# From Soft Trees to Hard Trees: Gains and Losses

**Xin Zeng** [1] **Jiayu Yao** [1] **Finale Doshi-Velez** [1] **Weiwei Pan** [1]

## Abstract

Trees are widely used as interpretable models. However, when they are greedily trained they can yield suboptimal predictive performance. Training soft trees, with probabilistic splits rather than deterministic ones, provides a way to supposedly globally optimize tree models. For interpretability purposes, a hard tree can be obtained from a soft tree by binarizing the probabilistic splits, called *hardening*. Unfortunately, the good performance of the soft model is often lost after hardening. We systematically study two factors contributing to the performance drop: first, the loss surface of the soft tree loss has many local optima (and thus the logic for using the soft tree loss becomes less clear), and second, the relative values of the soft tree loss do not correspond to relative values of the hard tree loss. We also demonstrate that simple mitigation methods in literature do not fully mitigate the performance drop.

## 1. Introduction

Interpretability is an important property of models that are deployed for high stakes decision-making tasks (Rudin, 2018; Brundage et al., 2020). In practice, trees are often considered to be inherently interpretable models and they are widely studied and applied for interpretability purposes (e.g. Das & Rad, 2020). The reason is that, in trees, each node represents a specific condition and the subsequent assignments of inputs to decision regions depend deterministically on satisfying these conditions. Thus, for each input, we can output one decision rule explaining the tree's prediction.

Although trees are human-interpretable, their optimization is challenging. First, trees are often trained greedily (e.g. Mehta et al., 1996), optimizing the split for one node at a time, which generally results in suboptimal models (Rudin et al., 2021). Second, trees cannot be easily incorporated into end-to-end training pipelines with other models since traditional tree training is not differentiable.

To address these issues, a body of works proposes to first train a soft tree—that is, a tree in which each split is probabilistic, and thus each input is assigned to each decision region with certain probability, and the model prediction is a weighted sum of the prediction of each region—and then *harden* the soft tree into a hard one. The idea is that the loss for the soft tree serves as a surrogate for the hard tree training. The soft tree loss has desirable properties such as differentiability and thus training can be jointly done with other models in a task pipeline (e.g. Frosst & Hinton, 2017; Tanno et al., 2018).

In practice, the hardening process works well for trees in classification settings due to the discretization nature of classification tasks (Frosst & Hinton, 2017; Coppens et al., 2019; Rajaguru & Prabhakar, 2017). Unfortunately, for regression tasks, these promises have not been realized: there is often a performance gap when obtaining hard trees via soft tree training. In this work, we systematically study two types of soft trees. We summarize two key factors contributing to the performance gap of trees on regression tasks. (1) Soft trees training is highly non-convex (with many local optima); thus, the training process is very sensitive to initialization and learning rate; thus, moving from optimizing a non-differentiable loss function to a continuous but highly non-convex may provide limited practical benefit. (2) The hardening process does not preserve the relative orderings of the loss: a soft tree with a low loss might harden to a tree with high loss, whereas a soft tree with slightly worse performance might harden to a much better hard tree. Because of the above factors, when designing soft tree losses as surrogates, we need to carefully investigate their smoothness and their consistency with the hard tree loss.

## 2. Related Work on Designing Soft Tree Losses

There is a large body of works that use soft tree loss as a surrogate for training a hard tree. For example, Irsoy et al. force the soft tree to behave like a hard tree by using a hyper-parameter to control the steepness of the sigmoid function that determines the split at each node in the soft tree. Xu et al. approximate the loss function using reparameterization tricks and optimize the tree using primal iteration. As another example, Frosst & Hinton propose adding a

[1]SEAS, Harvard University. Correspondence to: Xin Zeng <xinzeng@fas.harvard.edu>.

*Proceedings of the $38^{th}$ International Conference on Machine Learning*, PMLR 139, 2021. 

penalty term to the loss function, which avoids poor local optima by encouraging a balanced tree. However, these methods do not fully mitigate the performance gap. In this work, we specifically study the hyper-parameter trick (Irsoy et al., 2012) and show that there is a trade-off between the soft tree optimization and the performance gap due to hardening. We observe a similar trade-off in other methods (Appendix 7). We argue that systematic studies of other soft tree optimization works are also needed.

