Investigating Axis-Aligned Differentiable Trees through Neural Tangent Kernels

Ryuichi Kanoh [1 2] **Mahito Sugiyama** [1 2]

Abstract

Axis-aligned rules are known to induce an important inductive bias in machine learning models such as typical hard decision tree ensembles. However, theoretical understanding of the learning behavior is largely unrevealed due to the discrete nature of rules. To address this issue, we impose the axis-aligned constraint on *differentiable* decision trees, or *soft trees*, which relax the splitting process of decision trees and are trained using the gradient method. The differentiable property enables us to derive their *Neural Tangent Kernel* (NTK) that can analytically describe the training behavior. Two cases are realized: imposing the axis-aligned constraint throughout the entire training process, or only at the initial state. Moreover, we extend the NTK framework to handle various tree architectures simultaneously, and prove that any axis-aligned non-oblivious tree ensemble can be transformed into an axis-aligned oblivious tree ensemble with the same limiting NTK. By excluding non-oblivious trees from the search space, the cost of trial-and-error procedures required for model selection can be massively reduced.

1. Introduction

A *soft tree* is a differentiable model that continuously relaxes the splitting process in a decision tree and is trained using the gradient method. There are various reasons why formulating trees in a differentiable manner is beneficial. Soft tree ensemble models are recognized for their high empirical performance (Kontschieder et al., 2015; Popov et al., 2020; Hazimeh et al., 2020). In addition, unlike hard decision trees, soft tree models can be updated sequentially (Ke et al., 2019) and trained in conjunction with pre-training (Arik & Pfister, 2021), resulting in desirable traits for continuous service deployment in real-world settings. In this study, we consider ensemble learning using soft trees as weak learners.

In general, since all input features are taken into account in each splitting process of soft trees, splitting boundaries are oblique. The *axis-aligned splitting* is considered to be an important inductive bias in typical hard decision trees. Although a number of machine learning models have been proposed that are aware of axis-aligned partitioning (Chang et al., 2022; Humbird et al., 2019), it has not been theoretically clear what properties emerge when the axis-aligned constraints are imposed.

Recently, there has been progress in the theoretical analysis of soft tree ensembles (Kanoh & Sugiyama, 2022; 2023) using the *Neural Tangent Kernel* (NTK) (Jacot et al., 2018). The NTK framework provides analytical descriptions of ensemble learning with infinitely many soft trees. However, the current analysis is limited to soft trees with oblique splitting and cannot directly incorporate the axis-aligned constraint. In this paper, we extend the NTK concept to the axis-aligned soft tree ensembles to uncover the theoretical properties of the axis-aligned constraint.

2. Preliminary

2.1. Soft Tree Ensembles

We formulate regression using soft decision trees based on Kontschieder et al. (2015). Let $\boldsymbol{x} \in \mathbb{R}^{F \times N}$ be input data consisting of N samples with F features. Assume that there are M soft decision trees and each tree has $\mathcal{N}$ splitting nodes and $\mathcal{L}$ leaf nodes. We denote parameters of an m-th soft decision tree, where $m \in [M] = \{1, \dots, M\}$, as $\boldsymbol{w}_m \in \mathbb{R}^{F \times \mathcal{N}}$ and $\boldsymbol{b}_m \in \mathbb{R}^{1 \times \mathcal{N}}$ for splitting nodes and $\boldsymbol{\pi}_m \in \mathbb{R}^{1 \times \mathcal{L}}$ for leaf nodes.

Unlike typical hard decision trees, the leaf nodes $\ell \in [\mathcal{L}] = \{1, \dots, \mathcal{L}\}$ in soft decision trees hold values $\mu_{m,\ell}(\boldsymbol{x}_i, \boldsymbol{w}_m)$ that represent the probability of the input data reaching that leaf: $\mu_{m,\ell}(\boldsymbol{x}_i, \boldsymbol{w}_m, \boldsymbol{b}_m) = \prod_{n=1}^{\mathcal{N}} \sigma(\boldsymbol{w}_{m,n}^\top \boldsymbol{x}_i + \beta b_{m,n})^{\mathbb{1}_{\ell \swarrow n}} (1 - \sigma(\boldsymbol{w}_{m,n}^\top \boldsymbol{x}_i + \beta b_{m,n}))^{\mathbb{1}_{n \searrow \ell}}$, where $\boldsymbol{w} = (\boldsymbol{w}_1, \dots, \boldsymbol{w}_m)$, $\boldsymbol{b} = (\boldsymbol{b}_1, \dots, \boldsymbol{b}_m)$, and $\boldsymbol{\pi} = (\boldsymbol{\pi}_1, \dots, \boldsymbol{\pi}_m)$ correspond to feature selection, splitting threshold, and prediction value of the leaf, respectively, in a typical decision tree, and $\mathbb{1}_{\ell \swarrow n}(\mathbb{1}_{n \searrow \ell})$ is a binary function that returns 1 if the ℓ-th leaf is on the left (right) side of node n, and 0 other-

[1] National Institute of Informatics [2] The Graduate University for Advanced Studies, SOKENDAI. Correspondence to: Ryuichi Kanoh <kanoh@nii.ac.jp>.

Published at the Differentiable Almost Everything Workshop of the 40^{th} International Conference on Machine Learning, Honolulu, Hawaii, USA. July 2023. Copyright 2023 by the author(s).

wise. Parameters $\boldsymbol{w}$, $\boldsymbol{b}$, and $\boldsymbol{\pi}$ are randomly initialized using independent and identically distributed normal distributions with mean 0 and variance 1, and updated using gradient descent. The factor $\beta \in \mathbb{R}^+$ is a parameter that allows us to tune the influence of the bias on the training.

