NEURAL REGRESSION TREES

Anonymous authors
Paper under double-blind review

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.

1 INTRODUCTION

One of the most challenging and intensively studied problems in machine learning is regression—predicting the continuous value of a dependent variable (label) y from the independent variable (covariates) x. The relationship between $y \in \mathbb{Y}$ and $x \in \mathbb{X}$, a regression function $f : \mathbb{X} \to \mathbb{Y}$, is generally unknown and may not be deterministic. Varied approaches can be used to estimate the map f, such as linear regression Bishop (2006), nonparametric regression Wasserman (2006), neural nets Friedman et al. (2001), etc. Among these approaches, one school of methods is regression-via-classification (RvC) Weiss & Indurkhya (1995), in which the continuous dependent variable y is discretized into ordered bins and an estimation $\widehat{y}$ is made by classifying the independent variable x into one of the bins. Such class of methods addresses one issue with standard regression methods: different regions of $\mathbb{Y}$ may depend on different subsets of $\mathbb{X}$, and a common statistical estimator of f on the entire $\mathbb{Y}$ could have high bias on some regions of $\mathbb{Y}$. Therefore, a separate regression function is preferable for each region of $\mathbb{Y}$.

In this study, we focus on developing the optimal discretization strategy for RvC. For the sake of a concise explanation, we first define the problem clearly. To start with, we define

Partition *A partition* Π *on a set* $\mathbb{Y}$ *is*

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

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

In our case, $\mathbb{Y} \subset \mathbb{R}$, and a partition on $\mathbb{Y}$ effectively converts the continuous-valued variable y into a categorical one C. This process is often termed as *discretization* Fayyad & Irani (1993); Dougherty et al. (1995). Based on this discretization, a RvC system includes a classifier $h : x \mapsto \{C_1, \dots, C_N\}$ and a regressor $r : C_i \mapsto \mathbb{R}$. Using this system, a prediction for y is $\widehat{y} = r(h(x))$.

The key aspect of an RvC system is its method of partition Π. It uses a set of thresholds $\mathbb{T} := \{t_0, \dots, t_N\}$ to determine $\Pi(y) = C_n$ if $t_{n-1} \leq y \leq t_n$. Problems arise as how to set these thresholds. One approach is to set $\mathbb{T}$ by prior knowledge, such as equally probable intervals, equal width intervals, k-means clustering, etc. Dougherty et al. (1995); Torgo & Gama (1996; 1997). However, this approach could lead to potential difficulties in defining the optimal partition Fayyad & Irani (1993), hence the non-ideal performance. For instance, ad-hoc partitions can result in unbalanced partitions that are too easy or too difficult to regress; partitions that are not fine enough can lead to high regression bias.

Therefore, it is necessary to develop strategies to find the optimal discretization of the dependent variable, i.e., to find the optimal thresholds. We propose a top-down tree based approach, which grows a partition tree with each threshold on the node determined by a neural binary classifier, and the value of dependent variable y is predicted at the leaves. We call the resulting model a *neural regression tree*

(NRT). Specifically, we adopt a top-down binary splitting approach to recursively grow the partition tree. To determine the splitting threshold t_n for a node n, we use a binary classifier $h_n : \mathbb{X} \to \{-1, 1\}$ to find the optimal t_n^* by minimizing the classification error $t_n^* = \arg\min_{t_n} \mathcal{E}_{h_n}(\mathbb{D}_n, t_n)$, where $\mathcal{E}_{h_n}$ is the classification error of h_n, and $\mathbb{D}_n$ is the data on node n. The number of thresholds N, which determines the number of partitions and hence the depth of the tree, stops increasing when the classification performance saturates. Next, in order to determine the value y for any feature x we only need to find out which leave partition x belongs to following the partition tree, and estimate $\widehat{y}$ using a regression function on that bin.

The benefits of doing so are three-fold: (1) the discretization thresholds $\mathbb{T}$ are explicitly optimized to obtain optimal partition; (2) By adapting a top-down binary splitting approach, we reduce the computational complexity, since directly optimizing $\mathbb{T}$ is a hard problem as it scales exponentially with N; (3) the features are adapted to each node by the neural classifier, and hence the model promotes hierarchical representation learning, increasing model interpretability.

