Amortised Variational Inference for Hierarchical Mixture Models

Javier Antorán [1 *] **Jiayu Yao** [2 *] **Weiwei Pan** [2] **José Miguel Hernández-Lobato** [1 3 4] **Finale Doshi-Velez** [2]

Abstract

Hierarchical Mixtures of Experts (HME) are flexible and interpretable probabilistic models. However, existing approaches to learning tree-structured decision rules are prone to poor local optima. This work introduces an end-to-end differentiable amortised variational inference algorithm for HMEs. We use an RNN (dubbed RNN-Tree) to approximate the posterior distribution over tree node routing decisions. We show that our RNN-Tree finds better decision rules than greedily learnt trees, resulting in better generalisation performance. We also show how RNN-Trees' differentiability facilitates their integration into existing machine learning pipelines.

1. Introduction and Related Work

Hierarchical Mixtures of Experts (HME), which can be seen as probabilistic decision trees, are flexible yet interpretable models. An HME segments the input space into a nested set of regions. Each region corresponds to a leaf node. The data assigned to each leaf is fit with a simple predictor (an expert). Each input is routed to leaf node based on the sequential application of rules at decision nodes, where each rule that is applied depends on the outcomes of previous rules. An HME's predictions can be interpreted by extracting the rules determining the paths from the root node to the leaf nodes.

Unfortunately, learning the set of decision node rules that give optimal assignments of observations to leaf nodes is NP-hard (Hyafil & Rivest, 1976). This issue is compounded by the non-differentiability of decision node rules. Existing approaches either build trees greedily in a node-wise manner (Breiman et al., 1984), or apply end-to-end procedures which are prone to poor local optima (Ueda et al., 2000).

This work introduces an end-to-end differentiable approach for training HMEs, using Recurrent Neural Networks (RNN) as inference models. Specifically, we cast the sequential process of an input traversing a tree as an autoregressive process. We combine this with a "tree balance" prior, which expresses the belief that the data density should be similar in all tree regions. We perform amortised inference in this model using an RNN (dubbed *RNN-Tree*) to output a variational posterior over tree routing decisions. This results in an *end-to-end differentiable* training procedure for HMEs which retains their interpretability and captures model uncertainty. We find that RNN-Trees generalise in situations where greedily learnt trees fail. We also show how an RNN-Tree can be learnt in tandem with other differentiable models: we use an RNN-Tree as an interpretable heteroscedastic noise model for a Gaussian Process (GP).

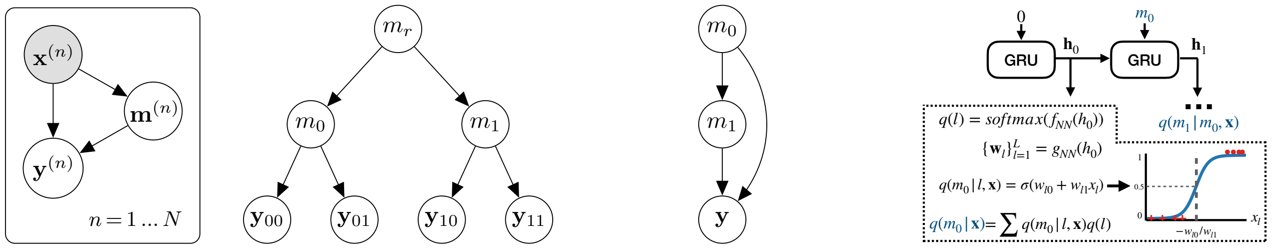
Related Work: Training HMEs is challenging due to the discrete nature of decision nodes' rules. Traditional tree learning approaches, such as CART (Breiman et al., 1984) and C4.5 (Quinlan, 1996) split greedily the input space to maximise some metric of interest (e.g. information gain). These methods are sensitive to noise in the data, suffering from high variance. Errors stemming from sub-optimal choices of earlier decision rules propagate throughout the hierarchical procedure (Hastie et al., 2001).

Jordan & Jacobs (1994) and Hehn & Hamprecht (2019) use the EM algorithm for Maximum Likelihood (ML) end-to-end learning of HMEs, treating observations' routing decisions as latent variables. However, ML exhibits a propensity for poorly performing local optima where too many mixture components are placed in some parts of input space and too few in others. Ueda et al. (2000) attempt to remedy this by adding split and merge operations to the EM procedure. Bishop & Svenskn (2002) place priors over both decision node rules' and leaf nodes' parameters. Their posteriors are approximated with Variational Inference (VI). Similarly to Jordan & Jacobs (1994) and Hehn & Hamprecht (2019), we treat each input's path through the tree as a latent vector. However, we place an explicit prior over routing decisions, resulting in a better behaved variational objective.

More recent approaches relax discrete decision rules with differentiable approximations. Yang et al. (2018) divide each input space dimension into an independent set of soft bins. Unfortunately, this results in exponential growth of the number of leaf nodes with the input dimensionality. The

[*]Equal contribution [1]University of Cambridge, Cambridge, UK [2]John A. Paulson School Of Engineering And Applied Sciences, Harvard University, MA, US [3]Microsoft Research, Cambridge, UK [4]The Alan Turing Institute, London, UK. Correspondence to: Javier Antorán <ja666@cam.ac.uk>.

Presented at the ICML 2020 Workshop on Uncertainty and Robustness in Deep Learning. Copyright 2020 by the author(s).



a) HME. b) Markovian decision nodes. c) Autoregressive decision nodes. d) RNN-Tree inference network.

Figure 1. Graphical models under consideration. In b) and c), all nodes' dependency on $\mathbf{x}$ and superscripts are omitted for clarity.

model is learnt by ML, leading to poor local optima. Frosst & Hinton (2017) preserve hierarchical structure and regularise the ML learning of tree parameters with a heuristic tree-balancing objective. We formalise this regulariser as a reverse KL divergence between a variational posterior over routing decisions and our "tree balance" prior.

2. HME Background and Notation

We introduce a dataset $\mathfrak{D} = \{X, Y\} = \{\mathbf{x}^{(n)}, \mathbf{y}^{(n)}\}_{n=1}^N$, with $\mathbf{x} \in \mathbb{R}^L$. We use bold letters to refer to vectors and capitals for matrices. An HME (Figure 1a), assigns each observation $\mathbf{x}^{(n)}$ to a leaf node with predictor $p(\mathbf{y}|\mathbf{m},\mathbf{x})$ based on the decision nodes' values $\mathbf{m}^{(n)}$. These values represent tree routing decisions. The model's likelihood is:

$$p(Y|X) = \prod_{n=1}^N \sum_{\mathbf{m}} p(\mathbf{y}^{(n)}|\mathbf{m}, \mathbf{x}^{(n)}) p(\mathbf{m}|\mathbf{x}^{(n)}). \quad (1)$$

We consider binary decision nodes. The dependencies between nodes are expressed with a tree graph, as shown in Figure 1 b). This model has a maximum depth[1] $D{=}2$ and thus 2^D leaf nodes. All nodes are conditionally independent given the value of their parent node and $\mathbf{x}$. Denoting the root node as m_r and intermediate node i as m_i, the probability of $\mathbf{x}$ being assigned to leaf $(1,0)$ is given by:

$$p(m_r{=}1, m_1{=}0|\mathbf{x}) = p(m_r{=}1|\mathbf{x}) p(m_1{=}0|m_r{=}1, \mathbf{x}) \quad (2)$$

This Markovian structure is combined with decision rules that only depend on a single dimension of the input space $l \in \{1 \dots L\}$. This yields a model that can be interpreted as a sequence of if-else rules that resemble: *"If x_l is larger than w, go left. Otherwise go right."* In this work, instead of learning decision node rule parameters, we perform probabilistic reasoning about tree routing decisions $\mathbf{m}$ directly.