The internal nodes operate using a decision function that resembles the sigmoid. In this paper, we use the scaled error function (Kanoh & Sugiyama, 2022; 2023): $\sigma(c) = \frac{1}{2}\operatorname{erf}(\alpha c) + \frac{1}{2} = \frac{1}{2}\left(\frac{2}{\sqrt{\pi}}\int_0^{\alpha c} e^{-t^2}\,\mathrm{d}t\right) + \frac{1}{2}$ for $c \in \mathbb{R}$. As the scaling factor $\alpha \in \mathbb{R}^+$ (Frosst & Hinton, 2017) tends towards infinity, sigmoid-like decision functions become step functions that correspond to (hard) Boolean operation.

The function $f_m : \mathbb{R}^F \times \mathbb{R}^{F \times \mathcal{N}} \times \mathbb{R}^{1 \times \mathcal{N}} \times \mathbb{R}^{1 \times \mathcal{L}} \to \mathbb{R}$ that returns the prediction of the m-th tree is given by the weighted sum of leaf-specific parameters $\pi_{m,\ell}$, weighted by the probability that the input data $\boldsymbol{x}_i$ reaches each leaf:

$$f_m(\boldsymbol{x}_i, \boldsymbol{w}_m, \boldsymbol{b}_m, \boldsymbol{\pi}_m) = \sum_{\ell=1}^{\mathcal{L}} \pi_{m,\ell}\,\mu_{m,\ell}(\boldsymbol{x}_i, \boldsymbol{w}_m, \boldsymbol{b}_m). \quad (1)$$

Furthermore, the function $f : \mathbb{R}^F \times \mathbb{R}^{M \times F \times \mathcal{N}} \times \mathbb{R}^{M \times 1 \times \mathcal{N}} \times \mathbb{R}^{M \times 1 \times \mathcal{L}} \to \mathbb{R}$ that returns the output of the model using M trees is formulated as follows:

$$f(\boldsymbol{x}_i, \boldsymbol{w}, \boldsymbol{b}, \boldsymbol{\pi}) = \frac{1}{\sqrt{M}} \sum_{m=1}^{M} f_m(\boldsymbol{x}_i, \boldsymbol{w}_m, \boldsymbol{b}_m, \boldsymbol{\pi}_m). \quad (2)$$

2.2. Neural Tangent Kernels

We introduce the NTK based on the gradient flow using training data $\boldsymbol{x} \in \mathbb{R}^{F \times N}$, the prediction target $\boldsymbol{y} \in \mathbb{R}^N$, trainable parameters $\boldsymbol{\theta}_\tau \in \mathbb{R}^P$ at time τ, and an arbitrary model function $g(\boldsymbol{x}, \boldsymbol{\theta}_\tau) : \mathbb{R}^{F \times N} \times \mathbb{R}^P \to \mathbb{R}^N$. With the learning rate η and the mean squared error loss function L, the gradient flow equation is given as

$$\frac{\partial \boldsymbol{\theta}_\tau}{\partial \tau} = -\eta \frac{\partial L(\boldsymbol{\theta}_\tau)}{\partial \boldsymbol{\theta}_\tau} = -\eta \frac{\partial g(\boldsymbol{x}, \boldsymbol{\theta}_\tau)}{\partial \boldsymbol{\theta}_\tau}(g(\boldsymbol{x}, \boldsymbol{\theta}_\tau) - \boldsymbol{y}). \quad (3)$$

Considering the formulation of the gradient flow in the function space using Equation (3), we obtain

$$\begin{aligned}\frac{\partial g(\boldsymbol{x}, \boldsymbol{\theta}_\tau)}{\partial \tau} &= \frac{\partial g(\boldsymbol{x}, \boldsymbol{\theta}_\tau)}{\partial \boldsymbol{\theta}_\tau}^\top \frac{\partial \boldsymbol{\theta}_\tau}{\partial \tau} \\ &= -\eta \underbrace{\frac{\partial g(\boldsymbol{x}, \boldsymbol{\theta}_\tau)}{\partial \boldsymbol{\theta}_\tau}^\top \frac{\partial g(\boldsymbol{x}, \boldsymbol{\theta}_\tau)}{\partial \boldsymbol{\theta}_\tau}}_{\text{Neural Tangent Kernel}}(g(\boldsymbol{x}, \boldsymbol{\theta}_\tau) - \boldsymbol{y}).\end{aligned} \quad (4)$$

Here, a matrix called the NTK, which has a shape of $\mathbb{R}^{N \times N}$, appears. In this paper, the (i,j)-th component of the matrix at τ is denoted as $\widehat{\Theta}_\tau(\boldsymbol{x}_i, \boldsymbol{x}_j)$.

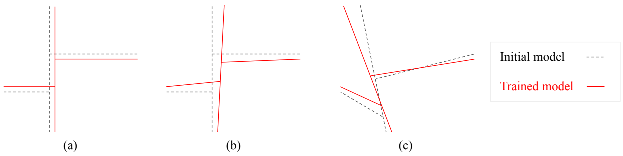

Figure 1. Schematics of training procedures. (a): AAA, Always Axis-Aligned, during training, (b): AAI, Axis-Aligned at Initialization, but not during training, (c): Oblique splitting conducted by typical soft trees.

From Equation (4), if the NTK does not change during training, the formulation of the gradient flow in the function space becomes a simple ordinary differential equation, and it becomes possible to analytically calculate how the model's output changes during training. When the NTK is positive definite, it is known that the kernel does not change from its initial value during the gradient descent with an infinitesimal step size when considering an infinite number of soft binary trees (Lee et al., 2019; Kanoh & Sugiyama, 2022; 2023) under the formulation described in Section 2.1.