To achieve the above, we propose two algorithms, a scanning search algorithm, and a gradient based method, against a loss function for which we call the *triviality loss* to solve for the discretization thresholds efficiently. To demonstrate the utility of the proposed approach we conduct experiments on a pair of challenging regression tasks: estimating the age and height of speakers from their voices. Experiments show that our model achieves significant performance gain over state-of-the-art baseline models.

2 Related Work

Regression Trees Tree-structured models have been around for a long time. Among them, the most closely related are the regression trees. Following our terminology, the regression tree can be described as $\widehat{y} = r(\Pi(x))$, where the partition is performed on independent variable instead of dependent variable. The first regression tree algorithm was presented by Morgan & Sonquist, where they propose a greedy approach to fit a piecewise constant function by recursively splitting the data into two subsets based on partition on $\mathbb{X}$. The optimal split is a result of minimizing the impurity which defines the homogeneity of the split. This algorithm set the basis for a whole line of research on classification and regression trees. Refined algorithms include CART Breiman et al. (1984), ID3 Quinlan (1986), m5 Quinlan et al. (1992), and C4.5 Quinlan (2014). Recent work combines the tree-structure and neural nets to gain the power of both structure learning and representation learning. Such work include the convolutional decision trees Laptev & Buhmann (2014), neural decision trees Xiao (2017); Balestriero (2017), adaptive neural trees Tanno et al. (2018), deep neural decision forests Kontschieder et al. (2015), and deep regression forests Shen et al. (2017).

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 dependent variables, enabling the features to adapt to the partitions of dependent variables.

Regression via Classification (RvC) The idea for RvC was presented by Weiss & Indurkhya. Their algorithm was based on k-means clustering to categorize numerical variables. Other conventional approaches Torgo & Gama (1996; 1997) 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.

3 Neural regression tree

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

3.1 Model Formulation

Formally, following the description of RvC in Section 1, a classification rule classifies the data into disjoint bins $h_\theta : x \mapsto \{C_0, \cdots, C_N\}$, where $C_i = \Pi(y)$ corresponds to $t_{i-1} \leq y \leq t_i$, and θ

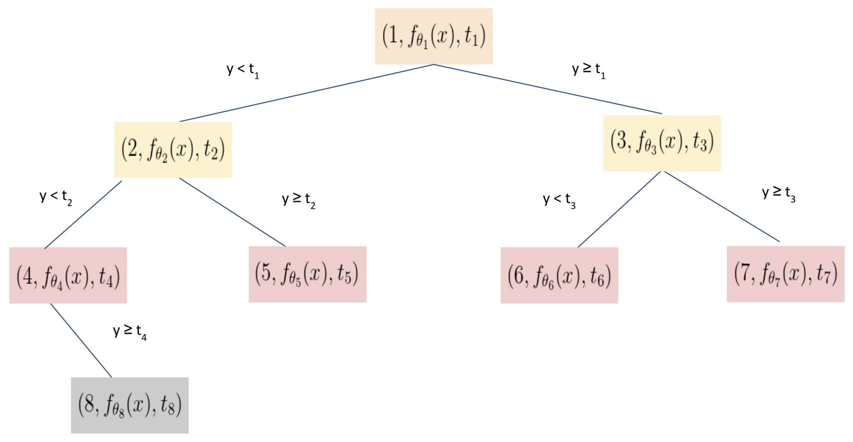

$(1, f_{\theta_1}(x), t_1)$
y < t₁
y ≥ t₁
$(2, f_{\theta_2}(x), t_2)$
$(3, f_{\theta_3}(x), t_3)$
y < t₂
y ≥ t₂
y < t₃
y ≥ t₃
$(4, f_{\theta_4}(x), t_4)$
$(5, f_{\theta_5}(x), t_5)$
$(6, f_{\theta_6}(x), t_6)$
$(7, f_{\theta_7}(x), t_7)$
y ≥ t₄
$(8, f_{\theta_8}(x), t_8)$

Figure 1: Illustration of neural regression tree.

parameterizes the classifier. Then, a regression rule predicts the value of the dependent variable

$$\widehat{y}(x) = r(h_\theta(x)), \tag{1}$$

where $r : C_i \mapsto [t_{i-1}, t_i]$ is any regression function that operates locally on all instances that are assigned to the bin C_i.