3. Autoregressive, Tree-Balanced HMEs

In this section, we lay out a probabilistic framework that is conducive to efficient and reliable inference in HMEs.

Autoregressive HME: The dependencies between decision nodes in an HME can be re-cast into a more compact autoregressive structure, as shown in Figure 1 c). Here, a single binary variable m_d determines if we should go right or left at each tree depth $d \in \{0 \dots D{-}1\}$, as opposed to individual nodes. The root node's value is referred to as m_0. Thus, a complete path through the tree is given by $\mathbf{m}$, a binary vector of length D. Each routing decision m_d is dependent on all outcomes at previous depths; $\mathbf{m}$ factorises as

$$p(\mathbf{m}|\mathbf{x}) = \prod_{d=0}^{D-1} p(m_d|\{m_i\}_{i=0}^{d-1}, \mathbf{x}). \quad (3)$$

We model node routing decisions with Bernoulli distributions $p(m_d|\{m_i\}_{i=0}^{d-1}, \mathbf{x}) = \mathrm{Bern}(m_d; \rho)$. Note, not all vectors $\mathbf{m} \in \{0,1\}^D$ will be valid for non-balanced tree graphs.

The Tree Balance Prior: Direct optimisation of (1) is prone to local optima that use decision nodes inefficiently (Ueda et al., 2000; Bishop & Svenskn, 2002). We can encourage usage of our tree's full capacity by assigning similar mass under the inputs' density $p(\mathbf{x})$ to each leaf node. Specifically, we place a Binomial prior on the number of datapoints that fall on either side of each decision node:

$$p(\{m_d^{(n)}\}_{n=1}^N|X) = \binom{N}{K} 0.5^N; \quad K = \sum_{n=1}^N m_d^{(n)}. \quad (4)$$

Under the tree balance prior, node routing decisions made for different inputs are no longer independent. We refer to $[\mathbf{m}^{(1)} \dots \mathbf{m}^{(N)}]^\intercal$ as the $(N \times D)$ matrix M, with rows $M_{n,:}$ and columns $M_{:,d}$. The HME's likelihood becomes:

$$p(Y|X) = \sum_M p(Y|M) p(M|X) \quad (5)$$

where the prior factorises across tree depth but not inputs:

$$p(M|X) = \prod_{d=0}^{D-1} p(M_{:,d}|X). \quad (6)$$

4. Amortised VI for HMEs with RNN-Trees

Computing (5) is generally intractable, as marginalising M requires enumerating 2^{D^N} possible states. Instead, we introduce an inference model over node routing decisions: $q(M|X) = \prod_{n=1}^N \prod_{d=0}^{D-1} q(m_d|M_{n,0:d-1}, \mathbf{x}^{(n)})$.

The ELBO: In Appendix A, we derive the following Evidence Lower Bound (ELBO):

$$\log p(Y|X) \geq \sum_{n=1}^N \mathbb{E}_{q(\mathbf{m}|\mathbf{x}^{(n)})}[\log p(\mathbf{y}^{(n)}|\mathbf{x}^{(n)}, \mathbf{m})] - KL(q(M|X) \,||\, p(M|X)). \quad (7)$$

[1]Depth d is the number of vertices between a node and the root.

The ELBO's likelihood term factorises across datapoints and can be evaluated in closed form by enumerating all leaf nodes. The KL divergence factorises across tree depth:

$$\begin{aligned} &KL(q(M|X)\,||\,p(M|X)) \\ &= \textstyle\sum_{d=0}^{D-1} \mathbb{E}_{q(M_{:,0:d-1}|X)}[KL(q(M_{:,d}|M_{:,0:d-1}|X)\,|| \\ &\qquad p(M_{:,d}|X))]. \end{aligned} \tag{8}$$

In Appendix B, we show how Stirling's approximation to the log factorial allows us to closely approximate individual KL terms in the above sum with a differentiable expression:

$$\begin{aligned} &KL(q(M_{:,d}|M_{:,0:d-1}|X)\,||\,p(M_{:,d}|X)) \\ &\approx -\textstyle\sum_{n=1}^{N} H(q(m_d|M_{n,0:d-1},\mathbf{x}^{(n)})) \\ &\quad -NH(q(m_d|M_{n,0:d-1})) + const. \end{aligned} \tag{9}$$

$H(\cdot)$ refers to a distribution's entropy. $q(m_d|M_{n,0:d-1}) = \mathbb{E}_{p(\mathbf{x})}[q(m_d|M_{n,0:d-1},\mathbf{x})]$ is the aggregate posterior. In Appendix D, we show that eqs. (7) to (9) formalise the tree regularisation introduced by Frosst & Hinton (2017).

RNN-Tree Variational Family: Having shown that HME routing sequences can be expressed autoregressively in Section 3, a natural choice for an inference network $q_{\boldsymbol{\phi}}(\mathbf{m}|\mathbf{x})$ is an RNN. We refer to this model as an RNN-Tree. We employ GRUs. Their weights $\boldsymbol{\phi}$ are variational parameters.

As shown in Figure 1 d), the RNN-Tree is run for D steps. At step d, it uses its hidden state $\mathbf{h}_d$ to predict parameters which are used to linearly split each input dimension (w_{l0}, w_{l1}). It also outputs the parameters of a categorical distribution over which dimension should be split $q_{\boldsymbol{\phi}}(l|\mathbf{h}_d)$:

$$\{w_{l0}, w_{l1}\}_{l=1}^{L} = g_{\text{NN}}(\mathbf{h}_d;\boldsymbol{\phi}); \tag{10}$$

$$q_{\boldsymbol{\phi}}(l|\mathbf{h}_d) = \text{softmax}(f_{\text{NN}}(\mathbf{h}_d;\boldsymbol{\phi})). \tag{11}$$

Parametrising these objects explicitly allows us to recover decision rules from the RNN-Tree's predictions; the threshold for dimension l is given by ${}^{-w_{l0}}/_{w_{l1}}$. A sigmoid is used to produce dimension-wise routing decision probabilities:

$$q_{\boldsymbol{\phi}}(m_d|\mathbf{h}_d,\mathbf{x},l) = \sigma(w_{l0} + w_{l1}x_l). \tag{12}$$

