Neural Tree Kernel Learning

Suwen Lin[*]
University of Notre Dame
Notre Dame, IN 46556, USA
slin4@nd.edu

Martin Wistuba
IBM Research
Dublin, D15HN66, Ireland
martin.wistuba@ibm.com

Ambrish Rawat
IBM Research
Dublin, D15HN66, Ireland
ambrish.rawat@ie.ibm.com

Nitesh Chawla
University of Notre Dame
Notre Dame, IN 46556, USA
nchawla@nd.edu

Abstract

We propose an interpretable and scalable kernel, *Neural Tree Kernel*, which combines neural networks with the principles of decision trees for developing kernel methods which are both, highly flexible and interpretable. Specifically, the *Neural Tree Kernel* transforms the inputs of base kernels, such as RBF kernels, through the designed neural trees. The proposed kernel operates as an independent unit and can be easily applied as a drop-in replacement for any base kernel in a kernel-based model, such as a Gaussian Process (GP). To validate the effectiveness of our *Neural Tree Kernel*, we conduct classification experiments on 20 datasets from the UCI repository and show the promising performance of our model. We also demonstrate the interpretability of our kernel in a classification task and its practical usefulness in automating machine learning for natural language tasks (Appendix A).

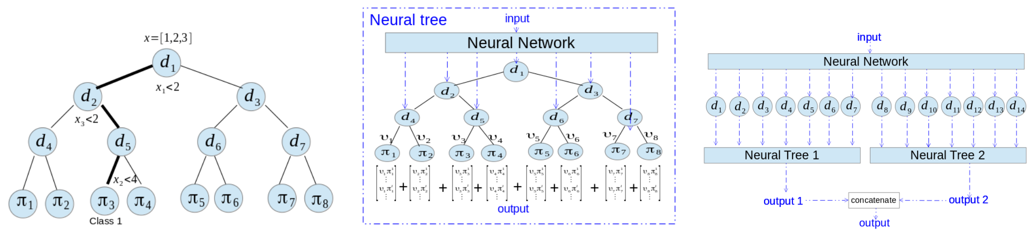
1 Introduction

With the advent of artificial intelligence, significant efforts have been devoted to research on neural network architectures and applications, such as Deep Neural Networks (Bengio et al., 2009) and Convolutional Neural Networks (CNN) (LeCun et al., 2015). Many works have further demonstrated the ability of neural networks to capture meaningful representations from data (Cao et al., 2017; Ballinger et al., 2018). On the other hand, Gaussian Processes (GPs) have attracted some attention, as they provide promising performance and quantify uncertainties in various tasks while being non-parametric (Rasmussen and Williams, 2005). Moreover, recent work has claimed that expressive kernel functions, when used in combination with a GP, are highly capable of discovering rich data structures (Wilson, 2014; Yang et al., 2015). Then, the question becomes whether we can combine and take advantage of these two methods.

To this end, recent research has shed light on the work of the integration of GP kernel development and neural network architectures. Wilson et al. (2016) developed deep kernels based on multi-layer neural networks and showed the decent performance on various tasks. However, it also carries its own limitations on the interpretability, as is often the case for neural networks and deep learning methods. In this paper, we further this investigation and resolve these concerns by introducing tree structures into the neural networks used in the kernel functions. Many have worked for the neural tree constructions, such as deep neural forest (Kontschieder et al., 2015) and neural trees (Yang et al., 2018). To the best of our knowledge, the exploration of tree structures to model kernel functions is yet unexplored.

[*]Work done while an intern at IBM Research.

Preprint. Under review.



(a) Example of a decision tree (b) Structure of *Neural Tree Kernel* (c) Structure of *neural forest*

Figure 1: The *Neural Tree Kernel* Framework.

For developing *Neural Tree Kernel* we first adopt and modify the neural tree structure from the work of Kontschieder et al. (2015). Then, we apply it as a transformation to the base kernel inputs, *e.g.* for an RBF kernel. We jointly learn the parameters of the neural tree structures and the base kernels via maximizing the log marginal likelihood of the GP.

2 The *Neural Tree Kernel* Framework

Designing suitable kernels for different tasks and datasets is a challenging task and a large amount of research has been devoted to develop appropriate kernel functions for various applications (Micchelli et al., 2006; Snoek et al., 2012; Yogatama and Mann, 2014). However, these conventional approaches fall short on model interpretability. Thus, we propose the *Neural Tree Kernel* (NTK), which employs a tree structure in its definition and consequently enables a natural way towards interpretability. Additionally, it also utilises the expressive power of neural network architectures to enable learning for high-dimensional unstructured data.

