\geq 1}\left\{pen_n(k_n) + \inf_{f\in\mathscr{F}_{k_n}}\int_{C^p}\left|f(x)-m(x)\right|^2\mu(\mathrm{dx})\right\} + O\left(\frac{1}{n}\right) \qquad (16)$$

Thus, (16) becomes

$$\begin{aligned}E\int_{C^p}\left|m_n(X)-m(X)\right|^2\mu(\mathrm{dx}) \leq \min_{k_n\in\{1,2,\ldots,n\}}&\left\{\frac{90L^2(2p+5)k_n(k_n+1)log\left(48ec_1^2(n+1)^6\right)+1}{n}\right.\\ &\left.+2\inf_{f\in\mathscr{F}_{k_n}}\int_{C^p}\left|f(x)-m(x)\right|^2\mu(\mathrm{dx})\right\} + O\left(\frac{1}{n}\right) \qquad (17)\end{aligned}$$

The approximation error $\inf_{f\in\mathscr{F}_{k_n}} \int_{C^p} \left|f(x) - m(x)\right|^2 \mu(\mathrm{dx})$ depends on the smoothness of the regression function. According to Theorem 3.4 of [33] and Corollary 1 of [28], for any deep feed-forward networks with two hidden layers satisfying the assumptions of Theorem 3, we have

$$\left|f(x) - m(x)\right| \le c.\left(\frac{1}{k_n}\right)^{\frac{\delta}{p}}$$

for all $x \in [0,1]^p$. Using (17), we have

$$E\int_{C^p} \left|m_n(X) - m(X)\right|^2 \mu(\mathrm{dx}) \le \frac{k_n^2 log(n)^6}{n} + 2c^2\left(\frac{1}{k_n}\right)^{2\delta/p} \tag{18}$$

for sufficiently large n.
Now we have to balance the approximation error with the bound on the covering number. Thus, taking

$$k_n = c^{\frac{2p}{2p+2\delta}}\left(\frac{n}{log(n)^6}\right)^{\frac{p}{2p+2\delta}},$$

and upon using (18), we get

$$E\int_{C^p} \left|m_n(X) - m(X)\right|^2 \mu(\mathrm{dx}) \le c_1\left(\frac{log(n)^6}{n}\right)^{\frac{2\delta}{2\delta+2p}},$$

where $c_1 = 3c^{\frac{4p}{2p+2\delta}}$. Thus, we obtained the desired convergence rate for the proposed NSNT model. The rate of convergence for the proposed model is 'near-optimal' upto some logarithmic factor according to [45].

Remark 4 Near-optimal sparse neural trees is a hybrid concept representation which has a built-in strategy for neural networks. The architecture is less complex and have less number of tuning parameters resulting in less training time. Since the algorithm uses the prior knowledge of decision tress, thus the algorithm has less restrictions on the geometry of the decision boundaries. Theorem 1 and 2 point out the consistency results of the algorithm. Theorem 3 also points out a remarkable property of the proposed NSNT model and the significance of the name for the proposal.

5 Performance Comparison on Real-world Datasets

In this section, we report the out-of-sample accuracy of our proposed NSNT model in comparison with state-of-the-art models on 40 regression datasets and 40 classification datasets obtained from the UCI Machine Learning Repository [17]. A summary of the datasets is available in Table 1. Our objective is to assess the relative strength of NSNT. We partition each dataset into training (50%), validation (25%), and testing sets (25%). We employ 10-fold cross-validation with different randomly assigned training, validation, and test sets.

We use the area under the receiver operating characteristic curve (AUC) as the performance metric for classification datasets and the coefficient of determination (R^2) for regression datasets. By convention, the higher the value of AUC and R^2, the better the model is. Further, we compare our proposed NSNT model mostly with popularly used greedy, optimal, and near-optimal classifiers. Multivariate adaptive regression splines (MARS) [18], regression trees [8], random forest [9], XGBoost [14], optimal tree for classification and regression (OCT and ORT) [3], and near-optimal nonlinear trees for classification and regression (NNCT and NNRT) [4] are used for comparison. The available state-of-the-art models were implemented as follows. NNCT and NNRT were implemented in the Julia programming language as in [4]. For testing the performance of MARS, we used the *earth* package in the R language [35]. For random forests, we used the random forest package in R [29]. For gradient-boosted trees, we used the XGBoost library [15] with value of the parameter $\rho = 0.1$. For OCT and ORT models, we used the *OptimalTrees* package in Julia programming language with standard auto-tuning complexity parameter as in [3]. We first compare all methods across all datasets and present the out-of-sample performance among all methods with a maximum depth of 12 and results are reported in Table 2 and 3.