The approximate posterior over node routing decisions is obtained by marginalising over the dimension which is split:

$$q_{\boldsymbol{\phi}}(m_d|\mathbf{h}_d,\mathbf{x}) = \textstyle\sum_{l=1}^{L} q_{\boldsymbol{\phi}}(m_d|\mathbf{h}_d,\mathbf{x},l)\,q_{\boldsymbol{\phi}}(l|\mathbf{h}_d). \tag{13}$$

Sampling binary outcomes $m_d \sim q_{\boldsymbol{\phi}}(m_d|\mathbf{h}_d,\mathbf{x})$ at each depth and feeding them back into the RNN results in $q_{\boldsymbol{\phi}}(m_{d+1}|\mathbf{h}_{d+1},\mathbf{x}) = q_{\boldsymbol{\phi}}(m_{d+1}|\{m_i\}_{i=0}^{d},\mathbf{x})$. However, this makes our RNN-Tree non-differentiable and requires 2^D inference network passes to compute all leaf node probabilities. Instead, decision probabilities plus quantisation noise, $q_{\boldsymbol{\phi}}(m_d|\mathbf{h}_d,\mathbf{x}) + \epsilon$, are fed back into the RNN at each depth. We choose $\epsilon \sim \mathcal{U}(-0.5, 0.5)$ in order to make the RNN robust to being fed binary outcomes at test time (Ballé et al., 2019). A lack of hard routing decisions during training allows the RNN-Tree's hidden state to track all tree paths' probabilities simultaneously. We obtain them from a single RNN-Tree roll-out by leveraging routing decisions' m_d conditional independence given the hidden state $\mathbf{h}_d$:

$$q_{\boldsymbol{\phi}}(\mathbf{m}|\mathbf{x}) = \textstyle\prod_{d=0}^{D-1} q_{\boldsymbol{\phi}}(m_d|\mathbf{h}_d,\mathbf{x}). \tag{14}$$

Recall that we require conditional distributions over routing decisions $q_{\boldsymbol{\phi}}(m_d|M_{n,0:d-1},\mathbf{x}^{(n)}))$ to compute (8) and (9). We recover these from $q_{\boldsymbol{\phi}}(m_d|\mathbf{h}_d,\mathbf{x})$ in Appendix C.

At test time, we choose the single most likely tree by replacing the softmax and sigmoid operators in (11) and (12) with $\max$ and step functions, respectively. Thus, the RNN-Tree successively queries different dimensions of a test input $\mathbf{x}^*$ with questions like "$x_l^* \lesseqgtr {}^{-w_{l0}}/_{w_{l1}}$?" D times before finally assigning the input to an expert at a leaf node.

5. Experiments

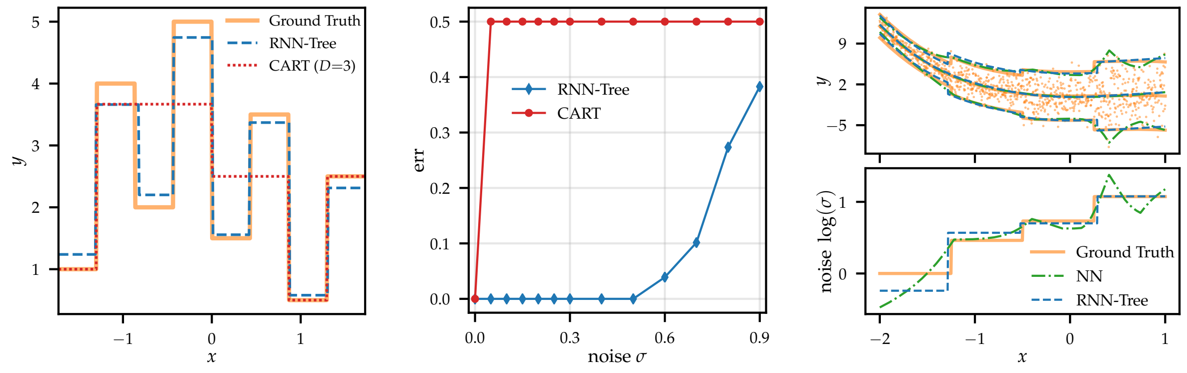
In our experiments, we employ experts that do not depend on the inputs $\mathbf{x}$, i.e. the same prediction is made for all inputs assigned to a given leaf node. We use the Gaussian likelihood for regression $p(\mathbf{y}|\mathbf{m}) = \mathcal{N}(\mathbf{y};\boldsymbol{\mu},1)$ and the Categorical for classification $p(\mathbf{y}|\mathbf{m}) = \text{Cat}(\mathbf{y};\boldsymbol{\tau})$. The Gaussian's mean $\boldsymbol{\mu}$ and categorical's class logits $\boldsymbol{\tau}$ are treated as hyperparameters $\boldsymbol{\theta} = (\boldsymbol{\mu},\boldsymbol{\tau})$. Variational parameters $\boldsymbol{\phi}$ and hyperparameters $\boldsymbol{\theta}$ are optimised jointly with RAdam (Liu et al., 2020). As a baseline, we provide results for CART, arguably the most popular tree construction approach. Full details on our experimental setup are in Appendix F.

5.1. Prediction Tasks

We construct two tasks in which the generative process of the data is tree-based. We use these to evaluate the efficiency of methods' node routing decisions. We also evaluate methods' predictive performance on 6 benchmark datasets.

Eight Level Pulse Regression: We fit CART and an RNN-Tree to an 8-level pulse function. Maximum tree depth is set to 3. Thus, full tree capacity is required to split the input space into 8 regions. Figure 2a shows that CART (red line) only splits the input space into 5 regions. The RNN-Tree (blue line) fits the data almost perfectly. CART builds trees in a greedy node-wise manner, leading it to make sub-optimal use of decision nodes. In Appendix G Figure 3, we show that CART requires minimal depth of 5 to fit this function. In contrast, an RNN-Tree makes efficient use of its decision nodes by learning all of their rules jointly.

Binary Parity Classification: We generate a dataset of 3000 random 8-bit arrays. Arrays with an even number of active bits are assigned class 1, others are assigned 0. We



(a) 8-level Pulse (b) Noisy Binary Parity (c) Heteroscedastic Regression Task

Figure 2. Experimental results.

add Gaussian noise of increasing σ to training inputs and train CART and RNN-Trees of depth 8 on this data. The test set contains all possible sequences in $[0,1]^8$. This task requires usage of full tree capacity. With noiseless data, both methods find optimal decision node rules. However, with even small amounts of noise, CART cannot solve the task. Noise causes CART to make bad node selections, from which training cannot recover, even with trees of up to depth 12. RNN-Trees show robustness to input noise, decreasing their performance gradually only after σ surpasses 0.5.