## 3. Background

We formalize soft trees using Hierarchical Mixture of Experts (HMEs) (Jordan & Jacobs, 1994). We consider a regression task consisting of $N$ observations, $\mathcal{D} = \{\mathbf{x}_n, \mathbf{y}_n\}_{n=1}^N$ with $\mathbf{x}_n \in \mathbb{R}^l$, $\mathbf{y}_n \in \mathbb{R}^k$. We consider a binary HME of depth $D$ with $2^D - 1$ *gating networks* at the non-terminal nodes and $2^D$ *expert networks* at the leaf nodes. The gating networks divide the input space into a set of regions with expert networks determining the predicted values of each region. Specifically, we denote the gating network at $i$-th non-terminal node as a binary variable,

$$z_i \sim \text{Bern}(p_i),\ p_i = \frac{1}{1 + e^{-\beta \mathbf{v}_i^\top \mathbf{x}_n}},\ \text{for } i = 1, \cdots, 2^D - 1 \tag{1}$$

where $\mathbf{v}_i$ defines the boundaries of divided regions and $\beta$ controls how soft the boundaries are (how fast one region transits into another). Denote the expert network at $j$-th leaf node as $\mathbf{t}_j$. The conditional distribution of the expert $\mathbf{t}_j$ given a input $\mathbf{x}$ is given by

$$p(\mathbf{t}|\mathbf{x}_n, \tau_j) = \mathcal{N}(\mathbf{t}|h_j(\mathbf{x}_n), \tau_j^{-1}\mathbb{I}),\ \text{for } j = 1, \cdots, 2^D$$

where $\tau_j$ is the precision (inverse variance) of the distribution, $\mathbb{I}$ is an identity matrix and $h_j$ is link functions that define the relationship between the inputs and the leaf nodes.

Denote the unique path with length $D$ to $j$-th leaf node as a vector $\boldsymbol{\xi}_j = \{z_{i_1}, \cdots, z_{i_D}\}$ where $i_d \in \{1, \cdots, 2^D - 1\}$. HMEs work by first assigning an input $\mathbf{x}$ to each leaf node $j$ with probability

$$p(\boldsymbol{\xi}_j|\mathbf{x}_n, \mathbf{v}) = \prod_{d=1}^{D} p(z_{i_d}|\mathbf{x}_n, \mathbf{v}),$$

then giving an output by using a weighted sum of the outputs of expert networks,

$$p(\mathbf{y}_n|\mathbf{x}_n, \mathbf{v}, \boldsymbol{\tau}) = \sum_{j}^{2^D} p(\mathbf{t}|h_j(\mathbf{x}_n), \tau_j) p(\boldsymbol{\xi}_j|\mathbf{x}_n, \mathbf{v}). \tag{2}$$

We are interested in the performance of hard trees. To obtain a hard tree, instead of marginalizing over leaf nodes, we assign the input to leaf nodes following the path with the greatest probability,

$$p_{\text{hard}}(\mathbf{y}_n|\mathbf{x}_n, \mathbf{v}, \boldsymbol{\tau}) = p(\mathbf{t}|h_{j^*}(\mathbf{x}_n), \tau_{j^*})$$

where $j^* = \arg\max_j p(\boldsymbol{\xi}_j|\mathbf{x}_n, \mathbf{v})$. We define the above process as *hardening*. By hardening, we hope to gain an interpretable model while retaining the predictive performance of the soft tree.

In this work, we investigate expert networks defined by two different link functions: (1) constant experts $h_j(\mathbf{x}_n) = \mathbf{c}$ where $\mathbf{c} \in \mathbb{R}^k$, (2) linear experts with $h_j(\mathbf{x}_n) = \mathbf{W}_j \mathbf{x}_n$ where $\mathbf{W}_j \in \mathbb{R}^{k \times l}$.

## 4. Experimental Setup

We define the difference in terms of the predictive performance between the soft and the corresponding hardened model as the *performance gap due to hardening*. We investigate the performance gap with two types of soft trees: HMEs with constant experts for easy analyses, and HMEs with linear experts for more complicated regression tasks.