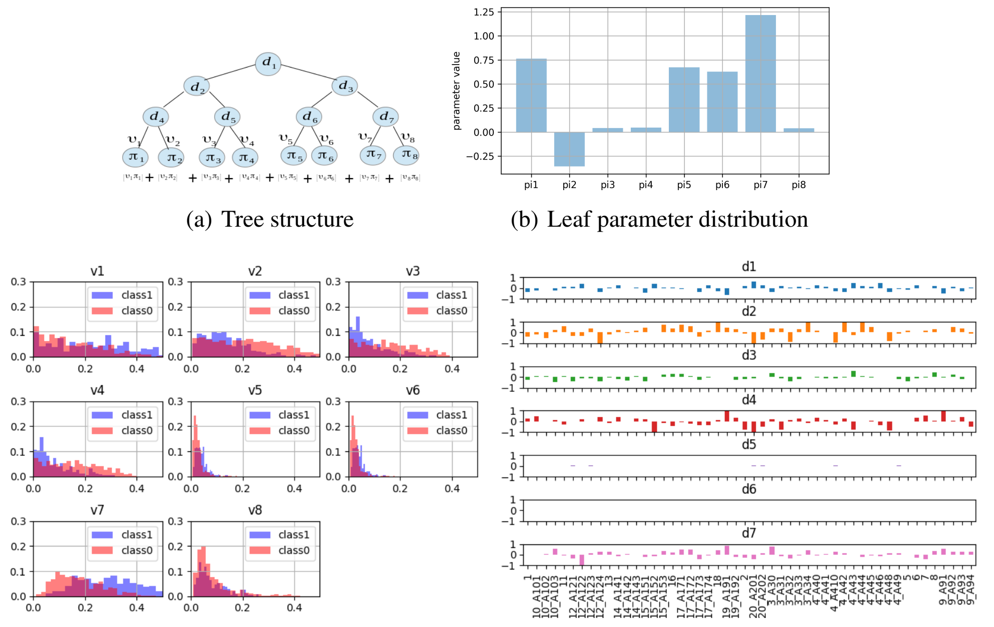
One of the appealing advantages of a decision tree is that one can easily trace paths in the tree structures to explain the relationship between input and output variables. We leverage this in our work and build upon the neural tree model (Kontschieder et al., 2015) to enable kernel learning. Specifically, starting from a base kernel $\kappa_{\boldsymbol{\theta}}(\boldsymbol{x}_i, \boldsymbol{x}_j)$ with hyperparameters $\boldsymbol{\theta}$, we extend the kernel by incorporating a neural tree to model the transformation mapping g of the inputs, $\kappa_{\boldsymbol{\theta}}(g_{\boldsymbol{\Psi}}(\boldsymbol{x}_i), g_{\boldsymbol{\Psi}}(\boldsymbol{x}_j))$. In order to enable joint learning for the parameters $\boldsymbol{\Psi}$ and $\boldsymbol{\theta}$, we design the transformation function $g_{\boldsymbol{\Psi}}(\boldsymbol{x})$ as a differentiable version of decision tree, namely a neural decision tree. The basic structure of our neural tree is shown in Figure 1(b). It shares many commonalities with a decision tree. It consists of a set of internal nodes $\mathcal{Q}$ and a set of leaf nodes $\mathcal{L}$. For an input $x \in X$, the output is obtained by traversing the paths from the root node to the leaf nodes. Each internal node $q \in \mathcal{Q}$ is associated with a decision function $d_q(\cdot)$, which defines the routing through the tree. Each leaf node $l \in \mathcal{L}$ is assigned a value, $\boldsymbol{\pi}_l$, that is used to estimate the model output. It differs from decision tree in two aspects. A standard decision tree has a deterministic routing and thus the output for an input is determined by only one leaf node (Figure 1(a)). However, the routing of a neural tree is stochastic. The output of the decision function of a node q,

$$d_{q,\boldsymbol{w}}(\boldsymbol{x}) = \varphi(f_q(\boldsymbol{x}; \boldsymbol{w}))\,, \tag{1}$$

where φ is the logistic function and f_q is a neural network with parameters $\boldsymbol{w}$. This defines the probability of selecting the left and right subtree, correspondingly. As a result, the output of the tree is not determined by just one leaf, but by the sum of all paths weighted by the path probability $\upsilon_{l,\boldsymbol{w}}$,

$$t_{\Psi}(\boldsymbol{x}) = \sum_{l \in \mathcal{L}} \upsilon_{l,\boldsymbol{w}}(\boldsymbol{x}) \boldsymbol{\pi}_l\,. \tag{2}$$