UCI Datasets: We compare the predictive performance of CART to that of RNN-Trees on six small-scale datasets from the *sklearn.datasets* package. We randomly select 80% of the data for training, 10% for validation (pruning for CART, early stopping for RNN-Tree), and 10% for testing. As shown in Table 1, RNN-Trees perform best on all tasks except "Digits", on which both methods perform comparably. End-to-end optimisation allows RNN-Trees to find better solutions than node-wise greedy approaches, even in moderately sized input spaces like "Breast" or "Digits".

Table 1. Mean results and standard deviations obtained on benchmark datasets across three random train-val-test splits. We indicate each dataset's input dimensionality and whether results indicate classification error (C) or regression RMSE (R). Lower is better.

Dataset	CART	RNN-Tree
Iris (4 C)	$\mathbf{0 \pm 0}$	$\mathbf{0 \pm 0}$
Boston (13 R)	0.311 ± 0.105	$\mathbf{0.218 \pm 0.043}$
Wine (13 C)	0.092 ± 0.026	$\mathbf{0.055 \pm 0.045}$
Diabetes (10 R)	0.855 ± 0.241	$\mathbf{0.583 \pm 0.079}$
Digits (64 C)	$\mathbf{0.161 \pm 0.022}$	0.168 ± 0.018
Breast (30 C)	0.058 ± 0.036	$\mathbf{0.023 \pm 0.008}$

5.2. Noise Estimation with an RNN-Tree

We demonstrate how RNN-Tree's differentiability benefits tasks that require end-to-end training. Specifically, we use RNN-Tree to characterise input-dependent noise. Summarising regions of high and low noise is essential when combining algorithmic predictions with human expertise. For example, Heslinga et al. (2019); Leibig et al. (2017) find that deferring high uncertainty cases for human inspection results in significant improvements to task performance.

Frequently, heteroscedasticity is modelled with separate predictors (e.g. NNs or GPs) for the mean and noise functions. But learning both jointly often results in non-identifiability, overfitting, and noise models that are not easily summarisable. We demonstrate that these issues can be mitigated by an RNN-Tree noise model. Here, we use a GP mean function. In Appendix E, we derive a variational objective based on the bound of Lázaro-Gredilla & Titsias (2011).

We construct a toy example where the true noise model is a four-step function (orange line in Figure 2c). We use a GP to model the function mean and compare a RNN-Tree noise model with a NN noise model. The NN (green line) generates a non-smooth noise function. It spikes around $x{\approx}0.5$ due to unfortunate data sampling. In contrast, our RNN-Tree (blue line) recovers the noise function more precisely, especially in high noise regions. We also compare our framework to the more common setup one NN is used to output both the mean and the noise (Kendall & Gal, 2017) - it too overfits on our task (Appendix Figure 4). Our approach shows lower variance than both baselines, which generate very different results across random restarts (Figure 5).

6. Conclusion

We re-cast HME routing decisions as an autoregressive process and introduce a tree balance prior that encourages efficient use of HME capacity. We develop an end-to-end differentiable amortised variational inference procedure for this model. Our RNN-Tree inference network parametrises an approximate posterior over tree paths. Our experiments show that the most likely tree under this distribution outperforms greedily learnt trees. In future work, we plan to make use of the full posterior distribution provided by RNN-Trees.

Acknowledgements

JA acknowledges support from Microsoft Research, through its PhD Scholarship Programme, and from the EPSRC. JY acknowledges support from NSF RI-1718306. We would like to thank Yann Dubois for a helpful discussion on optimising discrete trees.

References

Ballé, J., Laparra, V., and Simoncelli, E. End-to-end optimized image compression. In *5th International Conference on Learning Representations, ICLR 2017*, 2019.

Bishop, C. M. and Svenskn, M. Bayesian hierarchical mixtures of experts. In *Proceedings of the Nineteenth conference on Uncertainty in Artificial Intelligence*, pp. 57–64, 2002.

Breiman, L., Friedman, J., Olshen, R., and Stone, C. Classification and regression trees. 1984.

Frosst, N. and Hinton, G. Distilling a neural network into a soft decision tree. *arXiv preprint arXiv:1711.09784*, 2017.

Hastie, T., Tibshirani, R., and Friedman, J. *The Elements of Statistical Learning*. Springer Series in Statistics. Springer New York Inc., New York, NY, USA, 2001.

Hehn, T. M. and Hamprecht, F. A. End-to-end learning of deterministic decision trees. *Pattern Recognition*, pp. 612–627, 2019. ISSN 1611-3349. doi: 10.1007/978-3-030-12939-2_42. URL http://dx.doi.org/10.1007/978-3-030-12939-2_42.

Heslinga, F. G., Pluim, J. P. W., Houben, A. J. H. M., Schram, M. T., Henry, R. M. A., Stehouwer, C. D. A., van Greevenbroek, M. J., Berendschot, T. T. J. M., and Veta, M. Direct classification of type 2 diabetes from retinal fundus images in a population-based sample from the maastricht study, 2019.

Hyafil, L. and Rivest, R. L. Constructing optimal binary decision trees is np-complete. *Information Processing Letters*, 5(1):15 – 17, 1976. ISSN 0020-0190. doi: https://doi.org/10.1016/0020-0190(76)90095-8. URL http://www.sciencedirect.com/science/article/pii/0020019076900958.

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

Kendall, A. and Gal, Y. What uncertainties do we need in bayesian deep learning for computer vision? In *Advances in neural information processing systems*, pp. 5574–5584, 2017.

Lázaro-Gredilla, M. and Titsias, M. K. Variational heteroscedastic gaussian process regression. In *Proceedings of the 28th International Conference on International Conference on Machine Learning*, pp. 841–848, 2011.

Leibig, C., Allken, V., Ayhan, M. S., Berens, P., and Wahl, S. Leveraging uncertainty information from deep neural networks for disease detection. *Scientific reports*, 7(1): 1–14, 2017.

Liu, L., Jiang, H., He, P., Chen, W., Liu, X., Gao, J., and Han, J. On the variance of the adaptive learning rate and beyond. In *International Conference on Learning Representations*, 2020. URL https://openreview.net/forum?id=rkgz2aEKDr.

Quinlan, J. R. Improved use of continuous attributes in c4. 5. *Journal of artificial intelligence research*, 4:77–90, 1996.

Siegmund, D. An introduction to probability theory and its applications, volume 2 (william feller). *SIAM Rev.*, 11(2):295–297, April 1969. ISSN 0036-1445. doi: 10.1137/1011055. URL https://doi.org/10.1137/1011055.

Ueda, N., Nakano, R., Ghahramani, Z., and Hinton, G. E. Smem algorithm for mixture models. *Neural Comput.*, 12 (9):2109–2128, September 2000. ISSN 0899-7667. doi: 10.1162/089976600300015088. URL https://doi.org/10.1162/089976600300015088.

Yang, Y., Morillo, I. G., and Hospedales, T. M. Deep neural decision trees, 2018.

