

Neural Regression Trees

Shahan Ali Memon[†], Wenbo Zhao[†], Bhiksha Raj and Rita Singh
Carnegie Mellon University
Pittsburgh, Pennsylvania 15213, United States
{samemon, wzhaol, bhikshar, rsingh}@andrew.cmu.edu

Abstract—Regression-via-Classification (RvC) is the process of converting a regression problem to a classification one. Current approaches for RvC use ad-hoc discretization strategies and are suboptimal. We propose a neural regression tree model for RvC. In this model, we employ a joint optimization framework where we learn optimal discretization thresholds while simultaneously optimizing the features for each node in the tree. We empirically show the validity of our model by testing it on two challenging regression tasks where we establish the state of the art.

Index Terms—Regression-via-Classification, Regression Trees, Discretization

I. INTRODUCTION

One of the most challenging problems in machine learning is that of predicting a numeric attribute y of a datum from other features x , a task commonly referred to as regression¹. The relationship between the features and the predicted variable (which, for want of a better term, we will call a “response”) is generally unknown and may not be deterministic. The general approach to the problem is to assume a formula for the relationship, and to estimate the details of the formula from training data. Linear regression models assume a linear relationship between the features and the response. Other models such as neural networks assume a non-linear relationship. The problem here is that the model parameters that are appropriate for one regime of the data may not be appropriate for other regimes. Statistical fits of the model to the data will minimize a measure of the overall prediction error, but may not be truly appropriate for any specific subset of the data. Non-parametric regression models such as kernel regressions and regression trees attempt to deal with this by partitioning the feature space, and computing separate regressors within each partition. For high-dimensional data, however, any computed partition runs the risk of overfitting to the training data, necessitating simplifying strategies such as axis-aligned partitions [1], [2].

An alternative strategy, and one that is explored in this paper, is to *partition* the space based on the *response* variable. Formally, given a response variable y that takes values in some range (y_{\min}, y_{\max}) , we find a set of threshold values t_0, \dots, t_N , and map the response variable into bins as $y \mapsto$

C_n if $t_{n-1} < y \leq t_n$ for $n = 1, \dots, N$. This process, which effectively converts the continuous-valued response variable y into a categorical one C , is often referred to as *discretization*. The new response variable C is, in fact, not merely categorical, but *ordinal*, since its values can be strictly ordered. In order to determine the value y for any x we must now find out which bin C_n the feature x belongs to; once the appropriate bin has been identified, the actual estimated y can be computed in a variety of ways, e.g., the mean or median of the bin. Consequently, the problem of regression is transformed into one of classification. This process of converting a regression problem to a classification problem is uncommonly known as *Regression-via-Classification (RvC)* [3].

Naive implementation of RvC can, however, result in very poor regression. Inappropriate choice of bin boundaries $\{t_i\}$ can result in bins that are too difficult to classify (since classification accuracy actually depends on the distribution of feature x within the bins). Alternatively, although permitting near-perfect classification, the bins may be too wide, and the corresponding “regression” may be meaningless. Ideally, the RvC method must explicitly optimize the bin boundaries for both classification accuracy *and* regression accuracy. In addition, the actual classifier employed cannot be ignored; since the decision boundaries are themselves variable, the classifier and the boundaries must conform to one another.

We propose a hierarchical tree-structured model for RvC, which addresses all of the problems mentioned above. Jointly optimizing all the variables and classifiers involved against the classification accuracy and regression accuracy is a combinatorially hard problem. Instead of solving this directly, inspired by the original idea of regression trees [2], we follow a greedy strategy of hierarchical binary partition of the response variable y , where each split is locally optimized. This results in a tree-structured RvC model with a classifier at each node. We employ a simple margin-based linear classifier for each classification; however, the *features* derived from the data may be optimized for classification. Moreover, the structure of our model affords us an additional optimization: instead of using a single generic feature for classification, we can now optimize the features extracted from the data *individually* for each classifier in the tree. Since we employ a neural network to optimize the features

[†]Equal Contribution. ISBN 978-1-7281-1985-4/19/\$31.00 ©2019 IEEE

¹Terminology that is borrowed from the statistics literature

for classification, we refer to this framework as a *Neural Regression Tree* (NRT).

To demonstrate the utility of the proposed approach we conduct experiments on a pair of notoriously challenging regression tasks: estimating the age and height of speakers from their voice. We show that our model performs significantly better than other regression models, including those that are known to achieve the current state-of-the-art in these problems.

