Progressively Balanced Multi-class Neural Trees

Ameya Godbole Spoorthy Bhat Prithwijit Guha
Department of Electronics and Electrical Engineering
Indian Institute of Technology Guwahati
Email: {godbole,spoorthy,pguha}@iitg.ernet.in

***Abstract*—Decision trees are discriminative classifiers that hierarchically partition the input space to achieve regions containing instances having uniform class label. Existing works in this area have mostly focused on C4.5 trees that learn axis aligned partitions. On the other hand, neural trees learn oblique partitions from data and use lesser number of decision nodes hosting perceptrons. However, these perceptrons are susceptible to data imbalances. This motivated us to propose a progressively balanced neural tree where training dataset are balanced prior to perceptron learning. The second contribution is the optimization of the decision function with respect to entropy impurity based objective functions. This formulation also allows a parent node to have more than two child nodes. The proposed algorithm is benchmarked on ten standard datasets against three baseline multi-class classification algorithms.**

I. Introduction

Classification problems involving multiple classes aim at approximating the complex decision function $\mathcal{C} : \mathcal{R}^d \rightarrow \mathcal{Y}$ such that an input $\mathbf{x} \in \mathcal{R}^d$ is assigned a class label from $\mathcal{Y} = \{1, \dots c\}$. Decision trees are discriminative classifiers that decompose such complex decision functions into a number of simple rules by partitioning the input space using splitting nodes (or decision nodes) [1]. This is achieved by a sequential top-down process that hierarchically partitions the input space using (simple) decision functions learned at splitting nodes. The decision function parameters are learned with the objective of reducing class label based "impurities" in partitions. The tree terminates at partitions where majority instances have similar class label. Such partitions correspond to "pure" label nodes or leaf nodes of decision trees.

Existing research and applications of decision trees have mostly focused on C4.5 algorithm [2], [3]. This architecture sequentially learns only two optimal attributes (j, θ) at each splitting node leading to axis aligned partitions ($\mathbf{x}[j] \geq \theta$) of input space [4]. This algorithm is known for its simplicity, ease of learning and fast operation [5]. The C4.5 trees are also used as components of random forests for many classification applications [6]. However, these trees have the inherent drawback of creating large number of decision nodes for approximating simple oblique splits.

Neural tree architectures host neural networks at splitting nodes [7]. Such decision functions range from perceptrons (perceptron trees [8]) to MLP (MLP trees [7]) and SVM (SVM trees [9]). However, MLP and SVM are complex decision functions. Thus, perceptrons are mostly used at splitting nodes for reducing overall structural complexity of tree [10]. Perceptron based decision functions are capable of learning oblique splits. Hence, neural trees form lesser number of decision nodes compared to C4.5 trees. However, (discriminative) decision functions of neural trees are prone to data imbalances and are harder to learn compared to C4.5.

Research in neural trees have focused on various issues related to number of classes and child nodes [11], decision functions [12], [7], [9] and associated impurity measures [13], [14], data imbalances [15], tree structure [16], global vs. local optimization [16], [17] etc.

Existing works in neural trees have taken the number of child nodes as number of classes present in the data at any node, and trained discriminative classifiers. These approaches have problems in scaling up for applications with large number of classes. Consistent usage of more than two child nodes causes fast reduction of data in successive splits leading to insignificant partitions. Thus, researchers have proposed the use of two child nodes for splitting nodes even in multi-class applications [14]. However, for large datasets it is better to have multi-way splits at early levels to generate relatively smaller datasets for later nodes. These nodes may have two-way splits for avoiding data thinning. For example, the number of child nodes are decided from data in [11]. Nodes at lower depth should be restricted to two-way splits as they receive smaller datasets. This calls for decision functions capable of

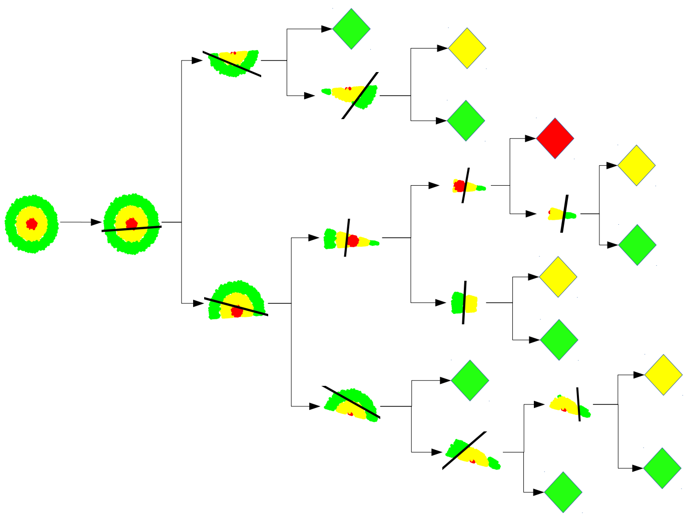
Fig. 1. Illustrating the multi-class neural tree constructed on a toy dataset of concentric ring-disc structure consisting three classes. Decision boundaries (derived from perceptron parameters) are shown along with the dataset splits. The label (leaf) nodes are shown with (class label) colored diamond shapes.

providing two or more splits.