The training procedure for the proposed NSNT model is as follows. A decision tree is first built using the 'scikit-learn' package [37] for tree designing. From decision tree, we extract the set of all split directions and split positions and use them to build neural network initialization parameters. The hybrid models are then trained using the TensorFlow library in python software [1]. The optimization with the network model is done by minimizing the empirical error on the training set. It is achieved by employing an iterative stochastic gradient-descent optimization technique. The architecture of the network is kept fixed, and thus the weights and offsets of the three-layered DFFNN model are also fixed. A natural idea is then to keep the structure of the network intact and let the parameters vary in a subsequent network training procedure with backpropagation neural network training. For this, we used the default functions available in TensorFlow. In our model set up, Neural Network was trained for 100 epochs. The default hyper-parameter values were chosen for the gradient-based optimization algorithm available in the Python machine learning software. We experimentally found that using a lower value for β_2 than that for β_1 is appropriate for achieving the high accuracy of the model. In this case, the initial parameters of the tan-hyperbolic activation function in the two layers were chosen as: $\beta_1 = 100, \beta_2 = 1$. This is also evident from the theoretical results presented in Section 3 and 4. This is also practically very significant, since for a relatively small β_2, the transition in the activation function from -1 to $+1$ is smoother, and a very stronger stochastic gradient signal reaches the first hidden layer in the backpropagation training. Similarly, a converse explanation can be given for β_1. The training time and memory requirements are also quite low for the proposed NSNT model compared to advanced deep neural network models. Our proposed model is faster, especially when trained on a GPU. In Table 3, we present the results of different classifiers

Table 1 Description of the regression and classification datasets. Here, n denotes the total number of observations and p denotes the number of features in the dataset.

Regression Datasets	n	p	Classification Datasets	n	p	No. of Classes
abalone	4176	7	acute-nephritis	120	6	2
ailerons	7153	40	appendicitis	106	7	2
airfoil-self-noise	1502	5	australian	690	14	2
automobile	391	8	breast	569	32	2
autompg	158	51	breast-w	569	30	2
like_sharing	17379	12	bupa	345	6	2
BlogFeedBack	52397	80	congress-voting	435	16	2
cart-artificial	40767	10	credit-a	690	15	2
CBM	11934	16	crx	690	15	2
combined_cycle_power_plant	9568	4	cylinder-bands	512	35	2
communities-and-crime	122	122	fertility	100	9	2
computer-hardware	208	36	fourclass	862	2	2
concrete_data	1030	8	german	1000	20	2
concrete-slump-test-compressive	102	7	heart-c	303	13	2
concrete-slump-test-flow	102	7	hepatitis	155	19	2
concrete-slump-test-slump	102	7	hill-valley	606	100	2
cpu-act	8191	21	ilpd-indian-liver	583	9	2
cpu-small	8191	12	ion	351	33	2
elevators	8751	18	iris	150	4	3
Facebook Comment Volume	50993	53	mammographic	961	5	2
friedman-artificial	40767	10	monks-1	124	6	2
geographic-origin	1059	68	monks-2	169	6	2
housing	505	13	monks-3	3190	6	2
hybrid-price	152	3	parkinsons	195	22	2
kin8 nm	8191	8	pima	768	8	2
lpga-2008	156	6	promoters	106	57	2
lpga-2009	145	11	ringnorm	300	20	2
Obesity_levels	2111	16	sonar	1593	256	2
larkinsons-telemonitoring-motor	5874	16	spect	80	22	2
parkinsons-telemonitoring-total	5874	16	spectf	80	44	2
QSAR_aquatic_toxicity	546	8	splice-libsvm	1000	60	2
Real_estate	414	6	tae	151	5	3
esidential-Building-Data-Set-1	372	107	threenorm	300	20	2
Residential-Building-Data-Set-2	372	107	tic-tac-toe	958	9	2
Slice_localization_data	53500	385	titanic	2201	3	2
TomsHardware	28178	96	transfusion	748	5	2
vote-for-clinton	2703	9	twonorm	300	20	2
wine-quality-red	1598	11	vote	435	16	2
wine-quality-white	4897	11	vote1	435	15	2
yacht-hydrodynamics	307	6	wine	178	13	3

on the standard UCI datasets. The experimental results on standard UCI regression datasets in comparison with the state-of-the-art models are given in Table 2.