II. RELATED WORK

a) Regression Trees: Tree-structured models have been around for a long time. Among them, the most closely related are the regression trees. A regression tree is a regression function in which the partition is performed on features x instead of response variable y . The first regression tree algorithm was presented by [4], where they propose a greedy approach to fit a piece-wise constant function by recursively splitting the data into two subsets based on partition on the features x . The optimal split is a result of minimizing the impurity which defines the homogeneity of the split. This algorithm sets the basis for a whole line of research on classification and regression trees. Improved algorithms include CART [5], ID3 [2], m5 [6], and C4.5 [7]. Recent work combines the tree-structure and neural nets to gain the power of both structure learning and representation learning. Such work includes the convolutional decision trees [8], neural decision trees [9], [10], adaptive neural trees [11], deep neural decision forests [12], and deep regression forests [13].

We emphasize that there is a fundamental difference between our approach and the traditional regression tree based approaches: instead of making the split based on the feature space, our splitting criteria is based on the response variables, enabling the features to adapt to the partitions of response variables.

b) Regression via Classification (RvC): The idea for RvC was presented by [14]. Their algorithm is based on k-means clustering to categorize numerical variables. Other conventional approaches [3], [15] for discretization of continuous values are based on equally probable (EP) or equal width (EW) intervals, where EP creates a set of intervals with same number of elements, while EW divides into intervals of same range. These approaches are ad-hoc. Instead, we propose a discretization strategy to learn the optimal thresholds by improving the neural classifiers.

c) Ordinal Regression: Because our model is essentially a method to discretize continuous values into ordered partitions, it can be somewhat compared to ordinal regression [16]. Ordinal regression is a class of regression analysis that operates on data where the response variable is categorical but exhibits an order relation. Naive approaches for ordinal regression often simplify the problem by ignoring the ordering information

and treating the response variables as nominal categories. A slightly sophisticated method [17] uses decomposition of the response variable into several binary ones (such as via binary ordered partitions) and estimating them using multiple models. Another relevant class of approaches uses the threshold models [17]. These approaches resemble our approach in that they assume unobserved continuous response variables underlying the ordinal responses, and use thresholds to discretize them, where a variety of models (such as support vector machines and perceptrons) are used to model the underlying response variables.

Our proposed model shares many characteristics with these approaches. However, one big difference is that in ordinal regression, the partitions are predefined by the domain problem and may not be optimized for statistical inference. Our model, on the other hand, is based on a data-driven partition strategy where partitions are optimized for more discriminative representation of the data hierarchy and better performance at the inference time.

III. NEURAL REGRESSION TREE

In this section, we formulate the neural regression tree model for optimal discretization of the response variables in RvC, and provide algorithms to optimize the model.

A. Partition

The key aspect of an RvC system is its method of partition Π . We define the partition Π on a set $\mathbb{Y} \subset \mathbb{R}$ as

$$\Pi(\mathbb{Y}) = \{C_1, \dots, C_N\}$$

satisfying $\bigcup_{n=1}^N C_n = \mathbb{Y}$ and C_n s are mutually disjoint. When acting on a $y \in \mathbb{Y}$, $\Pi(y) := C_n$ subjected to $y \in C_n$.

B. Model Formulation

Formally, following the description of RvC in Section I, an RvC framework consists of two main rules: a classification rule and a regression rule. The classification rule classifies an input x into disjoint bins, i.e., $h_\theta : x \mapsto \{C_1, \dots, C_N\}$ with parameter θ , where $C_n = \Pi(y)$ corresponds to $t_{n-1} < y \leq t_n$ for $n = 1, \dots, N$. The regression rule $r : (x, C_n) \mapsto (t_{n-1}, t_n]$ maps the combination of input x and class C_n onto the interval $(t_{n-1}, t_n]$. Then, the combined RvC rule that predicts the value of the response variable for an input x is

$$\hat{y}(x) = r(x, h_\theta(x)). \quad (1)$$

Instead of making a hard assignment of bins, alternatively, the classification rule h_θ may make a ‘‘soft’’ assignment by mapping an input x onto the N -dimensional probability simplex, i.e., $h_\theta : x \mapsto \mathbb{P}_N$ where \mathbb{P}_N represents the set of N -dimensional non-negative vectors whose entries sum to 1. The output of this classification rule is, therefore, the vector of *a posteriori*