Alternatively, the classification rule may compute the probability of a data point x being classified into bin C_i, $P(C_i \mid x) = h_\theta(x)$, and the regression rule is given by

$$\widehat{y}(x) = \mathbb{E}_{C_i}[r(h_\theta(x))] = \sum_{i=0}^{N} P(C_i \mid x) r(C_i). \tag{2}$$

Defining an error $\mathcal{E}(y, \widehat{y}(x))$ between the true y and the estimated $\widehat{y}(x)$, our objective is to learn the thresholds $\{t_0, \ldots, t_N\}$ and the parameters $\{\theta_1, \ldots, \theta_N\}$ such that the expected error is minimized

$$\{t_n^*\}, \{\theta_n^*\} \longleftarrow \arg\min_{t,\theta} \mathbb{E}_x \left[\mathcal{E}(y, \widehat{y}(x))\right]. \tag{3}$$

Note that the number of thresholds N too is a variable that may either be manually set or explicitly optimized. In practice, instead of minimizing the *expected* error, we will minimize the *empirical* average error avg$(\mathcal{E}(y_i, \widehat{y}(x_i)))$ 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 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 dependent 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 $l_0, \cdots, l_p$ represent the chain of nodes from root to leaf, where l_0 is the root and $l_p = l$ is the leaf itself. The *a posteriori* probability of the leaf is given by $P(l \mid x) = \prod_{r=1}^{p} P(l_r \mid l_{r-1}, x)$, where each $P(l_r \mid l_{r-1}, x)$ is given by the neural classifier on node l_{r-1}. Substitution into (2) yields the final regressed value of the dependent variable

$$\widehat{y}(x) = \sum_{l \in \text{leaves}} P(l \mid x) r(l), \tag{4}$$

where $r(l)$, 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 regression function, etc.

3.2 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 $\mathcal{E}(\mathbb{D}_n; t_n, \theta_n) = \text{avg}\,(\mathcal{E}(y, \widehat{y}_n(x)))$ between the true response values y and the estimated response $\widehat{y}_n(x)$ 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) must be assigned to child n' or n''. Let $\mathcal{E}(y(t_n), h_{\theta_n}(x))$ be a quantifier of the classification error (which is the binary cross entropy loss in our case) for any instance (x, y). We define the classification loss at node n as

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

The classification loss E_{θ_n, t_n} 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 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 Breiman et al. (1984) may also apply though)

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

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

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

where $\lambda \in (0, 1)$ is used to assign the relative importance of the two components of the loss.

In the optimization of (7), the loss function depends on t_n through $y(t_n)$, which is a discrete function of t_n. Hence, we have two possible ways of optimizing (7). 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 discrete function $\text{sign}(y - t_n)$ is replaced by a smooth, differentiable relaxation: $y(t_n) = 0.5(\tanh(\beta(y - t_n)) + 1)$, where β controls the steepness of the function and must typically be set to a large value ($\beta = 10$ in our settings). The triviality penalty is also redefined (to be differentiable w.r.t. t_n) as the proximity to the median $\mathcal{T}(t_n) = \|t_n - \text{median}(y \mid (x, y) \in \mathbb{D}_n)\|_2$, 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 is recursed down the tree. Algorithm 1 describes the entire training algorithm. The growth of the tree may be continued until the regression performance on a held-out set saturates.

```
Input: $\mathbb{D}$
Parameter: $\{t_n\}, \{\theta_n\}$
Output: $\{t_n^*\}, \{\theta_n^*\}$
Initialize $\mathbf{t} = \{\}$, $\boldsymbol{\theta} = \{\}$, $\mathbb{D}_1 \leftarrow \mathbb{D}$;
Function BuildTree($\mathbb{D}_n$)
    Initialize $t_n, \theta_n$;
    $t_n^*, \theta_n^* \leftarrow$ NeuralClassifier($\mathbb{D}_n, t_n, \theta_n$);
    $\mathbf{t} \leftarrow t_n^*$, $\boldsymbol{\theta} \leftarrow \theta_n^*$;
    $\mathbb{D}_{n'}, \mathbb{D}_{n''} \leftarrow$ 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($\mathbb{D}_1$);
```

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

4 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 Kim et al. (2007); Metze et al. (2007); Li et al. (2010); Dobry et al. (2011); Lee & Kwak (2012); Bahari et al. (2014); Barkana & Zhou (2015); Poorjam et al. (2015); Fu & Huang (2008).