2.3. Setup on Axis-Aligned Splitting

Since $\boldsymbol{w}_{m,n}$ is initialized randomly, the splitting is generally oblique. In this study, we analyze the NTK when some of the initial values of $\boldsymbol{w}_{m,n}$ are set to be zero. With this setting, the corresponding features are eliminated from consideration of the splitting direction. This is technically not straightforward, as Gaussian random initialization is generally assumed in the existing NTK approaches. In particular, we consider the axis-aligned case where only one feature is used for splitting. We conduct a theoretical analysis of two cases: one where the parameters with zero initialization are not updated during training, as illustrated in Figure 1(a), and the other where they are updated, as shown in Figure 1(b). These two cases are referred to as AAA ("A"lways "A"xis-"A"ligned) and AAI ("A"xis-"A"ligned at "I"nitialization, but not during training) in this paper.

The features to be assigned to each node need to be predetermined before training. This is different from typical decision trees, which search for features to be used for splitting in training. However, predetermining splitting features is used in practice. For example, Extremely Randomized Trees (Geurts et al., 2006) implemented in scikit-learn (Pedregosa et al., 2011) employs such predetermination.

3. Theoretical Results

3.1. The NTK Induced by Axis-Aligned Soft Trees

For an input vector $\boldsymbol{x}_i$, let $x_{i,s} \in \mathbb{R}$ be the sth component of $\boldsymbol{x}_i$. For both AAA and AAI conditions, at initialization,

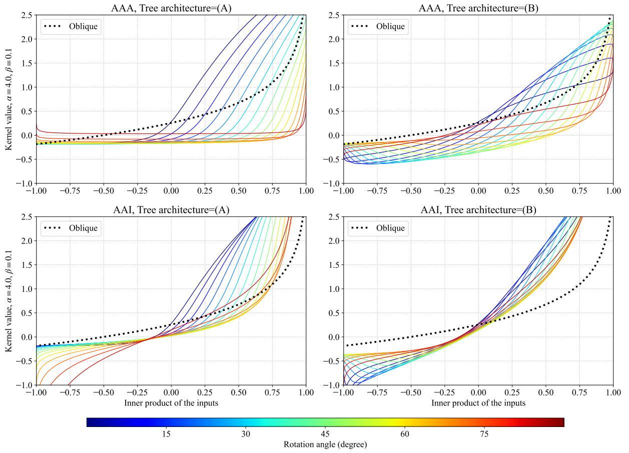

Figure 2. The rotation angle dependency of $\Theta(\boldsymbol{x}_i, \boldsymbol{x}_j)$. Different training procedures, AAA and AAI, are listed vertically, and two settings of tree architectures are listed horizontally. Dotted lines show the limiting NTK induced by typical oblique soft tree ensembles defined in (Kanoh & Sugiyama, 2022; 2023), which is rotationally invariant.

we can obtain the NTK induced by axis-aligned soft tree ensembles in a closed-form as the number of trees $M \to \infty$.

Theorem 3.1. *Assume all M trees have the same soft tree architecture. Let $\{a_1, \cdots, a_\ell, \cdots, a_\mathcal{L}\}$ denote the set of decomposed paths of the trees from the root to the leaves, and let $h(a_\ell) \subset \mathbb{N}$ be the set of feature indices used in each split of the input path a_ℓ. For any binary tree architectures, as the number M of trees goes to infinity, the NTK for an ensemble of axis-aligned soft trees converges in probability to the deterministic kernel:*

$$\begin{aligned}\Theta(\boldsymbol{x}_i, \boldsymbol{x}_j) &:= \lim_{M\to\infty} \widehat{\Theta}_0(\boldsymbol{x}_i, \boldsymbol{x}_j) \\ &= \underbrace{\sum_{\ell=1}^{\mathcal{L}} \sum_{s\in h(a_\ell)} \left(\Sigma_{\{i,j\},s} \dot{\mathcal{T}}_{\{i,j\},s} \prod_{t \in h(a_\ell)\setminus\{s\}} \mathcal{T}_{\{i,j\},t} \right)}_{\text{contribution from internal nodes}} \\ &\quad + \underbrace{\sum_{\ell=1}^{\mathcal{L}} \prod_{u\in h(a_\ell)} \mathcal{T}_{\{i,j\},u}}_{\text{contribution from leaves}}, \qquad (5)\end{aligned}$$

where $\Sigma_{\{i,j\},s} = x_{i,s}x_{j,s} + \beta^2$ if AAA is used, or $\Sigma_{\{i,j\},s} = \boldsymbol{x}_i^\top \boldsymbol{x}_j + \beta^2$ if AAI is used. By denoting $x_{i,s}x_{j,s} + \beta^2$ as $A_{\{i,j\},s}$, $\mathcal{T}_{\{i,j\},s} = \frac{1}{2\pi}\arcsin\left(\frac{\alpha^2 A_{\{i,j\},s}}{\sqrt{(\alpha^2 A_{\{i,i\},s}+0.5)(\alpha^2 A_{\{j,j\},s}+0.5)}}\right) + \frac{1}{4}$, $\dot{\mathcal{T}}_{\{i,j\},s} = \frac{\alpha^2}{\pi}\frac{1}{\sqrt{\left(1+2\alpha^2 A_{\{i,i\},s}\right)\left(1+2\alpha^2 A_{\{j,j\},s}\right) - 4\alpha^4 A_{\{i,j\},s}^2}}$.