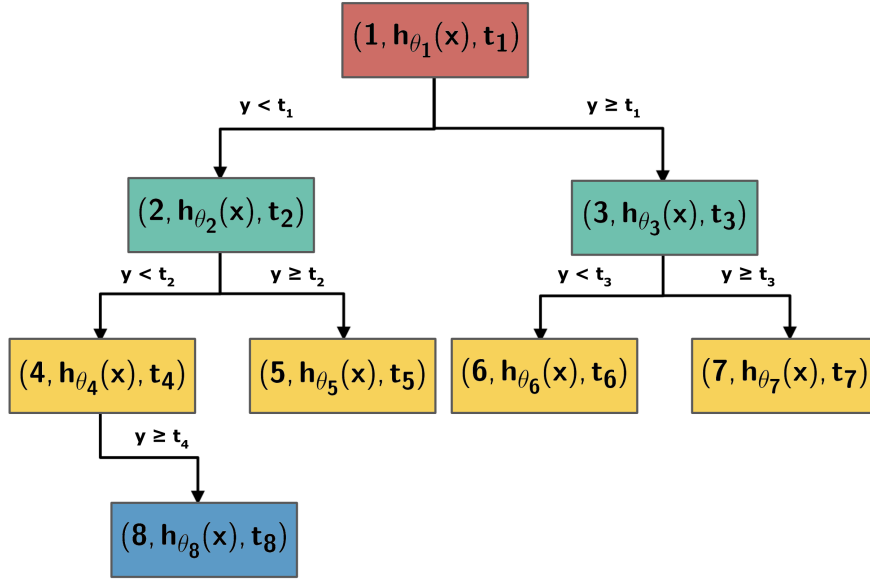


Fig. 1: Illustration of neural regression tree. Each node is equipped with a neural classifier h_θ . The splitting threshold t is dependent on the response variable y and is locally optimized.

probabilities over the N classes, i.e., $h_\theta^n(x) = P(C_n|x)$ where $h_\theta^n(x)$ represents the n^{th} component of $h_\theta(x)$. Hence, the RvC rule is given by

$$\hat{y}(x) = \mathbb{E}_{C_n}[r(x, C_n)] = \sum_{n=1}^N h_\theta^n(x) r_{C_n}(x), \quad (2)$$

where $r_{C_n}(\cdot) := r(\cdot, C_n)$ fixes the second coordinate of r at C_n . Defining an error $\mathcal{E}(y, \hat{y}(x))$ between the true y and the estimated $\hat{y}(x)$, our objective is to learn the thresholds $\{t_0, \dots, t_N\}$ and the parameters $\{\theta_0, \dots, \theta_N\}$ such that the expected error is minimized

$$\{t_n^*\}, \{\theta_n^*\} \leftarrow \arg \min_{t, \theta} \mathbb{E}_x [\mathcal{E}(y, \hat{y}(x))]. \quad (3)$$

Note that the number of thresholds $(N + 1)$ too is a variable that may either be manually set or explicitly optimized. In practice, instead of minimizing the *expected* error, we minimize the *empirical* average error $\text{avg}(\mathcal{E}(y_i, \hat{y}(x_i)))$ computed over a training set.

However, joint optimization of $\{t_n\}$ and $\{\theta_n\}$ is a hard problem as it scales exponentially with n . To solve this problem we adopt the idea of divide and conquer and recast the RvC problem in terms of a binary classification tree, where each of the nodes in the tree is greedily optimized. The structure of the proposed binary tree is shown in Figure 1.

We now describe the tree-growing algorithm. For notational convenience the nodes have been numbered such that for any two nodes n_1 and n_2 , if $n_1 < n_2$, n_1 occurs either to the left of n_2 or above it in the tree. Each node n in the tree has an associated threshold t_n , which is used to partition the

data into its two children n' and n'' (we will assume w.l.o.g. that $n' < n''$). A datum (x, y) is assigned to the “left” child n' if $y < t_n$, and to the “right” child n'' otherwise. The actual partitions of the response variable are the leaves of the tree. To partition the data, each node carries a classifier $h_{\theta_n} : x \mapsto \{n', n''\}$, which assigns any instance with features x to one of n' or n'' . In our instantiation of this model, the classifier h_{θ_n} is a neural classifier that not only classifies the features but also adapts and refines the features to each node.

Given an entire tree along with all of its parameters and an input x , we can compute the *a posteriori* probability of the partitions (i.e. the leaves) as follows. For any leaf l , let $l_0 \rightarrow \dots \rightarrow l_p$ represent the chain of nodes from root l_0 to the leaf itself $l_p \equiv l$. The *a posteriori* probability of the leaf is given by $P(l|x) = \prod_{r=1}^p P(l_r | l_{r-1}, x)$, where each $P(l_r | l_{r-1}, x)$ is given by the neural classifier on node l_{r-1} . Substitution into (2) yields the final regressed value of the response variable

$$\hat{y}(x) = \sum_{l \in \text{leaves}} P(l|x) r_l(x), \quad (4)$$

where $r_l(x) := r(x, l)$, in our setting, is simply the mean value of the leaf bin. Other options include the center of gravity of the leaf bin, using a specific regression function, etc.

C. Learning the Tree

We learn the tree in a greedy manner, optimizing each node individually. The procedure to optimize an individual node n is as follows. Let $\mathbb{D}_n = \{(x_i, y_i)\}$ represent the set of training instances arriving at node n . Let n' and n'' be the

children induced through threshold t_n . In principle, to locally optimize n , we must minimize the average regression error $\mathcal{E}(\mathbb{D}_n; t_n, \theta_n) = \text{avg}(\mathcal{E}(y, \hat{y}_n(x)))$ between the true response values y and the estimated response $\hat{y}_n(x)$ computed using only the subtree with its root at n . In practice, $\mathcal{E}(\mathbb{D}_n; t_n, \theta_n)$ is not computable, since the subtree at n is as yet unknown. Instead, we will approximate it through the *classification* accuracy of the classifier at n , with safeguards to ensure that the resultant classification is not trivial and permits useful regression.