For all experiments, the hyperparameter $\beta$ (Equation 1) is set to 1 unless otherwise stated. For each type of HMEs, we use different algorithms for inference. For HMEs with constant experts, we estimate parameters $\mathbf{c}, \mathbf{v}, \boldsymbol{\tau}$ by maximizing the likelihood,

$$\mathcal{L} = \sum_{n=1}^{N} p(\mathbf{y}_n|\mathbf{x}_n, \mathbf{v}, \boldsymbol{\tau})$$

where $p(\mathbf{y}_n|\mathbf{x}_n, \mathbf{v}, \boldsymbol{\tau})$ is given in Equation 2. We perform optimization with stochastic gradient descent (SGD), which helps avoid poor local optima.

For HMEs with linear constants, we perform Variational Inference (VI), which avoids severe overfitting of maximum likelihood estimates. (Bishop & Svensén, 2012).

**Datasets** We designed two toy datasets: (1) a step function with 4 pieces, which matches inductive biases of HMEs with constant experts; (2) a cubic function $y = 3x^3$ with a sparse data region, which is a common benchmark for uncertainty quantification (e.g. Hernández-Lobato & Adams, 2015; Sun et al., 2019; Yao et al., 2019). For the step function, we used a 2-layer HME (the minimal needed capacity) with constant experts. For the cubic function, we tested 5, 6, 7-layer HMEs with linear experts. To obtain the uncertainty estimate, we perform bootstrap by random sampling the training data with replacement, and then fitting one HME for each data resampling.

**Evaluation Metrics** For the step function, we compare the MSEs of soft and hard trees. For the cubic funcion, we investigate the likelihood. Denote the HME trained from $s$-th data resample $\mathcal{D}^{(s)}$ as $\texttt{HME}^{(s)}$. Given new data points

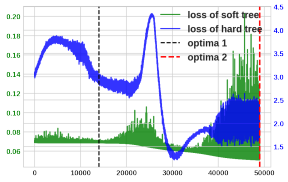
$\{\mathbf{x}_m^*, \mathbf{y}_m^*\}_{m=1}^M$, the test loglikelihood is computed as

$$\log \prod_{m=1}^{M} p(\mathbf{y}_m^*|\mathbf{x}_m^*, \mathcal{D}) = \frac{1}{S}\sum_{s=1}^{S}\sum_{m=1}^{M} \log p(\mathbf{y}_m^*|\mathbf{x}_m^*, \mathcal{D}^{(s)}, \texttt{HME}^{(s)})$$

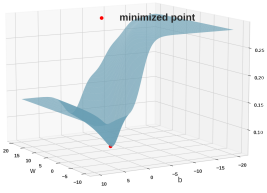
## 5. Experiments

In this section, we analyze factors contributing to the performance gap due to hardening and the resulting effects. We also study the efficacy of a method designed to reduce the performance gap using a hyper-parameter to control the steepness of the soft tree.

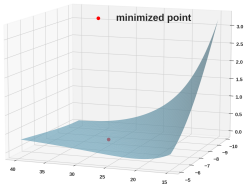
### 5.1. Ill-behaved Loss Landscape




(a) Loss during Training




(b) Loss near Local Optima 1




(c) Loss near Local Optima 2

*Figure 1.* The landscape of the loss function of the step function: (a) trace of the loss function of $50k$ training iterations. The green and blue curve represent the loss of the soft and the hard tree respectively. The black and the red line represents two different local optima of the soft tree encountered during training. The trend of the soft tree loss is not consistent to that of the hard tree loss. (b)(c) The surface plots of the loss function near the two local optima (black line and red line in (a), respectively) with the red dots representing the minimum points. We see that the loss surface exhibits both high curvature and large plateau.

**The loss function of soft trees has local optima with high curvature and large plateaus, which are hard to escape.** We investigate the loss function landscape of HMEs with constant experts. Figure 1a shows the trace of the soft tree loss of the step function during training. We see that SGD reaches the first local optima around $15k$ iteration. Figure 1b plots the loss landscape near the first optima, which is deep, and thus hard to escape. SGD exhibits oscillation ($20k - 30k$ iteration) along the high curvature direction as it tries to get away from the local optimum with large gradients. After SGD escapes the first local optimum, it reaches a better local optimum (the loss stops improving at the end). Figure 1c shows that the second local optimum has a large plateau, which may take very long for SGD to escape. The ill-behaved landscape is concerning given the HME is only of depth 2. When the depth of HME grows, the number of local optima may increase exponentially as the number of parameters increases, which makes optimization extremely difficult.