We compare our model with (1) a regression baseline using the support vector regression (SVR) Basak et al. (2007), and (2) a regression tree baseline using classification and regression tree (CART) Breiman et al. (1984). 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 (3) regression tree with the support vector machine (SVM-RT).

4.1 Data

To promote a fair comparison, we select two well-established public datasets in the speech community. For age estimation, we use the Fisher English corpus Cieri et al. (2004). 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 1. The division is made such that there is no overlap of speakers and all age groups are represented across the splits. Furthermore, Figure 2 shows the age distribution of the database for the three splits (train, development, and test) in relation to the Table 1.

Table 1: 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

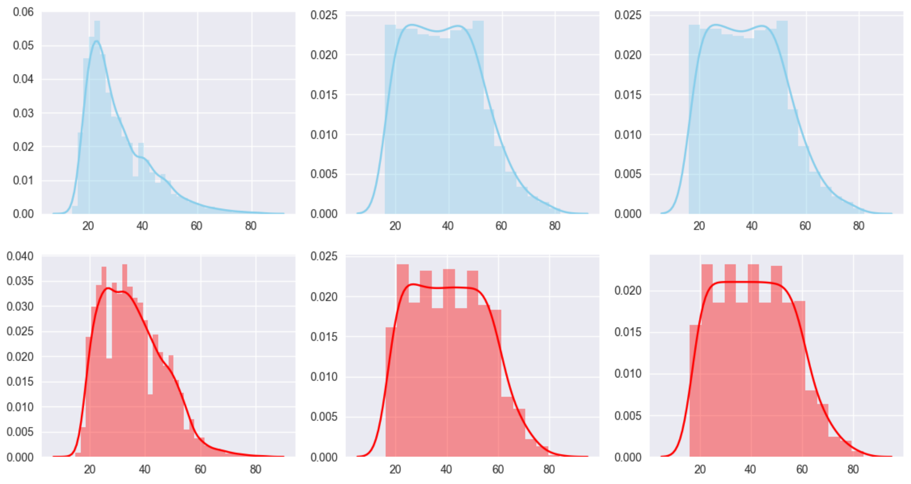


Figure 2: Age Distribution (in percentages) for male *(Top)* and female *(Bottom)* speakers for the fisher database for train *(Left)*, development *(Center)* and test *(Right)* set.

For height estimation, we use the NIST speaker recognition evaluation (SRE) 2008 corpus Kajarekar et al. (2009). We only have heights for 384 male speakers and 651 female speakers from it. Because of data scarcity issues, we evaluate this task using cross-validation. Table 2 and Figure 3 show the statistics for the NIST-SRE8 database.

Table 2: NIST-SRE8 Dataset Stats

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

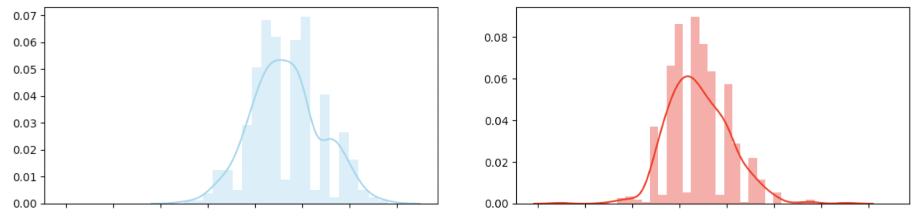


Figure 3: Age Distribution (in percentages) for male *(Left)* and female *(Right)* speakers for the NIST-SRE8 database.

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 1-minute segments.

To properly represent the speech signals, we adopt one of the most effective and well-studied representations, the i-vectors Dehak et al. (2011). 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 Sadjadi et al. (2016) and age estimation systems Shivakumar et al. (2014); Grzybowska & Kacprzak (2016). Respectively, 400-dimensional and 600-dimensional i-vectors are extracted for Fisher and SRE datasets using the state-of-the-art speaker identification system Dhamyal et al. (2018)

4.2 MODEL

The proposed neural regression tree is a binary tree with neural classification models as discussed in Section 3.1. The specifications for our model and the baseline models are shown in Table 3. The NRT is composed of a collection of 3-layer ReLU neural networks. The kernels, regularizations and parameters for SVRs and SVMs are obtained from experimenting on development set.