The difference between AAA and AAI is whether partial features of inputs are used or all features are used in $\Sigma_{\{i,j\},s}$.

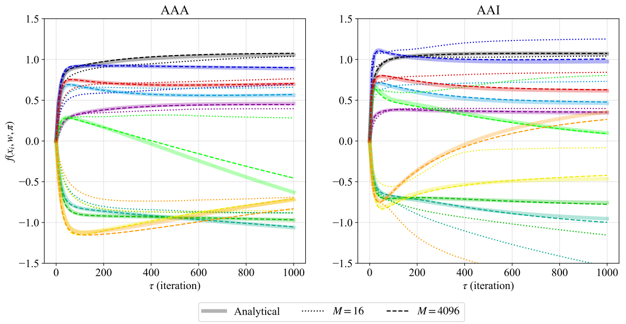

Figure 3. Output dynamics of test data points for axis-aligned soft tree ensembles with two conditions. (Left): AAA, Always Axis-Aligned, during training, (Right): AAI, Axis-Aligned at Initialization, but not during training. Each data point is represented by a different line color. The left and right figures are created using exactly the same training and test data.

In AAA, the impact of features that are not used for splitting is completely ignored, while in AAI, the kernel is affected by all features through the inner product of the inputs.

Figure 2 shows the visualization of $\Theta(\boldsymbol{x}_i, \boldsymbol{x}_j)$. We set $\alpha = 4.0$ and $\beta = 0.1$. We calculated the kernel values for two rotated vectors: $\boldsymbol{x}_i = (\cos(\omega), \sin(\omega))$, $\boldsymbol{x}_j = (\cos(\omega+\phi), \sin(\omega+\phi))$ where $\omega \in [0, \pi/2]$, and $\phi \in [0, \pi]$. The line colors show ω, and the x-axis shows ϕ. We use an oblivious tree with depth 2, where we use the first feature at both depths 1 and 2 for the architecture (A) (left column), and we use the first feature at depth 1 and the second feature at depth 2 for (B) (right column). We can see that the invariance with respect to rotational transformations for input has disappeared. This is different from the NTK induced by typical soft tree ensembles, shown by the dotted lines. Moreover, when we compare left and right plots, we can see that the kernel varies depending on the features used for splitting.

For both AAA and AAI, as the number of trees increases, the trajectory obtained analytically from the limiting kernel and the trajectory during gradient descent training become more identical, as shown in Figure 3. This demonstrates the validity of using the NTK framework to analyze the training behavior. For our experiment, we consider an ensemble of oblivious trees with $\alpha = 4.0, \beta = 0.1$, where the first feature is used for splitting at depth 1 and the second feature at depth 2. This model is trained using full-batch gradient descent with a learning rate of 0.1, and the initial outputs are shifted to zero (Chizat et al., 2019). The training and test datasets contain 10 randomly generated $F = 2$ dimensional points each. The prediction targets are also randomly generated. Based on Lee et al. (2019), to derive analytical trajectories, we use the limiting kernel

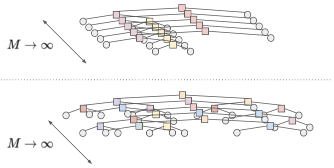

Figure 4. Ensemble trees with different tree architectures. The color of tree nodes indicates a feature used for splitting.

(Theorem 3.1), as $f(\boldsymbol{v}, \boldsymbol{\theta}_\tau) = \boldsymbol{H}(\boldsymbol{v}, \boldsymbol{x})\boldsymbol{H}(\boldsymbol{x}, \boldsymbol{x})^{-1}(\boldsymbol{I} - \exp[-\eta \boldsymbol{H}(\boldsymbol{x}, \boldsymbol{x})\tau])\boldsymbol{y}$, where $\boldsymbol{H}(\boldsymbol{x}, \boldsymbol{x}') \in \mathbb{R}^{N \times N'}$ denote the limiting NTK matrix for two input matrices, and $\boldsymbol{I}$ represent an identity matrix. The input vector $\boldsymbol{v} \in \mathbb{R}^F$ is arbitrary, and the training dataset and targets are denoted by $\boldsymbol{x} \in \mathbb{R}^{F \times N}$ and $\boldsymbol{y} \in \mathbb{R}^N$, respectively. The behavior of the prediction trajectory changes depending on the configurations (AAA or AAI), even when exactly the same training and test data are used.

3.2. The NTK Induced by Ensembles of Various Trees

In the previous subsections, we have assumed that soft tree ensembles are composed of weak learners with identical structures, as shown in the upper side of Figure 4. However, it can be more practical if the tree structure and features used for splitting vary in an ensemble, as shown in the bottom side of Figure 4. To address this issue, we theoretically analyze ensembles with various tree architectures mixed together. Assuming the existence of an infinite number of trees in an ensemble, the NTK can be computed analytically if the amount (ratio) of each structure in the ensemble is known.