Let $y(t_n) = \text{sign}(y - t_n)$ be a binary indicator function that indicates if an instance (x, y) has to be assigned to child n' or n'' . Let $\mathcal{E}(y(t_n), h_{\theta_n}(x))$ be a qualifier of the classification error (which can be binary cross entropy loss, hinge loss, etc.) for any instance (x, y) . We define the classification loss at node n as

$$E_{\theta_n, t_n} = \frac{1}{|\mathbb{D}_n|} \sum_{(x, y) \in \mathbb{D}_n} \mathcal{E}(y(t_n), h_{\theta_n}(x)), \quad (5)$$

where $|\mathbb{D}_n|$ is the size of \mathbb{D}_n . The classification loss (5) cannot be directly minimized w.r.t t_n , since this can lead to trivial solutions, e.g., setting t_n to an extreme value such that all data are assigned to a single class. While such a setting would result in perfect classification, it would contribute little to the regression. To prevent such solutions, we include a *triviality* penalty \mathcal{T} that attempts to ensure that the tree remains balanced in terms of the number of instances at each node. For our purpose, we define the triviality penalty at any node as the entropy of the distribution of instances over the partition induced by t_n (other triviality penalties such as the Gini index [5] may also apply though)

$$\mathcal{T}(t_n) = -p(t_n) \log p(t_n) - (1 - p(t_n)) \log(1 - p(t_n)), \quad (6)$$

where

$$p(t_n) = \frac{\sum_{(x, y) \in \mathbb{D}_n} (1 + y(t_n))}{2|\mathbb{D}_n|}.$$

The overall optimization of node n is performed as

$$\theta_n^*, t_n^* = \arg \min_{\theta_n, t_n} \lambda E_{\theta_n, t_n} + (1 - \lambda) \mathcal{T}(t_n), \quad (7)$$

where $\lambda \in (0, 1)$ is used to assign the relative importance of the two components of the loss, and is a hyper-parameter to be tuned.

In the optimization of (7), the loss function depends on t_n through $y(t_n)$, which is a discontinuous function of t_n . To get around this difficulty of optimizing (7), we have two possible ways: the **scan method** and the **gradient method**. In the first, we can *scan* through all possible values of t_n to select the one that results in the minimal loss. Alternatively, a faster gradient-descent approach is obtained by making the objective differentiable w.r.t. t_n . Here the discontinuous function $\text{sign}(y - t_n)$ is approximated by a differentiable relaxation: $y(t_n) = 0.5(\tanh(\beta(y - t_n)) + 1)$, where β controls the steepness of

Input: \mathbb{D}

Parameter: $\{t_n\}, \{\theta_n\}$

Output: $\{t_n^*\}, \{\theta_n^*\}$

Function BuildTree(\mathbb{D}_n)

 Initialize t_n, θ_n

$t_n^*, \theta_n^* \leftarrow \text{NeuralClassifier}(\mathbb{D}_n, t_n, \theta_n)$

$\mathbb{D}_{n'}, \mathbb{D}_{n''} \leftarrow \text{Partition}(\mathbb{D}_n, t_n^*)$

for \mathbb{D}_n in $\{\mathbb{D}_{n'}, \mathbb{D}_{n''}\}$ **do**

if \mathbb{D}_n is pure **then**

 | continue

else

 | BuildTree(\mathbb{D}_n)

end

end

BuildTree(\mathbb{D})

Algorithm 1: Learning neural regression tree. The tree is built recursively. For each node n , it adapts and classifies the features, and partitions the data based on the locally optimal classification threshold.

the function and must typically be set to a large value ($\beta = 10$ in our settings) for close approximation. The triviality penalty is also redefined (to be differentiable w.r.t. t_n) as the proximity to the median $\mathcal{T}(t_n) = \|t_n - \text{median}(y | (x, y) \in \mathbb{D}_n)\|_2$, since the median is the minimizer of (6). We use coordinate descent to optimize the resultant loss.

Once optimized, the data \mathbb{D}_n at n are partitioned into n' and n'' according to the threshold t_n^* , and the process proceeds recursively down the tree. The growth of the tree may be continued until the regression performance on a held-out set saturates. Algorithm 1 describes the entire training algorithm.

IV. EXPERIMENTS

We consider two regression tasks in the domain of speaker profiling—age estimation and height estimation from voice. The two tasks are generally considered two of the challenging tasks in the speech literature [18]–[26].