A. Derivation of ELBO (7)

We refer to our dataset as $\mathfrak{D}=\{X,Y\}$ with $X=\{\mathbf{x}^{(n)}\}_{n=1}^{N}$ and $Y=\{\mathbf{y}^{(n)}\}_{n=1}^{N}$. We introduce a variational family q that factorises across datapoints $q(M|X)=\prod_{n=1}^{N} q(\mathbf{m}|\mathbf{x}^{(n)})$. We assume the likelihood function at each leaf node also factorises across datapoints $p(Y|X,M)=\prod_{n=1}^{N} p(\mathbf{y}^{(n)}|\mathbf{x}^{(n)},\mathbf{m}^{(n)})$. We show that (7), which we refer to as $\mathcal{L}$, is a lower bound on $\log p(\mathfrak{D})=\log p(Y|X)$:

$$\begin{aligned}
&\mathrm{KL}(q(M|X)\,\|\,p(M|X,Y)\\
&=\mathbb{E}_{q(M|X)}[\log q(M|X)-\log p(M|X,Y)]\\
&=\mathbb{E}_{q(M|X)}\left[\log q(M|X)-\log\frac{p(Y|X,M)p(M|X)}{p(Y|X)}\right]\\
&=\mathbb{E}_{q(M|X)}[\log q(M|X)-\log p(Y|X,M)-\log p(M|X)+\log p(Y|X)]\\
&=\mathbb{E}_{q(M|X)}[-\log p(Y|X,M)]+\mathrm{KL}(q(M|X)\,\|\,p(M|X))+\log p(Y|X)\\
&=\log p(Y|X)+\mathrm{KL}(q(M|X)\,\|\,p(M|X))-\sum_{n=1}^{N}\mathbb{E}_{q(\mathbf{m}|\mathbf{x}^{(n)})}[\log p(\mathbf{y}^{(n)}|\mathbf{x}^{(n)},\mathbf{m})]\\
&=-\mathcal{L}+\log p(Y|X).
\end{aligned}\tag{15}$$

Using the non-negativity of the KL divergence, we can see that: $\mathcal{L}\le\log p(Y|X)$.

B. Deriving a Computationally Tractable Form of the KL divergence from eqs. (7) to (9)

We now discuss how to compute $\mathrm{KL}(q(M|X)\,\|\,p(M|X))$, where $p(M|X)$ is the tree balance prior (4). Recall $\mathbf{m}$ is a binary vector that refers to the routing decisions made at each depth and M is the matrix resulting from stacking these vectors for all of our training points. Also recall that M factorises autoregressively across depth $q(\mathbf{m}|\mathbf{x})=\prod_{d=0}^{D-1} q(m_d|\{m_i\}_{i=0}^{d-1},\mathbf{x})$. However, the tree balance prior does not factorise across observations. Recall that we use $M_{:,d}=\{m_d^{(n)}\}_{n=1}^{N}$ to refer to the array of all observations' routing decisions at a single tree depth d. Thus, we can write the distribution of all observations' node routing decisions as:

$$q(M|X)=\prod_{d=0}^{D-1} q(M_{:,d}|M_{:,0:d-1},\mathbf{x})\tag{16}$$

For now, we will assume our prior $p(M|X)$ also shares this form. Plugging (16) into the KL divergence:

$$\begin{aligned}
&\mathrm{KL}(q(M|X)\,\|\,p(M|X))=\\
&=\mathbb{E}_{\prod_{i=0}^{D-1} q(M_{:,i}|M_{:,0:i-1},X)}\left[\log\prod_{j=0}^{D-1} q(M_{:,j}|M_{:,0:j-1},X)-\log\prod_{r=0}^{D-1} p(M_{:,r}|M_{:,0:r-1},X)\right]\\
&=\mathbb{E}_{\prod_{i=0}^{D-1} q(M_{:,i}|M_{:,0:i-1},X)}\left[\sum_{j=0}^{D-1}\log q(M_{:,j}|M_{:,0:j-1},X)-\sum_{r=0}^{D-1}\log p(M_{:,r}|M_{:,0:r-1},X)\right]\\
&=\mathbb{E}_{\prod_{i=0}^{D-1} q(M_{:,i}|M_{:,0:i-1},X)}\left[\sum_{j=0}^{D-1}\log q(M_{:,j}|M_{:,0:j-1},X)-\log p(M_{:,j}|M_{:,0:j-1},X)\right]\\
&=\sum_{j=0}^{D-1}\mathbb{E}_{\prod_{i=0}^{j} q(M_{:,i}|M_{:,0:i-1},X)}\left[\log q(M_{:,j}|M_{:,0:j-1},X)-\log p(M_{:,j}|M_{:,0:j-1},X)\right]\\
&=\sum_{j=0}^{D-1}\mathbb{E}_{\prod_{i=0}^{j-1} q(M_{:,i}|M_{:,0:i-1},X)}\left[\mathrm{KL}(q(M_{:,j}|M_{:,0:j-1},X)\,\|\,p(M_{:,j}|M_{:,0:j-1},X)\right]\\
&=\sum_{j=0}^{D-1}\mathbb{E}_{q(M_{:,0:j-1}|X)}\left[\mathrm{KL}(q(M_{:,j}|M_{:,0:j-1},X)\,\|\,p(M_{:,j}|M_{:,0:j-1},X)\right]
\end{aligned}\tag{17}$$

This expression matches (8). We now show how to approximate $\mathrm{KL}(q(M_{:,j}|M_{:,0:j-1},X)\,\|\,p(M_{:,j}|M_{:,0:j-1},X)$. Because the tree balance prior is applied to per-depth routing decisions $M_{:,d}$, it factorises over depth $p(M|X)=\prod_{j=0}^{D-1}p(M_{:,j}|X)$. All inputs affect all routing variables. We take K to be the number of points for which the routing outcome is positive at a given depth. The prior is:

$$p(M_{:,d}|X)=\binom{N}{K}0.5^N;\quad K=\textstyle\sum_{n=1}^{N}M_{n,d} \tag{18}$$

The KL divergence of interest can be written as:

$$\mathrm{KL}(q(M_{:,d}|M_{:,0:d-1},X)\,\|\,p(M_{:,d}|X))=-H(q(M_{:,d}|M_{:,0:d-1},X))-\mathbb{E}_{q(M_{:,d}|M_{:,0:d-1},X)}[\log\binom{N}{K}+N\log 0.5] \tag{19}$$

The first term in (19) is the entropy of the variational distribution. It is separable: $H[q(M_{:,d}|M_{:,0:d-1},X)]=\sum_{n=1}^{N}H[q(m_d^{(n)}|\{m_j^{(n)}\}_{j=0}^{d-1},X)]$ and easy to compute. We focus on the second term: $\mathbb{E}_{q(M_{:,d}|M_{:,0:d-1},X)}[\log\binom{N}{K}0.5^N]$.