Proposition 3.2. *Let a function $p(\boldsymbol{x}_i, \boldsymbol{\theta}_\tau)$ be the sum of two model functions $q(\boldsymbol{x}_i, \boldsymbol{\theta}'_\tau)$ and $r(\boldsymbol{x}_i, \boldsymbol{\theta}''_\tau)$, where $\boldsymbol{\theta}'_\tau \in \mathbb{R}^{P'}$, $\boldsymbol{\theta}''_\tau \in \mathbb{R}^{P''}$ are trainable parameters, and $\boldsymbol{\theta}$ is the concatenation of trainable parameters $\boldsymbol{\theta}'_\tau$ and $\boldsymbol{\theta}''_\tau$. For any input pair $\boldsymbol{x}_i$ and $\boldsymbol{x}_j$, the NTK induced by p is equal to the sum of the NTKs of q and r: $\widehat{\Theta}^{(p)}_\tau(\boldsymbol{x}_i, \boldsymbol{x}_j) = \widehat{\Theta}^{(q)}_\tau(\boldsymbol{x}_i, \boldsymbol{x}_j) + \widehat{\Theta}^{(r)}_\tau(\boldsymbol{x}_i, \boldsymbol{x}_j)$.*

For example, let q and r be functions that represent perfect binary tree ensemble models with a depth of 1 and 2, respectively. In this case, the NTK induced by trees with a depth of 1 and 2 is the sum of the NTK induced by trees with a depth of 1 and the NTK induced by trees with a depth of 2.

3.3. Insight

The *oblivious tree* architecture is a practical design where the decision rules for tree splitting are shared across the same depth. This approach reduces the number of required splitting calculations from an exponential time complexity of $\mathcal{O}(2^D)$ to a linear time complexity of $\mathcal{O}(D)$, where D represents the depth of the perfect binary tree. This property makes the oblivious tree structure a popular choice in open-source libraries such as CatBoost (Prokhorenkova et al., 2018) and NODE (Popov et al., 2020). Kanoh & Sugiyama (2022; 2023) demonstrated that parameter sharing used in oblivious trees does not affect the NTK of soft tree ensembles. However, their analysis does not give any insight if only specific features are used for each splitting node.

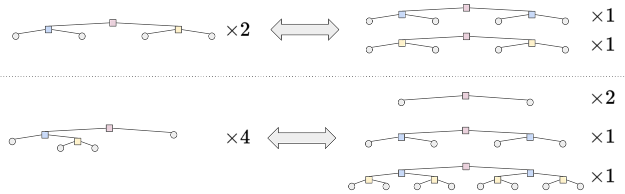

Figure 5. Conversion to oblivious trees inducing exactly the same NTK. The color of tree nodes indicates a feature used for splitting.

With Theorem 3.1 and Proposition 3.2, we show that we can always convert axis-aligned non-oblivious tree ensembles into axis-aligned oblivious tree ensembles that induce exactly the same limiting NTK:

Proposition 3.3. *Let D be the maximum depth of a single tree. For any axis-aligned tree ensembles, a set of oblivious trees that induce exactly the same limiting NTK can be always constructed using 2^{D-1} copies of the same tree.*

Figure 5 shows examples with $D = 2$ or 3. This insight supports the validity of using oblivious trees when using axis-aligned soft trees, as in Chang et al. (2022). Creating copies multiplies the NTK values by a constant factor, but theoretically, this does not have a significant impact. As Equation (4) shows, even if two kernels differ only up to a constant, adjusting the learning rate η can make their training behavior exactly the same.

Although various trial-and-error processes are necessary for model selection to determine features used at each node, this finding can reduce the number of processes by excluding non-oblivious trees from the search space.

4. Conclusion

In this paper, we have formulated the NTK induced by the axis-aligned soft tree ensembles, and we have succeeded in describing the analytical training trajectory. We have theoretically analyzed two scenarios, one where the axis-aligned constraint is applied throughout the training process, and the other where the initial model is axis-aligned and training proceeds without any constraints. We have also presented a theoretical framework to deal with non-identical tree structures simultaneously and used it to provide theoretical support for the validity of using oblivious trees.

References

Arik, S. and Pfister, T. TabNet: Attentive Interpretable Tabular Learning. *Proceedings of the AAAI Conference on Artificial Intelligence*, 2021.

Chang, C.-H., Caruana, R., and Goldenberg, A. NODE-GAM: Neural generalized additive model for interpretable deep learning. In *International Conference on Learning Representations*, 2022.

Chizat, L., Oyallon, E., and Bach, F. On Lazy Training in Differentiable Programming. In *Advances in Neural Information Processing Systems*, 2019.

Frosst, N. and Hinton, G. E. Distilling a Neural Network Into a Soft Decision Tree. *CoRR*, 2017.

Geurts, P., Ernst, D., and Wehenkel, L. Extremely Randomized Trees. *Machine Learning*, 2006.

Hazimeh, H., Ponomareva, N., Mol, P., Tan, Z., and Mazumder, R. The Tree Ensemble Layer: Differentiability meets Conditional Computation. In *Proceedings of the 37th International Conference on Machine Learning*, 2020.

Humbird, K. D., Peterson, J. L., and Mcclarren, R. G. Deep Neural Network Initialization With Decision Trees. *IEEE Transactions on Neural Networks and Learning Systems*, 2019.

Jacot, A., Gabriel, F., and Hongler, C. Neural Tangent Kernel: Convergence and Generalization in Neural Networks. In *Advances in Neural Information Processing Systems*, 2018.

Kanoh, R. and Sugiyama, M. A Neural Tangent Kernel Perspective of Infinite Tree Ensembles. In *International Conference on Learning Representations*, 2022.

Kanoh, R. and Sugiyama, M. Analyzing Tree Architectures in Ensembles via Neural Tangent Kernel. In *International Conference on Learning Representations*, 2023.

Ke, G., Xu, Z., Zhang, J., Bian, J., and Liu, T.-Y. DeepGBM: A Deep Learning Framework Distilled by GBDT for Online Prediction Tasks. In *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, 2019.

Kontschieder, P., Fiterau, M., Criminisi, A., and Bulò, S. R. Deep Neural Decision Forests. In *IEEE International Conference on Computer Vision*, 2015.