We compare our model with 1) a regression baseline using the support vector regression (SVR) [27], 2) a regression tree baseline using classification and regression tree (CART) [5], and 3) a neural net baseline with multilayer perceptron (MLP) structure. Furthermore, in order to show the effectiveness of the “neural part” of our NRT model, we further compare our neural regression tree with a third baseline 4) regression tree with the support vector machine (SVM-RT).

A. Data

To promote a fair comparison, we select two well-established public datasets in the speech community. For age estimation,

TABLE I: Fisher Dataset Partitions

	# of Speakers / Utterances	
	Male	Female
Train	3,100 / 28,178	4,813 / 45,041
Dev	1,000 / 9,860	1,000 / 9,587
Test	1,000 / 9,813	1,000 / 9,799

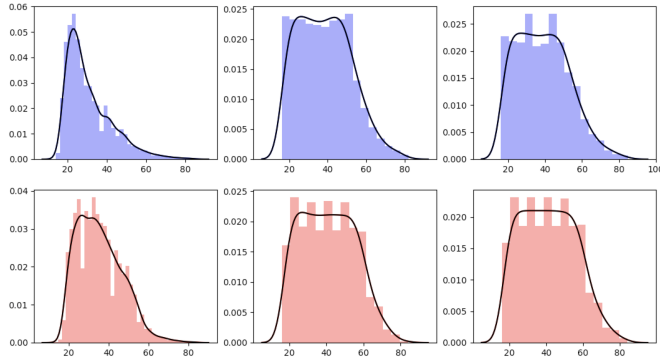


Fig. 2: Age distribution (in percentages) for male (*Top*) and female (*Bottom*) speakers for the Fisher database for train (*Left*), development (*Center*) and test (*Right*) sets. The horizontal axis is age.

TABLE II: NIST-SRE8 Dataset Stats

# of Speakers / Utterances	
Male	Female
384 / 33,493	651 / 59,530

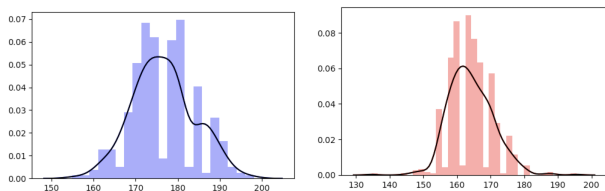


Fig. 3: Height distribution (in percentages) for male (*Left*) and female (*Right*) speakers for the NIST-SRE8 database. The horizontal axis is height.

we use the Fisher English corpus [28]. It consists of a 2-channel conversational telephone speech for 11,971 speakers, comprising of a total of 23,283 recordings. After removing 58 speakers with no age specified, we are left with 11,913 speakers with 5,100 male and 4,813 female speakers. To the best of our knowledge, the Fisher corpus is the largest English language database that includes the speaker age information to be used for the age estimation task. The division of the data for the age estimation task is shown in Table I. The division is made through stratified sampling such that there is no overlap of speakers and all age groups are represented across

TABLE III: Model Specifications

Model	Specification	
	Age	Height
NRT	Linear: (400, 1000)	
	Linear: (1000, 1000)	
	Linear: (1000, 1)	(Same as Age with input dim. of 600)
	Nonlin.: ReLU Optim.: Adam (lr 0.001)	
SVM-RT	Kernel: RBF	(Same as Age with linear kernel)
	Regul.: ℓ_1	
	Optim.: Scan (Sec. III-C)	
SVR	Kernel: RBF	Kernel: Linear
	Regul.: ℓ_1	Regul.: ℓ_1
CART	Criteria.: MSE	Criteria.: MSE
MLP	Linear: (400, 512)	Linear: (600, 2048)
	Linear: (512, 1)	Linear: (2048, 1)
	Nonlin.: ReLU	Nonlin.: ReLU
	Optim.: Adam (lr 0.01)	Optim.: Adam (lr 0.005)

the splits. Furthermore, Figure 2 shows the age distribution of the database for the three splits (train, development, test) in relation to the Table I.

For height estimation, we use the NIST speaker recognition evaluation (SRE) 2008 corpus [29]. We only obtain heights for 384 male speakers and 651 female speakers from it. Because of such data scarcity issues, we evaluate this task using cross-validation. Table II and Figure 3 show the statistics for the NIST-SRE8 database.

Since the recordings for both datasets have plenty of silences and the silences do not contribute to the information gain, Gaussian based voice activity detection (VAD) is performed on the recordings. Then, the resulting recordings are segmented to one-minute segments.

To properly represent the speech signals, we adopt one of the most effective and well-studied representations, the i-vectors [30]. I-vectors are statistical low-dimensional representations over the distributions of spectral features, and are commonly used in state-of-the-art speaker recognition systems [31] and age estimation systems [32], [33]. Respectively, 400-dimensional and 600-dimensional i-vectors are extracted for Fisher and SRE datasets using the state-of-the-art speaker identification system [34].