Table 3: Model Specifications

Model	**Specification**	
	Age	Height
NRT	Linear: (400, 1000) Linear: (1000, 1000) Linear: (1000, 1) Nonlin.: ReLU Optim.: Adam (lr 0.001)	(Same as Age with input dim. of 600)
SVM-RT	Kernel: RBF Regul.: ℓ_1 Optim.: Scan (Sec. 3.2)	(Same as Age with linear kernel)
SVR	Kernel: RBF Regul.: ℓ_1	Kernel: Linear Regul.: ℓ_1
CART	Criteri.: MSE	Criteri.: MSE

4.3 RESULTS

As a measure of the performance of our models and the baseline models on age and height estimation, we use the mean absolute error (MAE) and the root mean squared error (RMSE). The results are summarized in Table 4.

Table 4: Performance evaluation of neural regression tree and baselines.

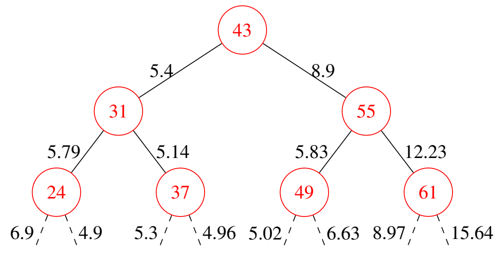
Task	Dataset	**Methods**	**Male**		**Female**	
			MAE	RMSE	MAE	RMSE
Age	Fisher	SVR	9.22	12.03	8.75	11.35
		CART	11.73	15.22	10.75	13.97
		SVM-RT	8.83	11.47	8.61	11.17
		NRT	**7.20**	**9.02**	**6.81**	**8.53**
Height	SRE	SVR	6.27	6.98	5.24	**5.77**
		CART	8.01	9.34	7.08	8.46
		SVM-RT	5.70	7.07	4.85	6.22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

For both age and height estimation, we see that the proposed neural regression tree 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 square error. Bagging or forest mechanisms may be used to reduce the variance. Furthermore, with the neural classifier in NRT being replaced by a SVM classifier (SVM-RT), we obtain higher error than NRT, implying 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 values than SVR and CART, hence strengthening our hypothesis that the our model can find optimal discretization thresholds for optimal dependent variable discretization even without the use of a neural network.

To test the significance of the results, we further conduct paired-wise 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 strong significance of the improvement. Similar results are obtained for height experiments as well. Hence we validate the significant performance improvement of our NRT method on estimating ages and heights over the baseline methods.

4.4 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. In our evaluation for the speaker age estimation task, we noticed that the regression error for the younger speakers was lower than the error in the case of older speakers. In other words, our model was 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 Stathopoulos et al. (2011). Figure 4 shows the per-level MAE for female and male speakers. The nodes represent the age thresholds used as splitting criteria at each level. The edges represent the regression error. The inherent structural properties of our model not only improve the overall regression performance as we saw in the previous section, but in the case of age estimation, also model the real world phenomenon. This is visible in 4 where the regression error can be seen to increase from left to right for both female and male speakers (except the left most nodes where the behavior does not follow possibly due to data scarcity issues).



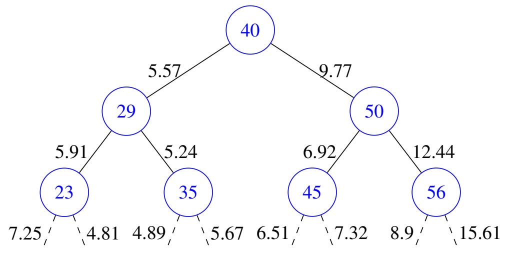


Figure 4: Regression error (MAE) for different age groups for female *(Left)* and male *(Right)* for the task of speaker age estimation.

4.5 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 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.

5 Conclusions

In this paper, we proposed a neural regression tree for optimal discretization of dependent variables in regression-by-classification tasks. It targeted the two difficulties in regression-by-classification: 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. In addition, we proposed a scan method and a gradient method to optimize the tree.

The proposed neural regression tree outperformed baselines in age and height estimation experiments, and demonstrated significant improvements.

REFERENCES

M. H. Bahari, M. McLaren, H. Van Hamme, and D. A. Van Leeuwen. Speaker age estimation using i-vectors. *Engineering Applications of Artificial Intelligence*, 34:99–108, 2014.