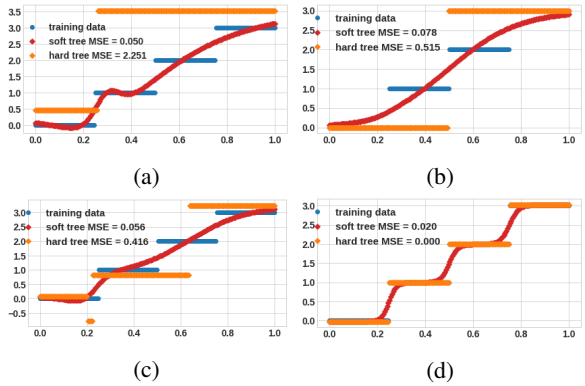
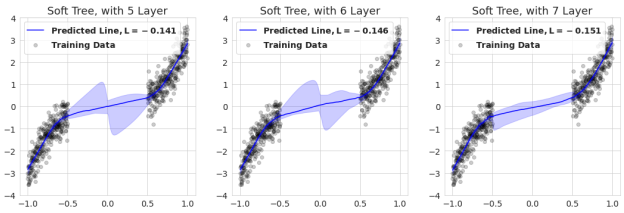



*Figure 2.* Plots of the best soft tree during training and the corresponding hard tree performance given different initialization strategy: (a) all parameters are randomly initialized (b) constant experts are fixed at ground truth and the gating networks are randomly initialized (c) constant experts are initialized near ground truth and the gating networks are randomly initialized (d) the gating networks are initialized near ground truth and the constant experts are randomly initialized. Trees achieve optima performance only when the gating networks are initialized close to the ground truth.

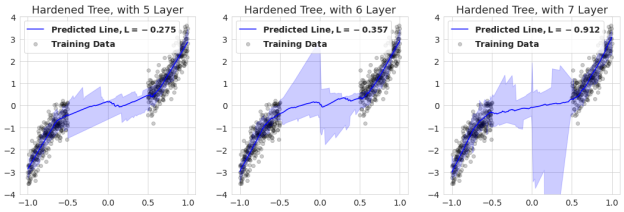
**Soft tree training is highly sensitivity to hyperparameters.** Because of the high curvature and the plateau of the loss landscape, learning is highly sensitive to the choice of initialization and learning rate, and thus the tree performance has a large variance.

Figure 2 shows the performance of HMEs with constant experts on the step function task given different initialization. We see that SGD converges to very different solutions for both soft and hard trees depending on where the parameters are initialized (More examples of different initialization in Figure 6). The soft tree can reach the optimal performance only when the gating networks are initialized near ground truth (i.e. the tree splits the input region reasonably well initially ). For flexible models like HMEs with linear experts optimized through VI, we also need multiple random restarts to converge to the global optimum. With reasonable initialization and learning rate, VI can find the best

solution quarter of the time (Figure 5). In practice, with more complicated data structure, the hyperparameter tuning gets trickier—again making soft tree optimization not necessarily an easier alternative to hard tree optimization.




(a) Performance of Soft Tree Ensembles




(b) Performance of Corresponding Hard Tree Ensembles

*Figure 3.* Plots of posterior predictive distribution with different depths on the cubic function task (a) the soft tree ensembles (b) the corresponding hard tree ensembles. For soft tree ensembles, the predictive variance of the data scarce region decreases as the depth increases while that increases for hard tree ensembles.

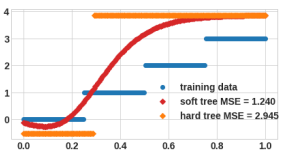
### 5.2. Inconsistency with the Hard Tree Loss

**Soft tree losses do not preserve the relative ordering of hard tree losses. A better soft tree does not necessarily harden to a better hard tree.** In Figure 1a, we see that during training, the trend of the hard tree loss (blue curve) is not consistent with the one of the soft tree (green curve). During training, the loss of the hard tree increases around the $20k$-th iteration and then decreases drastically around the $30k$-th iteration. Contrarily, the loss of the soft tree decreases in general. Comparing Figure 2a to 2b, we see that a better soft tree can result in a much worse hard tree.