To sum up, among five competitive methods, NSNT is the winner based on the performance metrics (AUC value for classification datasets and R^2 for regression datasets) on an average. From the statistical point of view on the nature of the problem, the robustness of the proposed method implies that the designed NSNT model utilizes the sparse framework and near-optimal rate of convergence asymptotically. Moreover, NNCT (NNRT) and XGBoost are 'second' and 'third' best performing classifiers (regression models) in terms of the performance metrics for the majority of the datasets as compared to the other traditional methods considered in this study. From the experimental evaluation of different classifiers, it can be concluded that, on average, the near-optimal sparse neural trees model outperforms other greedy, optimal and near-optimal statistical and machine learning models in a significant margin. Thus, the proposed NSNT method can be a 'good' choice for statistical learning in regression and classification datasets arising in various applied domains.

Table 2 Out of sample performance (R^2) among all methods with maximum depth twelve. The number in parenthesis indicates the standard deviation and the best results are made bold.

Regression Datasets	MARS [18]	Random Forest [9]	XGBoost [14]	ORT [3]	NNRT [4]	Proposed NSNT
abalone	0.559 (0.02)	0.488 (0.07)	0.536 (0.01)	0.549 (0.01)	0.564 (0.01)	**0.587 (0.01)**
ailerons	0.829 (0.01)	0.837 (0.007)	0.823 (0.01)	0.836 (0.01)	**0.846 (0.0)**	0.829 (0.005)
airfoil-self-noise	0.802 (0.01)	0.91 (0.013)	0.941 (0.006)	0.896 (0.003)	0.903 (0.007)	**0.945 (0.0)**
automobile	0.849 (0.04)	0.847 (0.01)	0.849 (0.06)	0.838 (0.03)	0.804 (0.022)	**0.851 (0.0)**
autompg	0.844 (0.03)	0.732 (0.03)	0.736 (0.04)	0.631 (0.02)	0.847 (0.02)	**0.854 (0.01)**
Bike_sharing	0.935 (0.004)	0.93 (0.003)	**0.951 (0.0)**	0.939 (0.01)	0.949 (0.01)	0.948 (0.004)
BlogFeedBack	0.417 (0.001)	0.406 (0.001)	**0.563 (0.0)**	0.431 (0.0)	0.459 (0.0)	0.487 (0.001)
cart-artificial	0.945 (0.0)	0.945 (0.0)	**0.948 (0.0)**	**0.948 (0.0)**	**0.948 (0.0)**	**0.948 (0.0)**
CBM	0.93 (0.01)	0.942 (0.004)	0.934 (0.0)	0.927 (0.03)	0.935 (0.0)	**0.945 (0.02)**
combined_cycle_power_plant	0.949 (0.01)	0.956 (0.01)	0.966 (0.004)	0.955 (0.004)	**0.973 (0.0)**	0.960 (0.003)
communities-and-crime	0.301 (0.15)	0.702 (0.11)	0.679 (0.12)	**0.749 (0.06)**	0.684 (0.06)	0.719 (0.09)
computer-hardware	0.961 (0.013)	0.929 (0.02)	0.929 (0.01)	0.973 (0.004)	0.986 (0.01)	**0.988 (0.01)**
concrete_data	0.853 (0.005)	0.884 (0.005)	0.929 (0.06)	0.919 (0.003)	0.923 (0.003)	**0.931 (0.03)**
concrete-slump-test-compressive	0.947 (0.045)	0.69 (0.075)	0.792 (0.03)	0.873 (0.03)	**0.953 (0.02)**	0.854 (0.05)