R. Balestriero. Neural decision trees. *arXiv preprint arXiv:1702.07360*, 2017.

B. D. Barkana and J. Zhou. A new pitch-range based feature set for a speaker's age and gender classification. *Applied Acoustics*, 98:52–61, 2015.

D. Basak, S. Pal, and D. C. Patranabis. Support vector regression. *Neural Information Processing-Letters and Reviews*, 11(10):203–224, 2007.

C. M. Bishop. *Pattern Recognition and Machine Learning*. Springer-Verlag, Berlin, Heidelberg, 2006.

L. Breiman, J. H. Friedman, R. A. Olshen, and C. J. Stone. *Classification and regression trees*. Wadsworth and Brooks, Monterey, CA, 1984.

C. Cieri, D. Miller, and K. Walker. The fisher corpus: a resource for the next generations of speech-to-text. In *LREC*, volume 4, pp. 69–71, 2004.

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*, 19(4):788–798, 2011.

H. Dhamyal, T. Zhou, B. Raj, and R. Singh. Optimizing neural network embeddings using pair-wise loss for text-independent speaker matching. In *INTERSPEECH*, 2018.

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*, 19(7):1975–1985, 2011.

J. Dougherty, R. Kohavi, and M. Sahami. Supervised and unsupervised discretization of continuous features. In *Machine Learning Proceedings*, pp. 194–202. Elsevier, 1995.

U. Fayyad and K. Irani. Multi-interval discretization of continuous-valued attributes for classification learning. 1993.

J. Friedman, T. Hastie, and R. Tibshirani. *The elements of statistical learning*, volume 1. Springer series in statistics New York, NY, USA:, 2001.

Y. Fu and T. S. Huang. Human age estimation with regression on discriminative aging manifold. *IEEE Transactions on Multimedia*, 10(4):578–584, 2008.

J. Grzybowska and S. Kacprzak. Speaker age classification and regression using i-vectors. In *INTERSPEECH*, pp. 1402–1406, 2016.

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*, pp. 4205–4208. IEEE, 2009.

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*, pp. 122–126. IEEE, 2007.

P. Kontschieder, M. Fiterau, A. Criminisi, and S. R. Bulo. Deep neural decision forests. In *2015 IEEE International Conference on Computer Vision*, pp. 1467–1475. IEEE, 2015.

D. Laptev and J. M. Buhmann. Convolutional decision trees for feature learning and segmentation. In *German Conference on Pattern Recognition*, pp. 95–106. Springer, 2014.

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*, 3(12), 2012.

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.

F. Metze, 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*, volume 4, pp. IV–1089. IEEE, 2007.

J. N. Morgan and J. A. Sonquist. Problems in the analysis of survey data, and a proposal. *Journal of the American Statistical Association*, 58(302):415–434, 1963.

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*, pp. 1–5. IEEE, 2015.

J. R. Quinlan. Induction of decision trees. *Machine learning*, 1(1):81–106, 1986.

J. R. Quinlan. *C4. 5: programs for machine learning*. Elsevier, 2014.

J. R. Quinlan et al. Learning with continuous classes. In *5th Australian joint conference on artificial intelligence*, volume 92, pp. 343–348. World Scientific, 1992.

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*, pp. 5040–5044. IEEE, 2016.

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.

P. G. Shivakumar, M. Li, V. Dhandhania, and S. S. Narayanan. Simplified and supervised i-vector modeling for speaker age regression. In *IEEE International Conference on Acoustics, Speech and Signal Processing*, pp. 4833–4837, 2014.

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*, 54(4):1011–1021, 2011.

R. Tanno, K. Arulkumaran, D. C. Alexander, A. Criminisi, and A. Nori. Adaptive neural trees. *arXiv preprint arXiv:1807.06699*, 2018.

L. Torgo and J. Gama. Regression by classification. In *Brazilian symposium on artificial intelligence*, pp. 51–60. Springer, 1996.

L. Torgo and J. Gama. Regression using classification algorithms. *Intelligent Data Analysis*, 1(4): 275–292, 1997.

L. Wasserman. *All of Nonparametric Statistics*. Springer-Verlag, Berlin, Heidelberg, 2006.

S. M. Weiss and N. Indurkhya. Rule-based machine learning methods for functional prediction. *Journal of Artificial Intelligence Research*, 3:383–403, 1995.

H. Xiao. Ndt: Neual decision tree towards fully functioned neural graph. *arXiv preprint arXiv:1712.05934*, 2017.