The decision functions at splitting nodes are learned by minimizing input data impurity. Different impurity measures like information gain [13], Gini index [14], [13] and variance reduction [13] are proposed in literature. Gini index impurity measures [14], [13] are mostly used for neural trees. Researchers have also worked on balancing neural tree structures [16] and node pruning [7]. Sequential top-down methods of constructing trees are often identified as local optimization based process. Global optimization techniques are also explored for tuning decision parameters of all nodes for a given tree structure [13].

This work proposes a progressively balanced neural tree with single layer perceptron based decision nodes. Most decision trees use a two-child node architecture for splitting nodes [14]. We believe that for large datasets, it may be beneficial to have more than two splits initially. At a later stage, a two-way split might be followed to avoid data density dilution in certain branches. This necessity calls for a flexible decision node architecture capable of accommodating two or more splits. This is achieved by a single layer perceptron with soft-max activation functions [16]. Soft-max outputs enable soft routing to determine contribution of each training instance to the entropy impurity of child nodes. Entropy impurity of child nodes is minimized with respect to decision function parameters. However, such discriminative decision functions are naturally sensitive to data imbalance [18][15]. Also, majority class data along with most instances from other classes will be routed to same child by a biased decision function. Thus, impurity in data will not reduce in successive depths of the tree. Hence, input data to each splitting node are balanced prior to decision function learning. To summarize, our proposal has the following contributions.

1) Splitting nodes host single layer perceptrons with soft-max activation functions. This provides flexibility in accommodating two or more child nodes.
2) Input data to each splitting node are balanced for learning unbiased decision functions.
3) Entropy impurity based loss function is minimized to learn perceptron parameters.

This paper is organized in the following manner. The proposed approach for constructing neural tree, entropy impurity based loss function and tree termination criteria are described in Section II. Our proposal is benchmarked on 10 datasets (from UCI machine learning repository) against three existing classification algorithms. The results of our experiments are presented in Section III. Finally, we conclude in Section IV and sketch the future extensions of the present proposal.

II. Neural Tree: Proposed Approach

Neural trees are hybrid structures that host neural networks in splitting nodes and (majority) class labels at leaf nodes. Decision functions are learned at splitting nodes by analyzing input data with a greedy goal of minimizing class impurity. Previous works have experimented with MLPs [7] and SVMs [9] at decision nodes. However, most neural trees use perceptrons [16] at decision nodes to reduce overall complexity. Existing works in neural trees have mostly focused on binary classification [15]. Comparatively there are fewer works primarily focusing on multi-category classification using neural trees [7], [16]. This work proposes neural tree construction for multi-class classification problems with flexibility of having multiple children at decision nodes. The decision functions are learned by optimizing parameters of multi-output single layer perceptrons with respect to entropy impurity loss function (Sub-section II-A). Prior to optimization, input data at each node are duly balanced for avoiding biases in decision boundaries (Sub-section II-B). The balanced data consists of both original and synthetic (re-sampled) instances. Note that the balanced data set is used to learn decision node perceptron only. The original instances are evaluated using the learned perceptron and routed towards the child node corresponding to maximum perceptron output. The synthetic instances are not propagated further. This process is repeated by sequentially forming splitting nodes till a set of termination conditions are satisfied (Sub-section II-C). The functional block diagram of the proposed neural tree is illustrated in Figure 2.

A. Designing Decision Nodes

A decision (splitting) node $\mathbf{N}$ routes an input data $\mathbf{x}$ ($\mathbf{x} \in \mathcal{R}^d$) towards one of its m ($m \geq 2$) child nodes. Consider the class labeled dataset $\mathcal{D} = \{(\mathbf{x}_i, y_i); i = 1, \dots \mathbf{N}_{pts}\}$ ($\mathbf{x}_i \in \mathcal{R}^d, y_i = 1, \dots c$) whose elements are routed to m different child nodes of $\mathcal{N}$. This is achieved by a single layer perceptron with soft-max activation functions at its m outputs. The single layer perceptron operates on input $\mathbf{x}_i$ to generate the following.

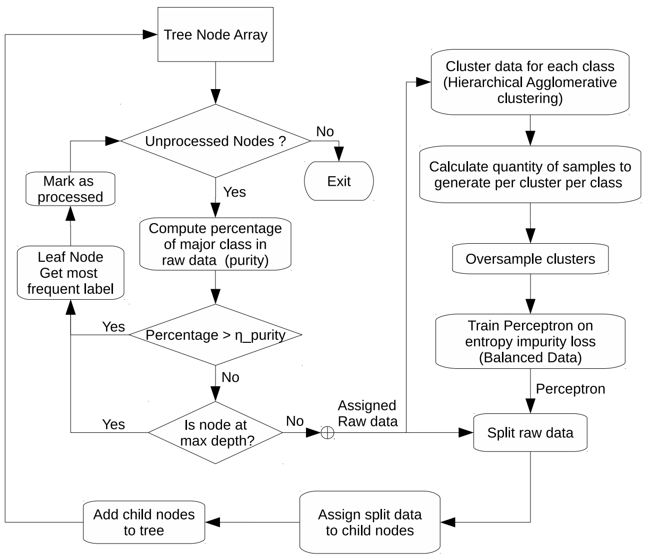
Tree Node Array
Unprocessed Nodes ?
No
Exit
Yes
Mark as processed
Compute percentage of major class in raw data (purity)
Leaf Node Get most frequent label
Yes
Percentage > η_purity
No
Yes
Is node at max depth?
No
Assigned Raw data
Cluster data for each class (Hierarchical Agglomerative clustering)
Calculate quantity of samples to generate per cluster per class
Oversample clusters
Train Perceptron on entropy impurity loss (Balanced Data)
Perceptron
Split raw data
Assign split data to child nodes
Add child nodes to tree