A neural forest is a combination of T neural trees where each tree is trained on a random subset of the data with feature drop out rate $\gamma \in (0, 1]$. We use a neural forest to model our *Neural Tree Kernel*. The output of the kernel is defined as the concatenation of the tree outputs (Figure 1(c)). We apply this kernel for GPs and learn all parameters jointly by maximizing the marginal likelihood. To address the scalability issue of GPs, we use the KISS-GP (Wilson and Nickisch, 2015).



(a) Tree structure

(b) Leaf parameter distribution

(c) Path probability distribution.

(d) Model weights of internal nodes.

Figure 2: Interpreting the model.

3 Experiments

Classification We evaluate the proposed *Neural Tree Kernel* on 20 classification tasks randomly chosen from the UCI repository (Dua and Graff, 2017) and compare it with two baselines: the RBF kernel Vert et al. (2004) and the deep kernel Wilson et al. (2016). Our proposed kernel provides the best result on 15 of these datasets, while DLK outperforms in 7 cases and the RBF kernel in none. Detailed results are provided in Table 1.

Interpretability We use the German Credit Risk dataset (Dua and Graff, 2017) to demonstrate model interpretability. The task is to evaluate the credit risk level of customers according to a set of attributes, involving 20 categorical and integer features. The tree structure is shown in Figure 2(a) and the values of the leaf parameters in Figure 2(b). Similarly, we visualize the distribution of path probabilities of the data in Figure 2(c) and the model weights of the decision functions in Figure 2(d). We note the following main observations for the obtained results. The leaf parameters π_3, π_4 and π_8 have small values in Figure 2(b) and the path probabilities υ_5 and υ_6 are also close to 0 in Figure 2(c). Thus, they have only little impact on the prediction and the leaf nodes one, two and seven are the most important. Figure 2(c) underlines this statement. The distribution of path probabilities differ significantly between the two classes for paths related to these three leaves. Now, we have a look at all features with high absolute values in the decision function which are d_2, d_4 and d_8 (Figure 2(d)). Those are the most important predictors for our model. These features describe attributes of the customer such as the ownership of house and existence of critical credits and are exactly those features we were assuming to be the most important. Finally, we notice that the network parameters for nodes d_5 and d_6 are much smaller than others. Thus, the corresponding node probabilities for these two are close to 0.5 and could be merged.

We also explored the efficacy of our model on a large-scale experiment of fine-tuning models for language tasks. However, for the sake of brevity we discuss this in Appendix A.

4 Conclusion

In this work, we propose the novel *Neural Tree Kernel* that is effective, scalable and interpretable on a number of inference tasks. In the proposed kernel, we introduce a neural network architecture

Table 1: Classification task performance *w.r.t.* accuracy and statistics on UCI datasets.

Data	RBF	DKL	*Neural Tree Kernel*	#categorical	#numeric	#samples	#class
balance-scale	0.8571	0.9683	**0.9762**	0	4	625	3
car	0.8642	0.9971	**1.0000**	6	0	1728	4
cleveland	0.5333	**0.5667**	**0.5667**	7	6	303	5
contraceptive	0.3986	**0.5203**	0.5068	4	5	1473	3
dermatology	0.8784	0.9865	**1.0000**	33	1	366	6
dna	0.8953	**0.9501**	0.9277	180	0	3186	3
iris	0.8667	0.9333	**0.9667**	0	4	150	3
new-thyroid	0.8372	0.9535	**0.9767**	0	5	215	3
nursey	0.9094	**1.0000**	**1.0000**	8	0	12958	4
page-blocks	0.9334	0.9626	**0.9644**	0	10	5473	5
pendigits	0.8628	0.9255	**0.9882**	0	16	10992	10
satimage	0.7885	0.8425	**0.8526**	0	36	6430	6
segment	0.8463	**0.9589**	0.9567	0	19	2310	7
splice	0.8868	0.9119	**0.9355**	60	0	3190	3
tae	0.3548	0.4194	**0.4839**	2	3	151	3
tic-tac-toe	0.8186	**0.9896**	0.9845	9	0	958	2
texture	0.8836	0.9155	**0.9945**	0	40	5500	11
vehicle	0.5439	0.7135	**0.7427**	0	18	846	4
waveform	0.7632	**0.8452**	0.8242	0	40	5000	3
wine-quality	0.3994	0.5170	**0.5201**	0	11	1599	6
Wins/Total	0/20	7/20	15/20	–	–	–	–

to construct tree structures the output of which is fed into a base kernel. Then we jointly learn the parameters of the tree and base kernel. Extensive experiments on a number of datasets demonstrate the effectiveness and efficacy of our kernel. Through a classification task, we demonstrate how the model can be interpreted and decisions can be explained. Additionally, we demonstrate the interpretability of our model stemming from the inherent tree structure and show the usefulness of our model in a large-scale experiment on hyperparameter optimisation.