Lee, J., Xiao, L., Schoenholz, S., Bahri, Y., Novak, R., Sohl-Dickstein, J., and Pennington, J. Wide Neural Networks of Any Depth Evolve as Linear Models Under Gradient Descent. In *Advances in Neural Information Processing Systems*, 2019.

Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., Blondel, M., Prettenhofer, P., Weiss, R., Dubourg, V., Vanderplas, J., Passos, A., Cournapeau, D., Brucher, M., Perrot, M., and Duchesnay, E. Scikit-learn: Machine Learning in Python. *Journal of Machine Learning Research*, 2011.

Popov, S., Morozov, S., and Babenko, A. Neural Oblivious Decision Ensembles for Deep Learning on Tabular Data. In *International Conference on Learning Representations*, 2020.

Prokhorenkova, L., Gusev, G., Vorobev, A., Dorogush, A. V., and Gulin, A. CatBoost: unbiased boosting with categorical features. In *Advances in Neural Information Processing Systems*, 2018.

A. The NTK Induced by Typical Soft Trees

The NTK induced by a typical soft tree ensemble with infinitely many trees is known to be obtained in closed-form at initialization.

Theorem A.1 ((Kanoh & Sugiyama, 2022; 2023))**.** *Assume all M trees have the same soft tree architecture. Let $\mathcal{Q}:\mathbb{N}\to\mathbb{N}\cup\{0\}$ be a function that takes the depth as input and returns the number of leaves connected to internal nodes at that depth. For any given tree architecture, as the number of trees in an ensemble of soft trees approaches infinity, the NTK converges in probability to a deterministic kernel:*

$$\begin{aligned}\Theta^{\text{Oblique}}(\boldsymbol{x}_i,\boldsymbol{x}_j) &\coloneqq \lim_{M\to\infty}\widehat{\Theta}_0^{\text{Oblique}}(\boldsymbol{x}_i,\boldsymbol{x}_j)\\ &=\sum_{d=1}^{D}\mathcal{Q}(d)\left(\underbrace{d\,\Sigma_{\{i,j\}}\mathcal{T}_{\{i,j\}}^{d-1}\dot{\mathcal{T}}_{\{i,j\}}}_{\text{contribution from internal nodes}}+\underbrace{\mathcal{T}_{\{i,j\}}^{d}}_{\text{contribution from leaves}}\right),\end{aligned} \tag{A.1}$$

where

$$\Sigma_{\{i,j\}}=\boldsymbol{x}_i^\top\boldsymbol{x}_j+\beta^2, \tag{A.2}$$

$$\mathcal{T}_{\{i,j\}}=\frac{1}{2\pi}\arcsin\left(\frac{\alpha^2\Sigma_{\{i,j\}}}{\sqrt{(\alpha^2\Sigma_{\{i,i\}}+0.5)(\alpha^2\Sigma_{\{j,j\}}+0.5)}}\right)+\frac{1}{4}, \tag{A.3}$$

$$\dot{\mathcal{T}}_{\{i,j\}}=\frac{\alpha^2}{\pi}\frac{1}{\sqrt{\left(1+2\alpha^2\Sigma_{\{i,i\}}\right)\left(1+2\alpha^2\Sigma_{\{j,j\}}\right)-4\alpha^4\Sigma_{\{i,j\}}^2}}. \tag{A.4}$$

This kernel has rotational invariance with respect to input data.

B. Proofs

B.1. Proof of Theorem 3.1

Proof. Based on the independence of parameters at each leaf and the symmetry of the decision function, Kanoh & Sugiyama (2023) showed that the NTK induced by arbitrary soft tree ensembles can be decomposed into the sum of the NTKs induced by the rule sets, which are constructed by paths from the tree root to leaves. This property of the independence of parameters at each leaf and the symmetry of the decision function also holds in our formulation (Section 2.1). Therefore, we formulate the NTK induced by rule sets and use it to derive the NTK induced by axis-aligned soft tree ensembles.

For simplicity, first we consider the case where $\beta=0$. Let D be the depth of a rule set, which is a path from the root to a leaf ℓ. We consider the contribution from internal nodes $\Theta^{(D,\text{Rule,nodes})}$ and the contribution from leaves $\Theta^{(D,\text{Rule,leaves})}$ separately, such that

$$\Theta^{(D,\text{Rule})}\left(\boldsymbol{x}_i,\boldsymbol{x}_j\right)=\Theta^{(D,\text{Rule,nodes})}\left(\boldsymbol{x}_i,\boldsymbol{x}_j\right)+\Theta^{(D,\text{Rule,leaves})}\left(\boldsymbol{x}_i,\boldsymbol{x}_j\right). \tag{B.1}$$

As for internal nodes, when we consider the axis-aligned case (Section 2.3), only a single parameter in $\boldsymbol{w}_{m,n}$ is non-zero. When calculating the NTK as shown in Equation (4), the parameter derivatives in terms of trainable parameters are considered. In the cases of AAA and AAI, they are given as follows:

$$\frac{f^{(D,\text{Rule})}\left(\boldsymbol{x}_i,\boldsymbol{w},\boldsymbol{\pi}\right)}{\partial w_{m,n,k}}=\frac{1}{\sqrt{M}}x_{i,s_n}\dot{\sigma}(w_{m,n,k}x_{i,k_n})f_m^{(D-1,\text{Rule})}\left(\boldsymbol{x}_i,\boldsymbol{w}_{m,-n},\boldsymbol{\pi}_m\right), \qquad (\text{AAA}) \tag{B.2}$$