B. Model

The specifications for our model and the baseline models are shown in Table III. The proposed NRT is a binary tree with neural classification models as explained in Section III-B, where the neural classifiers are 3-layer ReLU neural networks. Each model is associated with a set of hyper-parameters (e.g., the λ in (7), the number of neurons and layers for the neural nets, the margin penalty and kernel bandwidth for SVM and SVR, the depth for CART, etc.) that have to be tuned

TABLE IV: Performance evaluation of neural regression tree and baselines.

Task	Dataset	Methods	Male		Female	
			MAE	RMSE	MAE	RMSE
Age	Fisher	SVR	9.22	12.03	8.75	11.35
		CART	11.73	15.22	10.75	13.97
		MLP	9.06	11.91	8.21	10.75
		SVM-RT	8.83	11.47	8.61	11.17
		NRT	7.20	9.02	6.81	8.53
Height	SRE	SVR	6.27	6.98	5.24	5.77
		CART	8.01	9.34	7.08	8.46
		MLP	8.17	10.92	7.46	9.47
		SVM-RT	5.70	7.07	4.85	6.22
		NRT	5.43	6.40	4.27	6.07

on the development set. These hyper-parameters control the complexity and generalization ability of the corresponding models. We tune them based on the bias-variance trade-off until the best performance on development set has been achieved.

C. Results

To measure the performance of our models on the age and height estimation tasks, we use the mean absolute error (MAE) and the root mean squared error (RMSE) as evaluation metrics. The results are summarized in Table IV. To reduce the effect of weights initialization on the performance of models consisting neural nets, we run those models multiple (10) times with different initialization, and report the average performance error.

For both age and height estimation, we observe that the proposed neural regression tree model generally outperforms other baselines in both MAE and RMSE, except that for height task, the neural regression tree has slightly higher RMSE than SVR, indicating higher variance. This is reasonable as our NRT does not directly optimize on the mean squared error. Bagging or forest mechanisms may be used to reduce the variance. Furthermore, with the neural classifier in NRT being replaced by an SVM classifier (SVM-RT), we obtain higher errors than NRT, demonstrating the effectiveness of the neural part of the NRT as it enables the features to refine with each partition and adapt to each node. Nevertheless, SVM-RT still yields smaller MAE and RMSE values than SVR and CART, and is on par with the MLP on the age task; on the height task, SVM-RT outperforms SVR, CART and MLP in terms of MAE values while also showing relatively small variances. This consolidates our claim that even without the use of a neural network, our model can find optimal thresholds for the discretization of response variables. On the other hand, this also confirms that using neural nets without the tree adaptation only contributes to a small portion of the performance gain provided that the neural nets generalize well. Additionally, we observe that a simple-structured MLP, as compared to the MLP component

in NRT, is required to obtain reasonable performance—a more complex-structured MLP would not generalize well to the test set and yield high estimation bias. This, in turn, implies that our NRT model can employ high-complexity neural nets to adapt the features to be more discriminative as the discretization refines, while at the same time maintain the generalization ability of the model.

To test the significance of the results, we further conduct pairwise statistical significance tests. We hypothesize that the errors achieved from our NRT method are significantly smaller than the closest competitor SVR. Paired t-test for SVR *v.s.* SVM-RT and SVM-RT *v.s.* NRT yield p-values less than 2.2×10^{-16} , indicating significant improvement. Similar results are obtained for height experiments as well. Hence, we validate the significance of the performance improvement of our NRT method on estimating ages and heights over the baseline methods.

D. Node-based Error Analysis

The hierarchical nature of our formulation allows us to analyze our model on every level and every node of the tree in terms of its classification and regression error. Figure 4 shows the per-level regression errors in terms of MAE for female and male speakers, where the nodes represent the age thresholds used as splitting criteria at each level, and the edges represent the regression errors. We notice that regression error increases from left to right for both female and male speakers (except the leftmost nodes where the behavior does not follow possibly due to data scarcity issues), meaning the regression error for the younger speakers is lower than the error for older speakers. In other words, our model is able to discriminate better between younger speakers. This is in agreement with the fact that the vocal characteristics of humans undergo noticeable changes during earlier ages, and then relatively stabilize for a certain age interval [35]. Hence, the inherent structural properties of our model not only improve the overall regression performance as