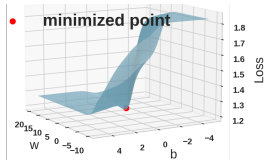
**Similar inconsistency can be observed in terms of log-likelihood.** Uncertainty quantification for discrete functions is challeging. Thus, we use the cubic function for the uncertainty estimate task. Figure 3 shows the posterior predictive distribution of tree ensembles with increasing tree depth. We see that soft trees with higher log-likelihood harden to trees with lower log-likelihood (In the data scarce region, soft trees with lower predictive posterior variance harden to trees with much higher predictive variance).

### 5.3. The Trade-off Between Soft Optimization and Performance Gap Due to Hardening

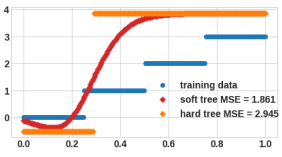
A commonly method for reducing the performance gap due to hardening is to encourage the soft tree to behave more like a hard one during training. However, we show that there is a trade-off between the ease of soft tree optimization and performance gap. We study this trade-off by varying the hyperparameter $\beta$ (Equation 1). A larger $\beta$ represents a steeper soft tree and thus a closer match between the soft and hard tree losses—but also a loss landscape that is as hard to optimize as the hard tree loss.



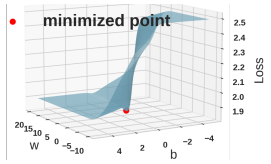

(a) Trees with $\beta = 3$




(b) Loss surface with $\beta = 3$




(c) Trees with $\beta = 5$




(d) Loss surface with $\beta = 5$

*Figure 4.* (a)(c) Plots of soft and hard tree performance with $\beta = 3, 5$. When $\beta$ increases, the soft and the hard tree become more similar. (b)(d) Plots of loss surface near the local optimum with $\beta = 3, 5$. When $\beta$ increases, the loss landscape becomes more pathological with both high curvature and large plateau.

**When $\beta$ increases, the performance gap due to hardening decreases. However, the loss function becomes harder to optimize.** We tested different $\beta$ on the step function task. Figure 4a, 4c show that when $\beta$ increases, the soft tree becomes steeper and performs more similarly to the hard tree (the difference between MSEs get smaller). Although the performance gap decreases, the loss landscape becomes more challenging. From Figure 4b, 4d, we see that the local optima exists both high curvature and large plateau (unlike 1b, 1c, there is only one pathology in each). The difficulty of managing this trade-off calls into question the practical utility of obtaining hard trees by training proxy soft trees.

## 6. Conclusion

This paper systematically studies factors contributing to the performance gap between soft trees and their hardened counterparts. We also show that simple methods for closing the performance gap do not necessarily yield hard trees with better predictive performance - as they trade-off between difficult soft optimization and performance gap due

to hardening. Although existing works aim to obtain predictive and interpretable models by globally optimizing soft trees and then hardening the solution, we show that this way of training hard trees does not get around fundamental issues on how fundamentally difficult it is to train a hard tree (Appendix Figure 7). We encourage careful investigation of the soft tree loss landscapes of relevant works and use of advanced optimization methods such as learning rate annealing to avoid poor local optima.

## Acknowledgements

JY and FDV acknowledge the support from NSF, IIS-2007076.

## References

Bishop, C. M. and Svensén, M. Bayesian hierarchical mixtures of experts. *CoRR*, abs/1212.2447, 2012. URL http://arxiv.org/abs/1212.2447.

Brundage, M., Avin, S., Wang, J., Belfield, H., Krueger, G., Hadfield, G., Khlaaf, H., Yang, J., Toner, H., Fong, R., et al. Toward trustworthy ai development: mechanisms for supporting verifiable claims. *arXiv preprint arXiv:2004.07213*, 2020.

Coppens, Y., Efthymiadis, K., Lenaerts, T., Nowé, A., Miller, T., Weber, R., and Magazzeni, D. Distilling deep reinforcement learning policies in soft decision trees. In *Proceedings of the IJCAI 2019 workshop on explainable artificial intelligence*, pp. 1–6, 2019.

Das, A. and Rad, P. Opportunities and challenges in explainable artificial intelligence (xai): A survey, 2020. URL https://arxiv.org/abs/2006.11371.