$$\frac{f^{(D,\text{Rule})}\left(\boldsymbol{x}_i,\boldsymbol{w},\boldsymbol{\pi}\right)}{\partial \boldsymbol{w}_{m,n}}=\frac{1}{\sqrt{M}}\boldsymbol{x}_i\dot{\sigma}(w_{m,n,k}x_{i,k_n})f_m^{(D-1,\text{Rule})}\left(\boldsymbol{x}_i,\boldsymbol{w}_{m,-n},\boldsymbol{\pi}_m\right), \qquad (\text{AAI}) \tag{B.3}$$

where x_{i,k_n} and $w_{m,n,k}$ are single features that are used in the n-th node and a single node parameter that corresponds to x_{i,k_n}, respectively. $\boldsymbol{w}_{m,-n}$ denotes the internal node parameter matrix except for the parameters of the node n. Since there are D possible locations for n, we obtain

$$\Theta^{(D,\text{Rule,nodes})}\left(\boldsymbol{x}_i, \boldsymbol{x}_j\right) = \sum_{s \in h(a_\ell)} \left(\Sigma_{\{i,j\},s} \dot{\mathcal{T}}_{\{i,j\},s} \prod_{t \in h(a_\ell) \setminus \{s\}} \mathcal{T}_{\{i,j\},t} \right), \tag{B.4}$$

where

$$\begin{aligned}
&\mathbb{E}_m \left[f_m^{(D,\text{Rule})}\left(\boldsymbol{x}_i, \boldsymbol{w}_m, \boldsymbol{\pi}_m\right) f_m^{(D,\text{Rule})}\left(\boldsymbol{x}_j, \boldsymbol{w}_m, \boldsymbol{\pi}_m\right) \right] \\
&= \mathbb{E}_m \left[\underbrace{\sigma(\boldsymbol{w}_{m,1}^\top \boldsymbol{x}_i)\sigma(\boldsymbol{w}_{m,1}^\top \boldsymbol{x}_j)}_{\to \mathcal{T}_{\{i,j\},s_1}} \underbrace{\sigma(\boldsymbol{w}_{m,2}^\top \boldsymbol{x}_i)\sigma(\boldsymbol{w}_{m,2}^\top \boldsymbol{x}_j)}_{\to \mathcal{T}_{\{i,j\},s_2}} \cdots \underbrace{\sigma(\boldsymbol{w}_{m,D}^\top \boldsymbol{x}_i)\sigma(\boldsymbol{w}_{m,D}^\top \boldsymbol{x}_j)}_{\to \mathcal{T}_{\{i,j\},s_D}} \underbrace{\pi_{m,1}^2}_{\to 1} \right] \\
&= \prod_{t \in h(a_\ell)} \mathcal{T}_{\{i,j\},t}
\end{aligned} \tag{B.5}$$

is used. Here, the symbol "→" denotes the expected value of the corresponding term will take.

For leaves, Since

$$\frac{f^{(D,\text{Rule})}\left(\boldsymbol{x}_i, \boldsymbol{w}, \boldsymbol{\pi}\right)}{\partial \pi_{m,1}} = \frac{1}{\pi_{m,1}\sqrt{M}} f_m^{(D,\text{Rule})}\left(\boldsymbol{x}_i, \boldsymbol{w}_m, \boldsymbol{\pi}_m\right), \tag{B.6}$$

we have

$$\Theta^{(D,\text{Rule,leaves})}\left(\boldsymbol{x}_i, \boldsymbol{x}_j\right) = \prod_{u \in h(a_\ell)} \mathcal{T}_{\{i,j\},u}. \tag{B.7}$$

Combining Equation (B.5) and Equation (B.7), we obtain

$$\Theta^{(D,\text{Rule,nodes})}\left(\boldsymbol{x}_i, \boldsymbol{x}_j\right) = \sum_{s \in h(a_\ell)} \left(\Sigma_{\{i,j\},s} \dot{\mathcal{T}}_{\{i,j\},s} \prod_{t \in h(a_\ell) \setminus \{s\}} \mathcal{T}_{\{i,j\},t} \right) + \prod_{u \in h(a_\ell)} \mathcal{T}_{\{i,j\},u}. \tag{B.8}$$

When we sum up this NTK over multiple rule sets constructed by multiple leaves, it becomes the NTK of the axis-aligned soft tree ensembles:

$$\Theta(\boldsymbol{x}_i, \boldsymbol{x}_j) = \sum_{\ell=1}^{\mathcal{L}} \left(\sum_{s \in h(a_\ell)} \Sigma_{\{i,j\},s} \dot{\mathcal{T}}_{\{i,j\},s} \prod_{t \in h(a_\ell) \setminus \{s\}} \mathcal{T}_{\{i,j\},t} + \prod_{u \in h(a_\ell)} \mathcal{T}_{\{i,j\},u} \right). \tag{B.9}$$

Up until this point, we have been considering the case where $\beta = 0$. It is straightforward to take the case $\beta \neq 0$ into account because, in the case of soft tree ensemble, the bias term can be represented by using an extra feature that takes a constant value β as input. This allows us to generally express the bias term by adding β^2 to $\Sigma_{\{i,j\},s}$.