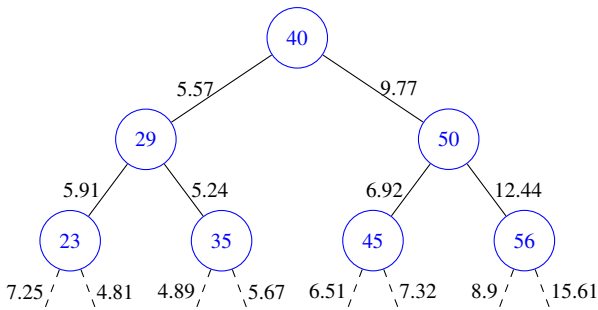
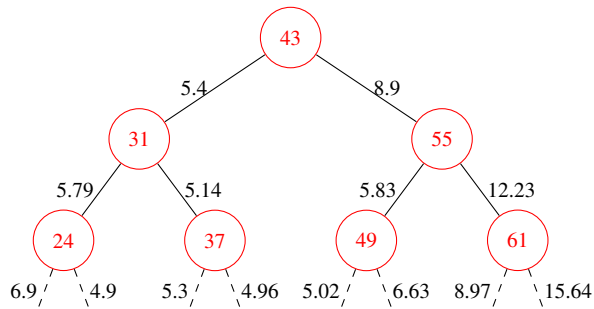


Fig. 4: Regression errors for different age groups for female (Left) and male (Right) for the task of speaker age estimation. Each node represents the age threshold used as splitting criterion, and each edge represents the regression error in terms of MAE.

we see in the previous section, but in the case of age estimation, also model the real world phenomenon.

E. Limitations

We acknowledge that our model might not be ubiquitous in its utility across all regression tasks. Our hypothesis is that it works well with tasks that can benefit from a partition based formulation. We empirically show that to be true for two such tasks above. However, in future we would like to test our model for other standard regression tasks. Furthermore, because our model formulation inherits its properties from the regression-via-classification (RvC) framework, the objective function is optimized to reduce the *classification error* rather than the *regression error*. This limits us in our ability to directly compare our model to other regression methods. In future, we intend to explore ways to directly minimize the regression error while employing the RvC framework.

V. CONCLUSIONS

In this paper, we proposed Neural Regression Trees (NRT) for optimal discretization of response variables in regression-via-classification (RvC) tasks. It targeted the two challenges in traditional RvC approaches: finding optimal discretization

thresholds, and selecting optimal set of features. We developed a discretization strategy by recursive binary partition based on the optimality of neural classifiers. Furthermore, for each partition node on the tree, it was able to locally optimize features to be more discriminative. We proposed a scan method and a gradient method to optimize the tree. The proposed NRT model outperformed baselines in age and height estimation experiments, and demonstrated significant improvements.

REFERENCES

- [1] L. Breiman, *Classification and regression trees*. Routledge, 2017.
- [2] J. R. Quinlan, "Induction of decision trees," *Machine learning*, vol. 1, no. 1, pp. 81–106, 1986.
- [3] L. Torgo and J. Gama, "Regression using classification algorithms," *Intelligent Data Analysis*, vol. 1, no. 4, pp. 275–292, 1997.
- [4] J. N. Morgan and J. A. Sonquist, "Problems in the analysis of survey data, and a proposal," *Journal of the American Statistical Association*, vol. 58, no. 302, pp. 415–434, 1963.
- [5] L. Breiman, J. H. Friedman, R. A. Olshen, and C. J. Stone, *Classification and regression trees*. Monterey, CA: Wadsworth and Brooks, 1984.
- [6] J. R. Quinlan *et al.*, "Learning with continuous classes," in *5th Australian Joint Conference on Artificial Intelligence*, vol. 92. World Scientific, 1992, pp. 343–348.
- [7] J. R. Quinlan, *C4.5: Programs for machine learning*. Elsevier, 2014.
- [8] D. Laptev and J. M. Buhmann, "Convolutional decision trees for feature learning and segmentation," in *German Conference on Pattern Recognition*. Springer, 2014, pp. 95–106.
- [9] H. Xiao, "NDT: Neural decision tree towards fully functioned neural graph," *arXiv preprint arXiv:1712.05934*, 2017.
- [10] R. Balestrero, "Neural decision trees," *arXiv preprint arXiv:1702.07360*, 2017.
- [11] R. Tanno, K. Arulkumaran, D. C. Alexander, A. Criminisi, and A. Nori, "Adaptive neural trees," *arXiv preprint arXiv:1807.06699*, 2018.
- [12] P. Kotschieder, M. Fiterau, A. Criminisi, and S. R. Buló, "Deep neural decision forests," in *2015 IEEE International Conference on Computer Vision*. IEEE, 2015, pp. 1467–1475.
- [13] W. Shen, Y. Guo, Y. Wang, K. Zhao, B. Wang, and A. Yuille, "Deep regression forests for age estimation," *arXiv preprint arXiv:1712.07195*, 2017.
- [14] S. M. Weiss and N. Indurkha, "Rule-based machine learning methods for functional prediction," *Journal of Artificial Intelligence Research*, vol. 3, pp. 383–403, 1995.
- [15] L. Torgo and J. Gama, "Regression by classification," in *Brazilian Symposium on Artificial Intelligence*. Springer, 1996, pp. 51–60.
- [16] P. McCullagh, "Regression models for ordinal data," *Journal of the Royal Statistical Society: Series B (Methodological)*, vol. 42, no. 2, pp. 109–127, 1980.
- [17] P. A. Gutiérrez, M. Perez-Ortiz, J. Sanchez-Monedero, F. Fernandez-Navarro, and C. Hervas-Martinez, "Ordinal regression methods: survey and experimental study," *IEEE Transactions on Knowledge and Data Engineering*, vol. 28, no. 1, pp. 127–146, 2016.
- [18] H. Kim, K. Bae, and H. Yoon, "Age and gender classification for a home-robot service," in *The 16th IEEE International Symposium on Robot and Human Interactive Communication*. IEEE, 2007, pp. 122–126.
- [19] F. Metzke, J. Ajmera, R. Englert, U. Bub, F. Burkhardt, J. Stegmann, C. Muller, R. Huber, B. Andrassy, J. G. Bauer *et al.*, "Comparison of four approaches to age and gender recognition for telephone applications," in *IEEE International Conference on Acoustics, Speech and Signal Processing*, vol. 4. IEEE, 2007, pp. IV–1089.
- [20] M. Li, C. Jung, and K. J. Han, "Combining five acoustic level modeling methods for automatic speaker age and gender recognition," in *Eleventh Annual Conference of the International Speech Communication Association*, 2010.