concrete-slump-test-flow	0.364 (0.113)	0.437 (0.10)	0.396 (0.13)	0.476 (0.07)	0.524 (0.05)	**0.555 (0.07)**
concrete-slump-test-slump	0.265 (0.100)	0.359 (0.13)	0.272 (0.1)	0.277 (0.06)	**0.439 (0.04)**	0.408 (0.02)
cpu-act	0.975 (0.001)	0.982 (0.0)	0.98 (0.0)	**0.984 (0.0)**	0.981 (0.0)	**0.984 (0.0)**
cpu-small	0.966 (0.001)	**0.975 (0.001)**	**0.975 (0.0)**	0.974 (0.002)	0.971 (0.001)	0.973 (0.02)
elevators	0.895 (0.02)	0.811 (0.01)	0.833 (0.01)	0.912 (0.004)	0.918 (0.004)	**0.920 (0.002)**
Facebook Comment Volume	0.478 (0.001)	0.435 (0.0)	**0.528 (0.0)**	0.426 (0.0)	0.437 (0.0)	0.489 (0.0)
friedman-artificial	0.902 (0.0)	0.932 (0.0)	0.952 (0.0)	**0.956 (0.0)**	**0.956 (0.0)**	**0.956 (0.0)**
geographic-origin	0.609 (0.06)	0.608 (0.07)	0.627 (0.06)	0.53 (0.02)	0.562 (0.03)	**0.638 (0.04)**
housing	0.507 (0.031)	0.852 (0.03)	**0.863 (0.03)**	0.804 (0.02)	0.791 (0.01)	0.856 (0.02)
hybrid-price	0.62 (0.109)	0.615 (0.07)	0.577 (0.11)	**0.648 (0.04)**	0.642 (0.053)	0.638 (0.044)
kin8_nm	0.75 (0.01)	0.674 (0.01)	0.671 (0.01)	0.851 (0.005)	0.809 (0.005)	**0.863 (0.002)**
lpga-2008	**0.897 (0.03)**	0.77 (0.01)	0.775 (0.04)	0.851 (0.02)	0.863 (0.013)	**0.897 (0.02)**
lpga-2009	0.84 (0.014)	0.893 (0.02)	0.885 (0.02)	**0.904 (0.009)**	0.889 (0.01)	0.884 (0.01)
Obesity_levels	0.842 (0.011)	0.867 (0.02)	**0.882 (0.01)**	0.85 (0.0)	0.863 (0.005)	0.875 (0.0)
parkinsons-telemonitoring-motor	0.3 (0.01)	**0.341 (0.01)**	0.264 (0.013)	0.281 (0.0)	0.245 (0.005)	0.295 (0.01)
parkinsons-telemonitoring-total	0.289 (0.018)	**0.355 (0.014)**	0.27 (0.019)	0.316 (0.01)	0.282 (0.0)	0.304 (0.015)
QSAR_aquatic_toxicity	0.401 (0.09)	0.539 (0.02)	0.541 (0.07)	0.569 (0.05)	0.585 (0.042)	**0.600 (0.0)**
Real_estate	0.597 (0.07)	0.626 (0.05)	0.638 (0.1)	0.636 (0.033)	**0.642 (0.033)**	0.640 (0.04)
Residential-Building-Data-Set-1	0.971 (0.06)	0.957 (0.01)	0.973 (0.005)	0.979 (0.003)	0.985 (0.003)	**0.988 (0.01)**
Residential-Building-Data-Set-2	0.973 (0.006)	0.943 (0.006)	0.981 (0.007)	0.978 (0.003)	0.984 (0.003)	**0.985 (0.005)**
slice_localization_data	0.896 (0.001)	0.983 (0.001)	0.966 (0.0)	0.932 (0.0)	**0.986 (0.0)**	0.975 (0.0)
TomsHardware	0.953 (0.003)	0.998 (0.0)	**0.999 (0.0)**	0.957 (0.004)	0.973 (0.002)	0.967 (0.0)
vote-for-clinton	0.255 (0.016)	**0.416 (0.02)**	0.389 (0.02)	0.395 (0.004)	0.353 (0.012)	0.350 (0.01)
wine-quality-red	0.423 (0.023)	0.435 (0.02)	0.389 (0.03)	0.304 (0.011)	0.345 (0.01)	**0.440 (0.02)**
wine-quality-white	0.3 (0.008)	0.459 (0.01)	0.422 (0.01)	0.349 (0.01)	0.318 (0.004)	**0.463 (0.01)**
yacht-hydrodynamics	0.994 (0.003)	0.994 (0.004)	**0.996 (0.002)**	0.991 (0.0)	**0.996 (0.0)**	**0.996 (0.02)**