Frosst, N. and Hinton, G. Distilling a neural network into a soft decision tree, 2017.

Hernández-Lobato, J. M. and Adams, R. Probabilistic backpropagation for scalable learning of bayesian neural networks. In *International conference on machine learning*, pp. 1861–1869. PMLR, 2015.

Irsoy, O., Yıldız, O. T., and Alpaydın, E. Soft decision trees. In *Proceedings of the 21st international conference on pattern recognition (ICPR2012)*, pp. 1819–1822. IEEE, 2012.

Jordan, M. I. and Jacobs, R. A. Hierarchical mixtures of experts and the em algorithm. *Neural computation*, 6(2): 181–214, 1994.

Mehta, M., Agrawal, R., and Rissanen, J. Sliq: A fast scalable classifier for data mining. In Apers, P., Bouzeghoub, M., and Gardarin, G. (eds.), *Advances in Database Technology — EDBT '96*, pp. 18–32, Berlin, Heidelberg, 1996. Springer Berlin Heidelberg. ISBN 978-3-540-49943-5.

Rajaguru, H. and Prabhakar, S. K. Sparse pca and soft decision tree classifiers for epilepsy classification from eeg signals. In *2017 International Conference of Electronics, Communication and Aerospace Technology (ICECA)*, volume 1, pp. 581–584. IEEE, 2017.

Rudin, C. Stop explaining black box machine learning models for high stakes decisions and use interpretable models instead. 2018. doi: 10.48550/ARXIV.1811.10154. URL https://arxiv.org/abs/1811.10154.

Rudin, C., Chen, C., Chen, Z., Huang, H., Semenova, L., and Zhong, C. Interpretable machine learning: Fundamental principles and 10 grand challenges. 2021. doi: 10.48550/ARXIV.2103.11251. URL https://arxiv.org/abs/2103.11251.

Sun, S., Zhang, G., Shi, J., and Grosse, R. Functional variational bayesian neural networks. *arXiv preprint arXiv:1903.05779*, 2019.

Tanno, R., Arulkumaran, K., Alexander, D. C., Criminisi, A., and Nori, A. V. Adaptive neural trees. *CoRR*, abs/1807.06699, 2018. URL http://arxiv.org/abs/1807.06699.

Xu, Z., Zhu, G., Yuan, C., and Huang, Y. One-stage tree: end-to-end tree builder and pruner. *Machine Learning*, pp. 1–27, 2021.

Yao, J., Pan, W., Ghosh, S., and Doshi-Velez, F. Quality of uncertainty quantification for bayesian neural network inference. *arXiv preprint arXiv:1906.09686*, 2019.

# A. Datasets & Hyperparameter

### Step Function

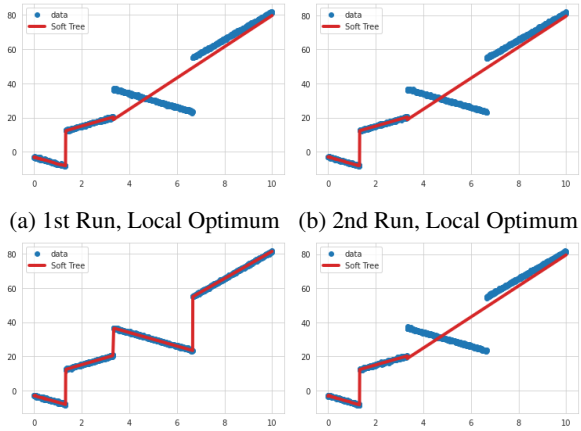
$$f(x) = \begin{cases} 0 & 0 < x \leq 0.25 \\ 1 & 0.25 < x \leq 0.5 \\ 2 & 0.5 < x \leq 0.75 \\ 3 & 0.75 < x \leq 1 \end{cases}$$

We used a 2-layer HME with constant experts. Parameters are randomly initialized with $\texttt{Unif}(0, 1)$ and solved through stochastic gradient descent. We perform optimization with learning rate of 0.001, batch size of 32, and number of epoch of 50000.