References

Alberto, T. C., Lochter, J. V., and Almeida, T. A. (2015). Tubespam: Comment spam filtering on youtube. In *2015 IEEE 14th International Conference on Machine Learning and Applications (ICMLA)*, pages 138–143. IEEE.

Ballinger, B., Hsieh, J., Singh, A., Sohoni, N., Wang, J., Tison, G. H., Marcus, G. M., Sanchez, J. M., Maguire, C., Olgin, J. E., et al. (2018). Deepheart: semi-supervised sequence learning for cardiovascular risk prediction. In *Thirty-Second AAAI Conference on Artificial Intelligence*.

Bengio, Y. et al. (2009). Learning deep architectures for ai. *Foundations and trends® in Machine Learning*, 2(1):1–127.

Cao, B., Zheng, L., Zhang, C., Yu, P. S., Piscitello, A., Zulueta, J., Ajilore, O., Ryan, K., and Leow, A. D. (2017). Deepmood: modeling mobile phone typing dynamics for mood detection. In *Proceedings of the 23rd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 747–755. ACM.

Devlin, J., Chang, M.-W., Lee, K., and Toutanova, K. (2018). Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*.

Dua, D. and Graff, C. (2017). UCI machine learning repository.

Howard, J. and Ruder, S. (2018). Universal language model fine-tuning for text classification. *arXiv preprint arXiv:1801.06146*.

Kontschieder, P., Fiterau, M., Criminisi, A., and Rota Bulo, S. (2015). Deep neural decision forests. In *Proceedings of the IEEE international conference on computer vision*, pages 1467–1475.

LeCun, Y., Bengio, Y., and Hinton, G. (2015). Deep learning. *nature*, 521(7553):436.

Maas, A. L., Daly, R. E., Pham, P. T., Huang, D., Ng, A. Y., and Potts, C. (2011). Learning word vectors for sentiment analysis. In *Proceedings of the 49th annual meeting of the association for computational linguistics: Human language technologies-volume 1*, pages 142–150. Association for Computational Linguistics.

Micchelli, C. A., Xu, Y., and Zhang, H. (2006). Universal kernels. *Journal of Machine Learning Research*, 7(Dec):2651–2667.

Rasmussen, C. E. and Williams, C. K. I. (2005). *Gaussian Processes for Machine Learning (Adaptive Computation and Machine Learning)*. The MIT Press.

Snoek, J., Larochelle, H., and Adams, R. P. (2012). Practical bayesian optimization of machine learning algorithms. In *Advances in neural information processing systems*, pages 2951–2959.

Vert, J.-P., Tsuda, K., and Schölkopf, B. (2004). A primer on kernel methods. *Kernel methods in computational biology*, 47:35–70.

Warstadt, A., Singh, A., and Bowman, S. R. (2018). Neural network acceptability judgments. *arXiv preprint arXiv:1805.12471*.

Wilson, A. and Nickisch, H. (2015). Kernel interpolation for scalable structured gaussian processes (kiss-gp). In *International Conference on Machine Learning*, pages 1775–1784.

Wilson, A. G. (2014). *Covariance kernels for fast automatic pattern discovery and extrapolation with Gaussian processes*. PhD thesis, University of Cambridge.

Wilson, A. G., Hu, Z., Salakhutdinov, R., and Xing, E. P. (2016). Deep kernel learning. In *Artificial Intelligence and Statistics*, pages 370–378.

Wistuba, M., Schilling, N., and Schmidt-Thieme, L. (2018). Scalable gaussian process-based transfer surrogates for hyperparameter optimization. *Machine Learning*, 107(1):43–78.

Yang, Y., Morillo, I. G., and Hospedales, T. M. (2018). Deep neural decision trees. *arXiv preprint arXiv:1806.06988*.

Yang, Z., Wilson, A., Smola, A., and Song, L. (2015). A la carte–learning fast kernels. In *Artificial Intelligence and Statistics*, pages 1098–1106.