6 Conclusions and Discussions

In recent years, several studies attempted to build classification trees in which the greedy sub-optimal construction approach is replaced by solving an optimization problem, usually in integer variables. These procedures, while successful against CART, are extremely time-consuming and can only address problems of moderate size. In this paper, we have proposed a new soft pruning approach to build classification trees and trained them using neural networks.

Table 3 Out of sample performance (ROC-AUC) among all methods with maximum depth twelve. The number in parenthesis indicates the standard deviation and the best results are made bold.

Classification Datasets	Regression Tree [8]	Random Forest [9]	OCT [3]	NNCT [4]	XGBoost [14]	Proposed NSNT
acute-nephritis	**1.0 (0.0)**	**1.0 (0.0)**	**1.0 (0.0)**	**1.0 (0.0)**	**1.0 (0.0)**	**1.0 (0.0)**
appendicitis	0.698 (0.139)	0.703 (0.183)	0.659 (0.181)	0.747 (0.147)	0.691 (0.188)	**0.792 (0.081)**
australian	0.797 (0.042)	0.858 (0.027)	0.798 (0.038)	0.860 (0.029)	0.856 (0.025)	**0.862 (0.013)**
breast	0.589 (0.082)	**0.65 (0.058)**	0.569 (0.07)	0.62 (0.058)	0.621 (0.077)	0.62 (0.024)
breast-w	0.944 (0.029)	**0.97 (0.024)**	0.95(0.024)	0.966(0.021)	0.968 (0.02)	0.965 (0.008)
bupa	0.597 (0.075)	0.685 (0.067)	0.597 (0.058)	0.695 (0.066)	0.704 (0.043)	**0.715 (0.038)**
congress-voting	0.945 (0.056)	**0.967 (0.042)**	0.957 (0.038)	0.963 (0.04)	0.961 (0.041)	0.936 (0.018)
credit-a	0.823 (0.045)	0.875 (0.028)	0.811 (0.051)	0.872 (0.035)	**0.878 (0.035)**	0.842 (0.02)
crx	0.821 (0.048)	0.864 (0.035)	0.812 (0.058)	**0.884 (0.042)**	0.883 (0.055)	0.854 (0.013)
cylinder-bands	0.647 (0.09)	0.72 (0.09)	0.642 (0.089)	**0.73 (0.091)**	0.708 (0.051)	0.69 (0.035)
fertility	0.519 (0.189)	0.539 (0.136)	0.546 (0.196)	0.564 (0.149)	0.592 (0.212)	**0.712 (0.089)**
fourclass	0.990 (0.013)	0.993 (0.011)	0.990 (0.012)	0.99 (0.012)	0.988 (0.01)	**0.996 (0.004)**
german	0.642 (0.047)	0.64 (0.025)	0.643 (0.039)	0.663 (0.048)	**0.701 (0.045)**	0.655 (0.022)
heart-c	0.725 (0.073)	0.795 (0.043)	0.733 (0.063)	0.81 (0.044)	**0.814 (0.053)**	0.745 (0.025)
hepatitis	0.695 (0.214)	0.708 (0.212)	0.749 (0.18)	0.792 (0.221)	0.588 (0.209)	**0.803 (0.041)**
hill-valley	0.546 (0.04)	0.581 (0.052)	0.508 (0.032)	0.58 (0.059)	**0.601 (0.059)**	0.552(0.027)
ilpd-indian-liver	0.573 (0.084)	0.563 (0.06)	0.598 (0.071)	**0.600 (0.076)**	0.59 (0.052)	0.537 (0.022)
ion	0.843 (0.043)	0.925 (0.04)	0.846 (0.061)	**0.924 (0.044)**	0.92 (0.017)	0.899 (0.011)
iris	0.935 (0.039)	**0.993 (0.015)**	0.735(0.023)	0.76 (0.013)	0.97 (0.024)	0.973 (0.01)
mammographic	0.772 (0.041)	0.784 (0.033)	0.771 (0.025)	0.786 (0.031)	**0.805 (0.034)**	0.788 (0.009)
monks-1	0.808 (0.128)	0.863 (0.084)	0.84 (0.112)	0.854 (0.106)	0.812 (0.04)	**0.92 (0.062)**