**Cubic Function** The training data is generated with $y = 3x^3 + \mathcal{N}(0, 0.2)$ with the input $x$ uniformly sampled from $[-1, -0.5] \bigcup [0.5, 1.0]$. We we tested 5, 6, 7-layer HMEs with linear experts. Parameters are randomly initialized with $\texttt{Unif}(0, 1)$ and solved via Variational Inference with learning rate of 0.001 number of epoch of 5000.

# B. Additional Experiments

### B.1. High sensitivity to hyperparameters

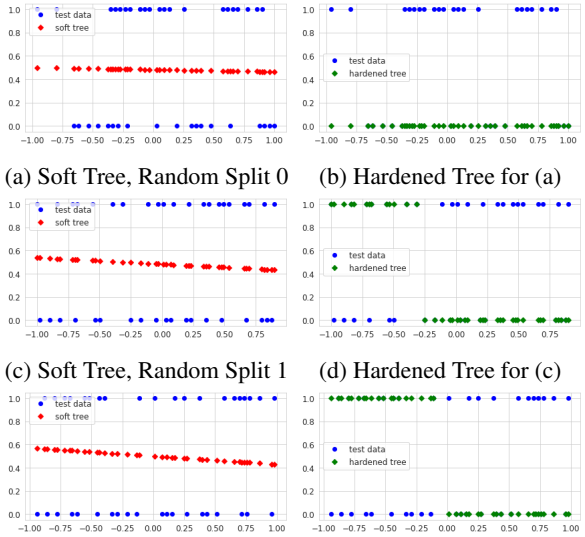



(a) 1st Run, Local Optimum (b) 2nd Run, Local Optimum

(c) 3rd Run, Global Optimum (d) 4th Run, Local Optimum

*Figure 5.* Plots of HMEs with random restart. HMEs converge to the global optimum one out of four runs with other times stuck at poor local optima.

Even for flexible models like HMEs with linear experts optimized through VI, we often need multiple random restarts to converge to an optimal solution. We tested HMEs with linear experts with a piece-wise function with linear components. Figure 5 shows that even for a simple 4-component piece-wise function, HMEs struggle to always find the global optimum during training.




(a) Soft Tree, Random Split 0 (b) Hardened Tree for (a)

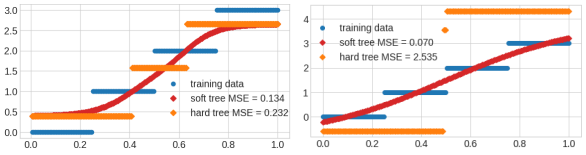
(c) Soft Tree, Random Split 1 (d) Hardened Tree for (c)

(e) Soft Tree, Random Split 2 (f) Hardened Tree for (e)

*Figure 6.* Plots of tree results with different data split. (a)(c)(e) Soft trees learnt with different training/test data split. (b)(d)(f) Corresponding hard trees. Although the soft trees look similar among different random states, the corresponding hardened trees vary a lot from each other.

Due to the ill-behaved loss landscape, trees are highly sensitive to hyperparameters such as data split, parameter initialization, learning rate etc. Figure 6 shows a regression task where the function oscillates within a small neighborhood and requires a large tree. We split the training and test data with different random states and fit HMEs with linear experts for the training data and evaluate the test data. As we can see from the figures, although the soft trees look similar among different random states, the corresponding hardened trees vary a lot from each other.

### B.2. The Trade-off Between Soft Optimization and Performance Gap of Other Related Works




(a) One Stage Tree (b) Tree with regularization

*Figure 7.* Plots of trees obtained from optimizing different loss functions: (a) One Stage Tree: although the soft and hard tree perform similarly, both trees perform badly in terms of MSEs. (b) Trees with balance regularization: the soft tree performs well with the hard tree performs much worse than the soft tree.

In addition to controlling the steepness of the soft tree, there

are other works focus on designing soft tree losses as a surrogate for training the hard tree. We tested two common approaches.(1) One Stage Tree, which approximates the loss function using reparameterization tricks and then optimizes the tree using primal iteration Xu et al.. (2) Soft tree loss with a regularization, which helps SGD avoid poor local optima by encouraging more balanced trees (Frosst & Hinton, 2017). From Figure 7, we also see the trade-off between optimization and the performance gap. Although One Stage Tree gives similar soft and hard trees, both trees perform badly due to optimization difficulty. Trees with a balanced regularization gives well-performing soft trees but much worse hard trees.