□

B.2. Proof of Proposition 3.2

Proof. The NTK induced by this model can be decomposed into the sum of the NTKs of each tree architecture as follows:

$$\begin{aligned}
\widehat{\Theta}_\tau^{(p)}(\boldsymbol{x}_i, \boldsymbol{x}_j) &= \left\langle \frac{\partial(q(\boldsymbol{x}_i, \boldsymbol{\theta}_1) + r(\boldsymbol{x}_i, \boldsymbol{\theta}_2))}{\partial \boldsymbol{\theta}}, \frac{\partial(q(\boldsymbol{x}_j, \boldsymbol{\theta}_1) + r(\boldsymbol{x}_j, \boldsymbol{\theta}_2))}{\partial \boldsymbol{\theta}} \right\rangle \\
&= \underbrace{\left\langle \frac{\partial q(\boldsymbol{x}_i, \boldsymbol{\theta}_1)}{\partial \boldsymbol{\theta}_1}, \frac{\partial q(\boldsymbol{x}_j, \boldsymbol{\theta}_1)}{\partial \boldsymbol{\theta}_1} \right\rangle}_{\widehat{\Theta}_\tau^{(q)}(\boldsymbol{x}_i, \boldsymbol{x}_j)} + \underbrace{\left\langle \frac{\partial r(\boldsymbol{x}_i, \boldsymbol{\theta}_2)}{\partial \boldsymbol{\theta}_2}, \frac{\partial r(\boldsymbol{x}_j, \boldsymbol{\theta}_2)}{\partial \boldsymbol{\theta}_2} \right\rangle}_{\widehat{\Theta}_\tau^{(r)}(\boldsymbol{x}_i, \boldsymbol{x}_j)}.
\end{aligned} \tag{B.10}$$

By repeatedly using this property, the proposition can treat any number of sub-models. □

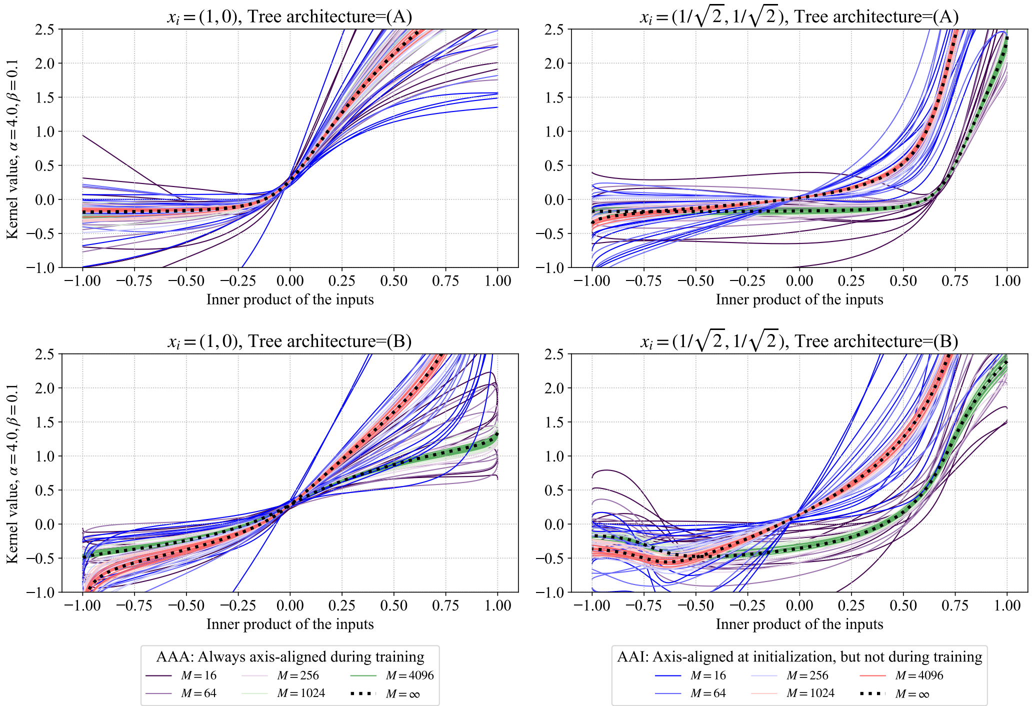

Figure A.1. An empirical demonstration of convergence of $\widehat{\Theta}_0(\boldsymbol{x}_i, \boldsymbol{x}_j)$ to the fixed limit $\Theta(\boldsymbol{x}_i, \boldsymbol{x}_j)$ as M increases. Two cases using different features are listed vertically, and two settings of vectors for computing the kernel are listed horizontally. In the upper left figure, the two cases overlap.

B.3. Proof of Proposition 3.3

Proof. The number of leaves of a perfect binary tree with the depth d is always 2^d. Also, since we are considering binary splitting at any node, the number of leaves at each node is 2. Therefore, if we multiply this by 2^{d-1}, we get 2^d leaves at the same depth required for a perfect binary tree. Therefore, for the maximum depth D of a tree, by having 2^{D-1} copies of the same tree, we can generate all oblivious trees corresponding to root-to-leaf paths. □

C. Convergence of the Kernels

Figure A.1 shows the convergence of the kernels as the number M of trees increases. We set $\alpha = 4.0$ and $\beta = 0.1$. The kernels induced by finite trees $M = \{16, 64, 256, 1024, 4096\}$ are computed numerically by re-initializing the parameters 10 times. We plot two cases: $\boldsymbol{x}_i = (1, 0), \boldsymbol{x}_j = (\cos(\omega), \sin(\omega))$ with $\omega \in [0, \pi]$, and $\boldsymbol{x}_i = (\frac{1}{\sqrt{2}}, \frac{1}{\sqrt{2}}), \boldsymbol{x}_j = (\cos(\omega), \sin(\omega))$ with $\omega \in [\frac{\pi}{4}, \frac{5\pi}{4}]$. This visualization confirms that as the number of trees increases, the kernel asymptotically approaches the formula defined in Theorem 3.1.

Acknowledgement

This work was supported by JST, CREST Grant Number JPMJCR22D3, Japan, and JSPS KAKENHI Grant Number JP21H03503.