monks-2	**0.764 (0.117)**	0.631 (0.107)	0.754 (0.103)	0.618 (0.14)	0.736 (0.159)	0.663 (0.045)
monks-3	0.901 (0.063)	0.91 (0.087)	0.894 (0.063)	0.901 (0.073)	0.926 (0.087)	**0.957 (0.015)**
parkinsons	0.764 (0.121)	0.856 (0.111)	0.795 (0.123)	0.846 (0.111)	0.878 (0.079)	**0.92 (0.025)**
pima	0.69 (0.056)	0.704 (0.045)	0.679 (0.055)	0.712 (0.061)	0.714 (0.048)	**0.723 (0.025)**
promoters	0.707 (0.147)	0.84 (0.129)	0.783 (0.125)	0.875 (0.083)	**0.89 (0.106)**	0.846 (0.05)
ringnorm	0.788 (0.102)	**0.914 (0.057)**	0.794 (0.076)	0.897 (0.09)	0.889 (0.054)	0.90 (0.029)
sonar	0.726 (0.114)	0.8 (0.072)	**0.885 (0.091)**	0.791 (0.097)	0.878 (0.079)	0.801 (0.05)
spect	0.684 (0.084)	0.707 (0.1)	0.65 (0.074)	0.711 (0.01)	0.701 (0.113)	**0.715 (0.031)**
spectf	0.642 (0.127)	0.643 (0.102)	0.647 (0.1)	0.661 (0.094)	0.629 (0.108)	**0.687 (0.048)**
splice-libsvm	0.904 (0.015)	0.933 (0.014)	0.929 (0.028)	0.934 (0.023)	**0.968 (0.021)**	0.921 (0.013)
tae	0.75 (0.083)	0.754 (0.092)	0.602 (0.029)	0.599 (0.09)	0.708 (0.086)	**0.761 (0.036)**
threenorm	0.69 (0.097)	0.80 (0.047)	0.69 (0.068)	0.79 (0.084)	0.801 (0.021)	**0.803 (0.027)**
tic-tac-toe	0.882 (0.036)	0.923 (0.036)	0.896 (0.034)	**0.925 (0.029)**	0.908 (0.005)	0.902 (0.021)
titanic	0.715 (0.071)	0.719 (0.102)	0.773 (0.105)	0.706 (0.082)	0.794 (0.058)	**0.821 (0.045)**
transfusion	0.638 (0.064)	0.599 (0.061)	**0.619 (0.066)**	0.605 (0.066)	0.605 (0.064)	0.614 (0.02)
twonorm	0.757 (0.071)	0.933 (0.036)	0.813 (0.055)	0.933 (0.026)	0.937 (0.043)	**0.947 (0.032)**
vote	0.924 (0.04)	0.955 (0.026)	0.934 (0.042)	0.954 (0.024)	0.958 (0.029)	**0.971 (0.01)**
vote1	0.832 (0.059)	0.895 (0.032)	0.842 (0.061)	**0.896 (0.047)**	0.883 (0.042)	0.886 (0.013)
wine	0.935 (0.039)	0.943 (0.015)	0.735 (0.023)	0.76 (0.013)	0.970 (0.033)	**0.973 (0.01)**

By replacing the binary decisions with randomized decisions along with neural trees, the resulting sparse framework is smooth and only contains continuous variables, allowing one to use gradient information.

This paper developed an easily interpretable near-optimal sparse neural tree that is asymptotically consistent, near-optimal convergence rate, scalable, and accurate as compared to state-of-the-art models. The model is empirically shown to perform consistently for a wide range of datasets of various sizes. Strong empirical shreds of evidence show that with limited running time, our method outperformed recent benchmarks, achieving significant improvement over XGBoost, OCT, NNCT among many others. Both theoretically and experimentally, we evaluated the performance of the proposed NSNT model. Improving the framework for imbalanced classification problems and survival regression problems can be considered as future research questions.