We leverage Stirling's approximation: $n!\approx\underbrace{n\log n-n}_{\text{Stirling's}}$ (Siegmund, 1969) in order to approximate $\log\binom{N}{K}$:

$$\begin{aligned}\log\binom{N}{K}&=\log N!-\log K!-\log(N-K)!\\&\approx N\log N-N-K\log K+K-(N-K)\log(N-K)+(N-K)\\&=N\log N-K\log K-(N-K)\log(N-K)\end{aligned} \tag{20}$$

Plugging (20) into $\mathbb{E}_{q(M_{:,d}|M_{:,0:d-1},X)}[\log\binom{N}{K}0.5^N]$, we obtain:

$$\begin{aligned}&\mathbb{E}_{q(M_{:,d}|M_{:,0:d-1},X)}[\log\binom{N}{K}0.5^N]\\&\approx E_{q(M_{:,d}|M_{:,0:d-1},X)}[-K\log K-(N-K)\log(N-K)]+N\log N+N\log 0.5\\&=N\underbrace{E_{q(M_{:,d}|M_{:,0:d-1},X)}[-\frac{K}{N}\log\frac{K}{N}-(1-\frac{K}{N})\log(1-\frac{K}{N})]}_{H[q(M_{:,d}|M_{:,0:d-1})]}+\underbrace{(-\log N+N\log N+N\log 0.5)}_{\text{constant}}\end{aligned} \tag{21}$$

We arrive at the following approximation to the KL divergence of interest:

$$\mathrm{KL}(q(M_{:,d}|M_{:,0:d-1},X)\,\|\,p(M_{:,d}|X))\approx\sum_{n=1}^{N}H[q(m_d^{(n)}|\{m_j^{(n)}\}_{j=0}^{d-1},X)]-NH[q(M_{:,d}|M_{:,0:d-1})]+\text{constant} \tag{22}$$

Here, the aggregate posterior is computed as $q(M_{:,d}|M_{:,0:d-1})=\sum_{n=1}^{N}q(M_{n,d}|M_{n,0:d-1},\mathbf{x}^{(n)})$. Expression (22) can be evaluated in closed form and is differentiable.

C. Computing our ELBO's KL Divergence from RNN-Tree Outputs

In the previous appendix, we discuss how we can tractably approximate $\mathrm{KL}(q(M|X)\,\|\,p(M|X))$. This requires access to conditional distributions over node routing decisions $q_{\boldsymbol{\phi}}(m_d|\{m_i\}_{i=0}^{d-1},\mathbf{x})$. However, during its training phase, our RNN-Tree parametrises the joint distribution over all tree paths directly $q_{\boldsymbol{\phi}}(\mathbf{m}|\mathbf{x},\mathbf{h}_d)=q_{\boldsymbol{\phi}}(\{m_d\}_{d=0}^{D-1}|\mathbf{x},\mathbf{h}_d)=\prod_{d=0}^{D-1}q(m_d|\mathbf{h}_d,\mathbf{x})$. Recall that stochasticity in the RNN-Tree's hidden state $\mathbf{h}_d$ is induced by quantisation noise $\epsilon\sim\mathcal{U}(-0.5,0.5)$. Performing a RNN-Tree roll-out can be seen as marginalising the hidden state at each depth with a single Monte Carlo sample $q(m_d|\mathbf{x})=\mathbb{E}_{p(\epsilon)}[q(m_d|\mathbf{h}_d{=}f(\epsilon),\mathbf{x})]$. We then obtain conditional node decision distributions using the product rule:

$$q(m_d|\{m_i\}_{i=0}^{d-1},\mathbf{x})=\frac{q(\{m_i\}_{i=0}^{d}|,\mathbf{x})}{q(\{m_i\}_{i=0}^{d-1}|\mathbf{x})}. \tag{23}$$

We use a similar approach to obtain conditional distributions over node routing decisions with $\mathbf{x}$ marginalised:

$$\begin{aligned}q(m_d|\{m_i\}_{i=0}^{d-1})&=\frac{q(\{m_i\}_{i=0}^{d})}{q(\{m_i\}_{i=0}^{d-1})}=\frac{\mathbb{E}_{p(\mathbf{x})}[q(\{m_i\}_{i=0}^{d}|\mathbf{x})]}{\mathbb{E}_{p(\mathbf{x})}[q(\{m_i\}_{i=0}^{d-1}|\mathbf{x})]}=\frac{\mathbb{E}_{p(\mathbf{x})}[q(\{m_i\}_{i=0}^{d-1}|\mathbf{x})q(m_d|\mathbf{x})]}{\mathbb{E}_{p(\mathbf{x})}[q(\{m_i\}_{i=0}^{d-1}|\mathbf{x})]}\\&\approx\frac{\sum_{n=1}^{N}q(\{m_i\}_{i=0}^{d-1}|\mathbf{x}^{(n)})q(m_d|\mathbf{x}^{(n)})}{\sum_{n=1}^{N}q(\{m_i\}_{i=0}^{d-1}|\mathbf{x}^{(n)})}\end{aligned} \tag{24}$$

D. Relation of our ELBO to the Objective of (Frosst & Hinton, 2017)

Following the notation of (Frosst & Hinton, 2017), we revert to the non-autoregressive tree structure of Figure 1 b) and refer to a single decision node as i. $P^i(\mathbf{x})$ is the probability of an input $\mathbf{x}$ reaching node i and $p_i(\mathbf{x})$ is the probability of that input being assigned a positive outcome by node i; $p(m_i{=}1){=}p_i$. The objective minimised by (Frosst & Hinton, 2017) is:

$$\mathcal{L}_a = \underbrace{\sum_{\mathbf{x}} -\log\left(\sum_{\text{leaf}} p(\text{leaf}|\mathbf{x}) \log p(\mathbf{y}|\text{leaf})\right)}_{\text{I}} \underbrace{- N\sum_i \lambda(i)\left(0.5\log(\alpha_i) + 0.5\log(1-\alpha_i)\right)}_{\text{II}} \tag{25}$$

Term I resembles the likelihood term from (7), with an additional log introduced. In term II, α_i is obtained for each decision node i as:

$$\alpha_i = \frac{\sum_{n=1}^{N} P^i(\mathbf{x})p_i(\mathbf{x})}{\sum_{n=1}^{N} P^i(\mathbf{x})} \tag{26}$$

This resembles (24). Drawing a parallelism with our proposed variational formulation, the average of α_i for all nodes at a certain depth d yields our conditional aggregate posterior: $q(m_d|\{m_i\}_{i=0}^{d-1}) = (1/2^d)\cdot\sum_{i\in\text{depth}d}\alpha_i$. Indeed, (Frosst & Hinton, 2017) heuristically set the weighing term $\lambda_i = 1/2^{d_i}$, where d_i is the depth of node i. Returning to our regular notation, (Frosst & Hinton, 2017)'s regularisation term II takes the form:

$$N\,\mathbb{E}_{p(M_{:,d})}[q(M_{:,d}|M_{:,0:d-1})] \tag{27}$$

with $p(M_{:,d}) = 0.5$. This function is very similar to $NH[q(M_{:,d}|M_{:,0:d-1})]$, the second term in our KL divergence (22). Both are convex and have the same optima.