Yogatama, D. and Mann, G. (2014). Efficient transfer learning method for automatic hyperparameter tuning. In *Artificial intelligence and statistics*, pages 1077–1085.

A Case Study: AutoML

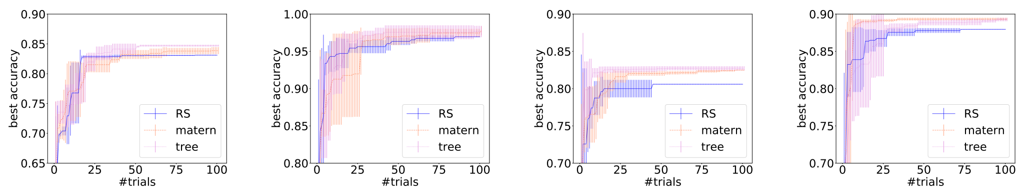
In this experiment we address the following research question: How does our Neural Tree Kernel perform compared to contemporary baselines in the application of automated machine learning (AutoML)? For this, we use a Gaussian Process with the proposed *Neural Tree Kernel* in a Bayesian optimization setting and optimize the hyperparameters of machine learning pipelines to fine-tune BERT-based classifiers (Devlin et al., 2018) on four text datasets with varied statistics (Table 2). BERT is a very popular language model which is pre-trained on a large text corpus so as to yield high performance on transfer learning tasks. However this requires high computational effort during training (Devlin et al., 2018). á

Table 2: Detailed dataset statistics for the AutoML experiments.

Data	#class	#sample	reference
CoLA	2	9077	Warstadt et al. (2018)
YouTube	2	1954	Alberto et al. (2015)
RDBN	4	8844	project data
IMDB	2	50000	Maas et al. (2011)

Experiment description Bayesian optimization has become of the most popular means to optimize hyperparameters automatically Snoek et al. (2012). This framework consists of two components, a surrogate model, which predicts a hyperparameter configuration's performance, and an acquisition function, which evaluates the expected utility of a hyperparameter configuration based on the surrogate model's predicted distribution. Sequentially, the hyperparameter configuration that maximizes the expected utility is selected and the model is trained with these settings. The resulting response is employed to update the surrogate model. GPs are the most common choice for the surrogate model and they have successfully demonstrated their usefulness in different scenarios (Snoek et al., 2012; Wistuba et al., 2018). For the acquisition function we choose expected improvement.

In the experiments, we set 80% data for training, 10% for validation and 10% data for testing. We further adopt Random Search (RS) and GP with Matérn kernel (matern) as baselines. The considered hyperparameters are learning rate schedule, additional layers, layer freezing, batch size, number of epochs and regularization parameters such as weight decay and dropout. We elaborate now on the specific search space in this experiment. Three different learning rate schedulers are considered: linear schedule, cosine schedule and constant schedule, each of them having one or more additional hyperparameters. The learning rate ranges from 10^{-8} to 10^{-3}, the warmup proportion from 0 to 0.5, the ratio for gradual layer learning rate from 1 to 5 (Howard and Ruder, 2018). Further choices are the addition of no to up to three additional dense layers, the batch size (16,24,32), number of epochs (1-5), dropout rate (0-0.8) and the weight decay ranges from 10^{-8} to 10^{-2} Finally, it is decided whether the layers are incrementally unfrozen during training (Howard and Ruder, 2018). Note that we use the same starting points for both the runs with Matérn and *Neural Tree Kernel* and fix the maximal number of considered hyperparameters to 100. The experiment is repeated five times, the average test performance across these repetitions is shown in Figure 3.



(a) CoLA (b) YouTube (c) RDBN (d) IMDB

Figure 3: Optimization progress for Random Search (RS), *Matérn* kernel and *Neural Tree Kernel*.

Results From Figure 3, we have the following main observations:

- The *Neural Tree Kernel* achieves the best performance within the 100 trials compared to the baselines on CoLA, RDBN and YouTube datasets while its performance on IMDB is still comparable. It demonstrates the advantage of our *Neural Tree Kernel* also for applications in AutoML.
- Moreover, it is notable that Bayesian optimization with our kernel converges faster to well-performing hyperparameters than the baselines for three datasets. Therefore, fewer computational-expensive BERT-based classifiers are required to train.

In conclusion, the above observations suggest that the *Neural Tree Kernel* is a strong addition to current state-of-the-art hyperparameter optimization framework, yielding a promising performance.