Conflict of interest

The authors declare that they have no conflict of interest.

References

1. Abadi, M., Barham, P., Chen, J., Chen, Z., Davis, A., Dean, J., Devin, M., Ghemawat, S., Irving, G., Isard, M., et al.: Tensorflow: a system for large-scale machine learning. In: OSDI, vol. 16, pp. 265–283 (2016)
2. Bauer, B., Kohler, M., et al.: On deep learning as a remedy for the curse of dimensionality in nonparametric regression. The Annals of Statistics **47**(4), 2261–2285 (2019)
3. Bertsimas, D., Dunn, J.: Optimal classification trees. Machine Learning **106**(7), 1039–1082 (2017)
4. Bertsimas, D., Dunn, J., Wang, Y.: Near-optimal nonlinear regression trees. Operations Research Letters **49**(2), 201–206 (2021)
5. Biau, G., Scornet, E., Welbl, J.: Neural random forests. Sankhya A pp. 1–40 (2018)
6. Blanquero, R., Carrizosa, E., Molero-Río, C., Morales, D.R.: Sparsity in optimal randomized classification trees. European Journal of Operational Research **284**(1), 255–272 (2020)
7. Blum, A.L., Rivest, R.L.: Training a 3-node neural network is np-complete. Neural Networks **5**(1), 117–127 (1992)
8. Breiman, L.: Classification and regression trees. Routledge (1984)
9. Breiman, L.: Random forests. Machine learning **45**(1), 5–32 (2001)
10. Brent, R.P.: Fast training algorithms for multilayer neural nets. IEEE Transactions on Neural Networks **2**(3), 346–354 (1991)
11. Chakraborty, T., Chakraborty, A.K.: Hellinger net: A hybrid imbalance learning model to improve software defect prediction. IEEE Transactions on Reliability (2020)
12. Chakraborty, T., Chakraborty, A.K., Chattopadhyay, S.: A novel distribution-free hybrid regression model for manufacturing process efficiency improvement. Journal of Computational and Applied Mathematics **362**, 130–142 (2019)
13. Chakraborty, T., Chakraborty, A.K., Murthy, C.: A nonparametric ensemble binary classifier and its statistical properties. Statistics & Probability Letters **149**, 16–23 (2019)
14. Chen, T., Guestrin, C.: Xgboost: A scalable tree boosting system. In: Proceedings of the 22nd ACM SIGKDD, pp. 785–794 (2016)
15. Chen, T., He, T., Benesty, M., Khotilovich, V., Tang, Y., Cho, H., et al.: Xgboost: extreme gradient boosting. R package version 0.4-2 **1**(4) (2015)
16. Chen, Y., Abraham, A., Yang, B.: Feature selection and classification using flexible neural tree. Neurocomputing **70**(1-3), 305–313 (2006)
17. Dua, D., Graff, C., et al.: Uci machine learning repository. UCI (2017)
18. Friedman, J.H.: Multivariate adaptive regression splines. The annals of statistics pp. 1–67 (1991)
19. Frosst, N., Hinton, G.: Distilling a neural network into a soft decision tree. arXiv preprint arXiv:1711.09784 (2017)
20. Guliyev, N.J., Ismailov, V.E.: Approximation capability of two hidden layer feedforward neural networks with fixed weights. Neurocomputing **316**, 262–269 (2018)
21. Györfi, L., Kohler, M., Krzyzak, A., Walk, H.: A distribution-free theory of nonparametric regression. Springer Science & Business Media (2006)
22. Hamers, M., Kohler, M.: A bound on the expected maximal deviation of averages from their means. Statistics & Probability Letters **62**(2), 137–144 (2003)
23. Hornik, K., Stinchcombe, M., White, H.: Multilayer feedforward networks are universal approximators. Neural Networks **2**(5), 359–366 (1989)
24. Hu, X., Rudin, C., Seltzer, M.: Optimal sparse decision trees. Advances in Neural Information Processing Systems (NeurIPS) (2019)
25. Humbird, K.D., Peterson, J.L., McClarren, R.G.: Deep neural network initialization with decision trees. IEEE Transactions on Neural Networks and Learning Systems (2018)