E. ELBO Derivation for our Heteroscedastic GP from Section 5.2

Suppose we are given a dataset $\mathfrak{D} = \{X, \mathbf{y}\} = \{\mathbf{x}^{(n)}, y^{(n)}\}_{n=1}^N$. We use bold letters to refer to vectors and capitals for matrices, $X \in \mathbb{R}^{N\times L}, \mathbf{y}\in\mathbb{R}^{N\times 1}, \mathbf{x}\in\mathbb{R}^{L\times 1}$. We model the dataset as a latent function with some input-dependent noise: $y^{(n)} = f(\mathbf{x}^{(n)}) + \epsilon^{(n)}$, where $\epsilon^{(n)} \sim \mathcal{N}(0, r(\mathbf{x}^{(n)}))$. Here, $r(\mathbf{x})$ is modelled with an RNN-Tree whose parameters are θ, $r(\mathbf{x}) = e^{2T_\theta(\mathbf{x})}$). We place a GP prior on the function, $f \sim GP(0, K(\mathbf{x}, \mathbf{x}'))$. In addition, we put tree balance priors introduced in Section 3 over intermediate leave nodes $\mathbf{m}$. Since the joint posterior $p(\mathbf{f}, \mathbf{m}|X, \mathbf{y})$ is intractable, we follow (Lázaro-Gredilla & Titsias, 2011) in approximating it as $q(\mathbf{f})\cdot q(\mathbf{m})$. Using Jensen's inequality, we have

$$log(\mathbf{y}|X) \geq \int q(\mathbf{f})q(\mathbf{m})\log\frac{p(\mathbf{y}|\mathbf{f},X,\mathbf{m})p(\mathbf{f}|X)p(\mathbf{m})}{q(\mathbf{f})q(\mathbf{m})}d\mathbf{f}d\mathbf{m} = \mathcal{L}_1(q(\mathbf{f}), q(\mathbf{m})) \tag{28}$$

To obtain a tighter lower bound, we replace $q(\mathbf{f})$ with $q^*(\mathbf{f})$ which maximises $\mathcal{L}_1(q(\mathbf{f}), q(\mathbf{m})$ for a given $q(\mathbf{m})$. By variational Bayesian theory, we have the optimal distribution of $q(\mathbf{f})$ as

$$q^*(\mathbf{f}) = \frac{1}{\log Z(q(\mathbf{m}))}e^{\int q(\mathbf{m})\log p(y|\mathbf{f},X,\mathbf{m})p(\mathbf{f}|X)d\mathbf{m}}$$

where $\log Z(q(\mathbf{m}))$ is a normalising constant and $\log Z(q(\mathbf{m})) = \int e^{\int q(\mathbf{m})\log p(\mathbf{y}|\mathbf{f},X,\mathbf{m})d\mathbf{m}}p(\mathbf{f}|X)d\mathbf{f}$.

Plugging $q^*(\mathbf{f})$ back into (28), we have

$$\begin{aligned}\log(\mathbf{y}|X) \geq \mathcal{L}_1(q^*(\mathbf{f}), q(\mathbf{m})) &= \int q^*(\mathbf{f})\frac{\log Z(q(\mathbf{m}))q^*(\mathbf{f})}{q^*(\mathbf{f})}d\mathbf{f} - KL(q(\mathbf{m})\|p(\mathbf{m}))\\ &= \log Z(q(\mathbf{m})) - KL(q(\mathbf{m})\|p(\mathbf{m}))\\ &= \mathcal{L}_2(q(\mathbf{m})) \geq \mathcal{L}_1(q(\mathbf{f}), q(\mathbf{m}))\end{aligned}$$

$\log Z(q(\mathbf{m}))$ can be calculated in closed form since $q(\mathbf{m})$ is categorical. Recall that $r(\mathbf{x}^{(n)}) = e^{2T(\mathbf{x}^{(n)})}$

$$\begin{aligned}\log Z(q(\mathbf{m})) &= \log \int e^{\sum_n^N \sum_j^{2^D} q(m_j|\mathbf{x}^{(n)})\left[-\frac{1}{2}(y^{(n)}-f^{(n)})^2 e^{-2m_j} - m_j - \frac{1}{2}\log 2\pi\right]} \mathcal{N}(\mathbf{f}|0,\mathbf{K})d\mathbf{f} \\ &= \log \int e^{\sum_n^N \left[-\frac{1}{2}(y^{(n)}-f^{(n)})^2 \left(\sum_j q(m_j|\mathbf{x}^{(n)})e^{-2m_j}\right) - \sum_j q(m_j|\mathbf{x}^{(n)})m_j - \frac{1}{2}log 2\pi\right]} \mathcal{N}(\mathbf{f}|0,\mathbf{K})d\mathbf{f} \\ &= \log \int \mathcal{N}(\mathbf{y}|\mathbf{f},\mathbf{R})e^{\mathcal{C}}\mathcal{N}(\mathbf{f}|0,\mathbf{K})d\mathbf{f}\end{aligned}$$

where $\mathbf{R}$ is a diagonal matrix with $R_{nn} = (\sum_j q(m_j|\mathbf{x}^{(n)})e^{-2m_j})^{-1}$, and $C = \frac{1}{2}\sum_n \log R_{nn} - \sum_j \sum_n q(m_j|x^{(n)})m_j$ Hence

$$\mathcal{L}_2(q(\mathbf{m})) = \log \mathcal{N}(\mathbf{y}|0,\mathbf{K}+\mathbf{R}) + C - KL(q(\mathbf{m})\|p(\mathbf{m}))$$

When inferring the posterior predictive distribution for a new data points X', instead of marginalising over tree models $q(\mathbf{m}_j)$, we let $q(\mathbf{m}'|\mathbf{x}') = \arg\max_{\mathbf{m}_j} q(\mathbf{m}|\mathbf{x}')$. We then have

$$p(\mathbf{f}'|X,X',\mathbf{y}) = \int p(\mathbf{f},\mathbf{f}'|X,X',\mathbf{y},\mathbf{m}')q^*(\mathbf{f})d\mathbf{f} = \mathcal{N}(\mathbf{f}'|\bar{\mathbf{f}}', cov(\mathbf{f}'))$$

where $\bar{\mathbf{f}}' = \mathbf{K}(X',X)(\mathbf{K}(X,X)+\mathbf{R}')^{-1}\mathbf{y}$, $cov(\mathbf{f}') = \mathbf{K}(X',X') - \mathbf{K}(X',X)(\mathbf{K}+\mathbf{R}^*)^{-1}\mathbf{K}(X,X')$, and $\mathbf{R}$ is a diagonal matrix with diagonal terms $\mathbf{R}'_{ii} = e^{2\max_m q(m|\mathbf{x}')}$

F. Experiment Setup

We provide additional details which are not included in the main text on the experiments performed in Section 5.