- [21] G. Dobry, R. M. Hecht, M. Avigal, and Y. Zigel, "Supervector dimension reduction for efficient speaker age estimation based on the acoustic speech signal," *IEEE Transactions on Audio, Speech, and Language Processing*, vol. 19, no. 7, pp. 1975–1985, 2011.
- [22] M. Lee and K. Kwak, "Performance comparison of gender and age group recognition for human-robot interaction," *International Journal of Advanced Computer Science and Applications*, vol. 3, no. 12, 2012.
- [23] M. H. Bahari, M. McLaren, H. Van Hamme, and D. A. Van Leeuwen, "Speaker age estimation using i-vectors," *Engineering Applications of Artificial Intelligence*, vol. 34, pp. 99–108, 2014.
- [24] B. D. Barkana and J. Zhou, "A new pitch-range based feature set for a speaker's age and gender classification," *Applied Acoustics*, vol. 98, pp. 52–61, 2015.
- [25] A. H. Poorjam, M. H. Bahari, V. Vasilakakis *et al.*, "Height estimation from speech signals using i-vectors and least-squares support vector regression," in *2015 38th International Conference on Telecommunications and Signal Processing*. IEEE, 2015, pp. 1–5.
- [26] Y. Fu and T. S. Huang, "Human age estimation with regression on discriminative aging manifold," *IEEE Transactions on Multimedia*, vol. 10, no. 4, pp. 578–584, 2008.
- [27] D. Basak, S. Pal, and D. C. Patranabis, "Support vector regression," *Neural Information Processing-Letters and Reviews*, vol. 11, no. 10, pp. 203–224, 2007.
- [28] C. Cieri, D. Miller, and K. Walker, "The Fisher corpus: a resource for the next generations of speech-to-text." in *LREC*, vol. 4, 2004, pp. 69–71.
- [29] S. S. Kajarekar, N. Scheffer, M. Graciarena, E. Shriberg, A. Stolcke, L. Ferrer, and T. Bocklet, "The SRI NIST 2008 speaker recognition evaluation system," in *IEEE International Conference on Acoustics, Speech and Signal Processing*. IEEE, 2009, pp. 4205–4208.
- [30] N. Dehak, P. J. Kenny, R. Dehak, P. Dumouchel, and P. Ouellet, "Front-end factor analysis for speaker verification," *IEEE Transactions on Audio, Speech, and Language Processing*, vol. 19, no. 4, pp. 788–798, 2011.
- [31] S. O. Sadjadi, S. Ganapathy, and J. W. Pelecanos, "Speaker age estimation on conversational telephone speech using senone posterior based i-vectors," in *IEEE International Conference on Acoustics, Speech and Signal Processing*. IEEE, 2016, pp. 5040–5044.
- [32] P. G. Shivakumar, M. Li, V. Dhandhanian, and S. S. Narayanan, "Simplified and supervised i-vector modeling for speaker age regression," in *IEEE International Conference on Acoustics, Speech and Signal Processing*, 2014, pp. 4833–4837.
- [33] J. Grzybowska and S. Kacprzak, "Speaker age classification and regression using i-vectors." in *INTERSPEECH*, 2016, pp. 1402–1406.
- [34] H. Dharmyal, T. Zhou, B. Raj, and R. Singh, "Optimizing neural network embeddings using pair-wise loss for text-independent speaker matching," in *INTERSPEECH*, 2018.
- [35] E. T. Stathopoulos, J. E. Huber, and J. E. Sussman, "Changes in acoustic characteristics of the voice across the life span: measures from individuals 4–93 years of age," *Journal of Speech, Language, and Hearing Research*, vol. 54, no. 4, pp. 1011–1021, 2011.