26. Ismailov, V.E.: On the approximation by neural networks with bounded number of neurons in hidden layers. Journal of Mathematical Analysis and Applications **417**(2), 963–969 (2014)
27. Kohler, M.: Nonparametric regression with additional measurement errors in the dependent variable. Journal of Statistical Planning and Inference **136**(10), 3339–3361 (2006)
28. Kohler, M., Krzyżak, A.: Adaptive regression estimation with multilayer feedforward neural networks. Nonparametric Statistics **17**(8), 891–913 (2005)
29. Liaw, A., Wiener, M.: randomforest: Breiman and cutler's random forests for classification and regression. R package version **4**, 6–10 (2015)
30. Lin, J., Zhong, C., Hu, D., Rudin, C., Seltzer, M.: Generalized and scalable optimal sparse decision trees. In: International Conference on Machine Learning, pp. 6150–6160. PMLR (2020)
31. Lugosi, G., Nobel, A.: Consistency of data-driven histogram methods for density estimation and classification. The Annals of Statistics **24**(2), 687–706 (1996)
32. Lugosi, G., Zeger, K.: Nonparametric estimation via empirical risk minimization. IEEE Transactions on information theory **41**(3), 677–687 (1995)
33. Mhaskar, H.N.: Approximation properties of a multilayered feedforward artificial neural network. Advances in Computational Mathematics **1**(1), 61–80 (1993)
34. Mhaskar, H.N., Poggio, T.: Deep vs. shallow networks: An approximation theory perspective. Analysis and Applications **14**(06), 829–848 (2016)
35. Milborrow, M.S.: Package 'earth' (2020)
36. Nobel, A.: Histogram regression estimation using data-dependent partitions. The Annals of Statistics **24**(3), 1084–1105 (1996)
37. Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., Blondel, M., Prettenhofer, P., Weiss, R., Dubourg, V., et al.: Scikit-learn: Machine learning in python. Journal of machine learning research **12**(Oct), 2825–2830 (2011)
38. Rani, A., Foresti, G.L., Micheloni, C.: A neural tree for classification using convex objective function. Pattern Recognition Letters **68**, 41–47 (2015)
39. Rumelhart, D.E., Hinton, G.E., Williams, R.J.: Learning internal representations by error propagation. Tech. rep., California Univ San Diego La Jolla Inst for Cognitive Science (1985)
40. Sakar, A., Mammone, R.J.: Growing and pruning neural tree networks. IEEE Transactions on Computers **42**(3), 291–299 (1993)
41. Sethi, I.K.: Entropy nets: from decision trees to neural networks. Proceedings of the IEEE **78**(10), 1605–1613 (1990)
42. Sethi, I.K.: Neural implementation of tree classifiers. IEEE transactions on systems, man, and cybernetics **25**(8), 1243–1249 (1995)
43. Setiono, R., Leow, W.K.: On mapping decision trees and neural networks. Knowledge-Based Systems **12**(3), 95–99 (1999)
44. Sirat, J., Nadal, J.: Neural trees: a new tool for classification. Network: Computation in Neural Systems **1**(4), 423–438 (1990)
45. Stone, C.J.: Optimal global rates of convergence for nonparametric regression. The Annals of Statistics pp. 1040–1053 (1982)
46. Tanno, R., Arulkumaran, K., Alexander, D., Criminisi, A., Nori, A.: Adaptive neural trees. In: International Conference on Machine Learning, pp. 6166–6175. PMLR (2019)
47. Tsujino, K., Nishida, S.: Implementation and refinement of decision trees using neural networks for hybrid knowledge acquisition. Artificial Intelligence in Engineering **9**(4), 265–276 (1995)
48. Utgoff, P.E.: Perceptron trees: A case study in hybrid concept representations. Connection Science **1**(4), 377–391 (1989)
49. Verwer, S., Zhang, Y.: Learning optimal classification trees using a binary linear program formulation. In: Proceedings of the AAAI Conference on Artificial Intelligence, vol. 33, pp. 1625–1632 (2019)
50. Zhou, Z.H., Chen, Z.Q.: Hybrid decision tree. Knowledge-based systems **15**(8), 515–528 (2002)