F.1. Eight Level Pulse Regression

In the this task, we use a dataset given by the following function

$$y = \begin{cases} 1 & -4 \le x < -3 \\ 4 & -3 \le x < -2 \\ 2.5 & -2 \le x < -1 \\ 5 & -1 \le x < 0 \\ 1.5 & 0 \le x < 1 \\ 3.5 & 1 \le x < 2 \\ 0.5 & 2 \le x < 3 \\ 2.5 & 3 \le x \le 4 \end{cases}$$

The inputs $\{x\}_{i=1}^N$ is uniformly spaced in $[-4, 4]$ with $N = 400$.

F.2. Binary Parity Classification

The dataset is composed of 3000 samples drawn from:

$$\mathbf{x}^* = \mathbf{b} + \epsilon; \quad \mathbf{b} \sim \prod_{j=1}^{8} \text{Bern}(b_j; 0.5); \quad \epsilon \sim \mathcal{N}(0, \sigma)$$

σ is varied in the range $[0, 0.9]$. Targets are computed from $\mathbf{b}$, before adding Gaussian noise.

F.3. Noise Estimation Task

For this task, we generate the dataset as $y = f(x) + \epsilon$ where $f(x) = -x^3 + x^2$, $\epsilon \sim \mathcal{N}(0, \sigma^2)$ and

$$\log \sigma = \begin{cases} 0 & -2 \leq x < -5/4 \\ 2/3 & -5/4 \leq x < -1/2 \\ 2 & -1/2 \leq x < 1/4 \\ 10/3 & 1/4 \leq x \leq 1 \end{cases}$$

The inputs $\{x\}_{i=1}^N$ is uniformly spaced in $[-2, 1]$ with $N = 1,200$.

We perform joint inference of the mean function (GP) and the noise function (an RNN-Tree or an NN) but with different learning rate. To select the hyperparameter for an NN, we randomly select 80% of the data for training, 10% for validation and 10% for testing. For the NN, we tested learning rate in $\eta \in \{0.001, 0.003, 0.005, 0.01\}$, architecture in 1 hidden layer with 16 nodes and 1 hidden layer with 32 nodes. For Heteroscedastic NN (details in Appendix G.2), in addition to the learning rate and the architecture same as above, we tested $l2$ penalisation $\lambda \in \{0.001, 0.003, 0.005, 0.01\}$. The optimal parameter setting is selected by the best test loglikelihood.

G. Additional Experimental Results

G.1. Eight Level Pulse Regression

Figure 3 shows the predictive performance of CART with various maximum depth. We see that CART requires a depth of at least 5 to fit the function.

G.2. Noise Estimation Task

We compare our framework to the common used practice described in (Kendall & Gal, 2017) where one NN predicts the noise and the mean function at the same time. Denote the parameters of the NN as θ, the input as $\mathbf{x}$ and the target variable as $\mathbf{y}$. The NN outputs $\mathbf{f}_\theta(\mathbf{x}) = [\hat{\mathbf{f}}, \log \hat{\sigma}]$ where $\hat{\mathbf{x}}$ denotes the predictive mean and $\hat{\sigma}^2$ denotes the estimated noise. The model is learnt by minimising the following

$$\mathcal{L}_{NN}(\theta) = \frac{1}{N} \sum_{n=1}^{N} \left[\frac{1}{2\sigma(\mathbf{x}^{(n)})^2} \|\mathbf{y}^{(n)} - \mathbf{f}(\mathbf{x}^{(n)})\|_2 + \log \sigma(\mathbf{x}^{(n)}) \right] + \lambda \|\theta\|_2$$

where $\lambda\|\theta\|_2$ is a L_2 penalisation term. Figure 4 shows that Hetero-NN overfits as it has multiple spikes. Such behaviour is not desirable for noise summarisation.

Additionally, Figure 5 shows that our framework has lower variance than the other two given multiple random restarts. Hetero-NN has the highest variance among all three.

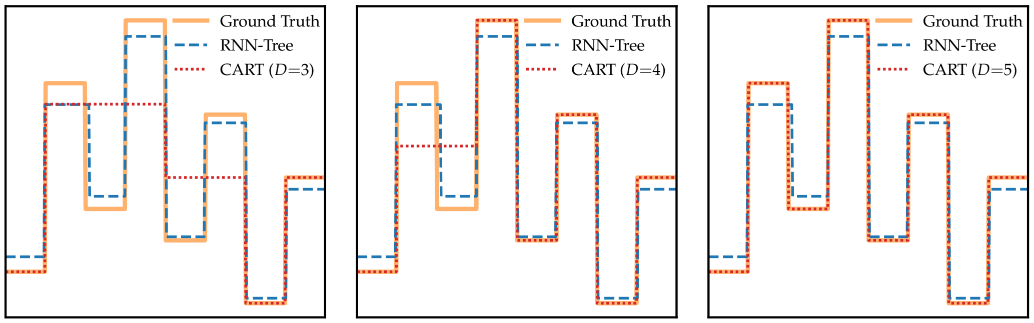


Figure 3. Prediction performance of CART of maximum depth D=3, 4, 5.

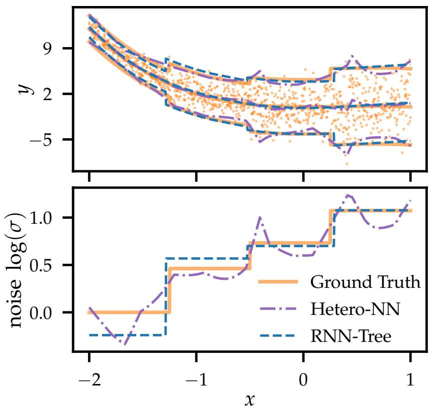


Figure 4. Predictive performance of Hetero-NN.

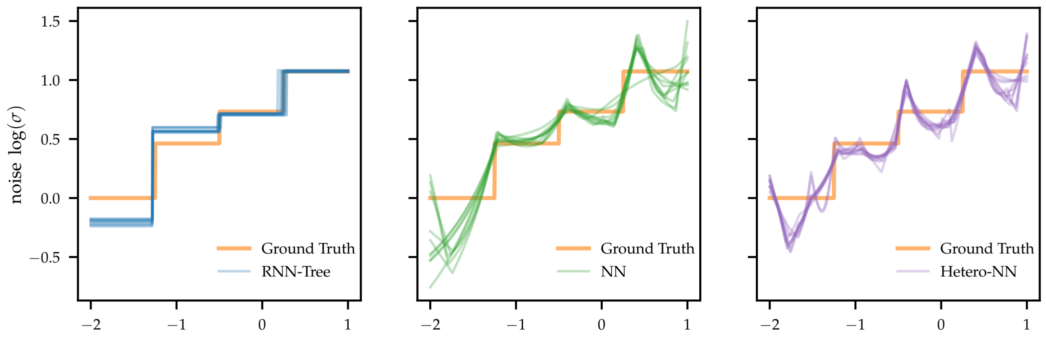


Figure 5. Estimated noise function given multiple ranodm restarts.