Fig. 2. Functional block diagram of the proposed Progressively Balanced Multi-class Neural Tree. Input data to a decision node is balanced for training the single layer perceptron. Original instances are routed to a splitting node child according to learned perceptron response. The tree terminates based on either class label purity, entropy impurity drop or data insufficiency.

$$u_{ji} = \mathbf{w}_j^T \mathbf{x}_i + b_j \tag{1}$$

$$q_{ji} = \frac{e^{u_{ji}}}{\sum_{l=1}^{m} e^{u_{li}}} \tag{2}$$

We treat q_{ji} as probability of x_i being routed to j^{th} child. Thus, the expected number of data points belonging to r^{th} class and routed to j^{th} child is given by

$$n_{jr} = \sum_{i=1}^{\mathbf{N}_{pts}} q_{ji} \delta[r - y_i] \tag{3}$$

where, $\delta[\cdot]$ is the Kronecker Delta function. Hence, expected number of data points (coming from all classes) routed to j^{th} child is $n_j = \sum_{r=1}^{c} n_{jr}$. Thus, the entropy impurity of the j^{th} child is given by

$$H_j = -\sum_{r=1}^{c} \left(\frac{n_{jr}}{n_j}\right) ln \left(\frac{n_{jr}}{n_j}\right) \tag{4}$$

.

Thus, the weighted net impurity arising out of all child nodes of $\mathbf{N}$ is given by

$$H = \sum_{j=1}^{m} \left(\frac{n_j}{\mathbf{N}_{pts}}\right) H_j \tag{5}$$

The net entropy impurity H is a function of decision node parameters $\{(\mathbf{w}_j, b_j); j = 1, \dots m\}$. Minimization of H will ensure maximization of impurity drop $\Delta H(\mathbf{N})$ from $\mathbf{N}$ to its child nodes. Entropy impurity H is minimized with respect to weight and bias parameters to achieve optimal decision boundaries. This optimization is performed using Adam Optimizer [19] used by TensorFlow package [20].

B. Data Balancing

Performance of discriminative decision functions (perceptrons in our case) degrade with imbalances in training datasets. Widely varying training data set sizes of different categories cause inter-class imbalances. Samples with rare occurrence or frequent repetitions lead to intra-class imbalance. Such data imbalances are generally handled by algorithmic modifications or dataset re-sampling. In context of decision trees, either impurity measure [21], [22] is modified or the cost of minority class misclassification is increased [23]. These techniques have been successful in handling inter-class imbalances [21], [22] but, often failed in handling intra-class ones [24], [25], [26], [27]. Re-sampling techniques either over-sample or under-sample training data [28], [29], [22], [30] and can possibly remove both inter-class and intra-class imbalances [25]. Over-sampling adds synthetic data to training set leading to large training sets and possible over-fitting to outliers for minority classes. Under-sampling causes information loss of (near) majority classes thereby degrading performance [22]. Decision nodes divide data with the objective of minimizing impurities. Thus, balanced training data in one decision node might lead to imbalanced datasets in immediate child nodes [29]. Hence, we propose to “progressively balance” datasets for training perceptrons at all decision nodes. This proposal uses cluster based oversampling (CBO, henceforth) [25] for multi-class datasets. The following steps summarize the methodology adopted for data balancing.

1) Subject instances of all classes to hierarchical agglomerative clustering.
2) Identify majority (maximum instances) class c_{maj} in dataset $\mathcal{D}$ containing c classes. Record number of clusters (k_{maj}) in majority class. Count number of instances (n_{maj}) in largest majority class cluster.
3) Inflate all majority class clusters to size n_{maj}. Note that the over-sampled majority class contains $n_{maj} \times k_{maj}$ instances.
4) Record number of clusters k_{nomaj} in any non-majority class c_{nomaj}. Oversample and inflate each cluster of class c_{nomaj} to contain $\frac{n_{maj}}{k_{nomaj}}$ instances.

Clusters are oversampled by generating synthetic points as linear combinations of cluster mean and original data points in cluster. An original data point $\mathbf{x}_{orig}$ from cluster S is chosen randomly. The synthetic point is generated as $\mathbf{x}_{synth} = (1 - \alpha)\mathbf{x}_{orig} + \alpha\mu_S$ ($\alpha \in (0, 1)$) where, μ_S is the cluster mean. This process is repeated to inflate cluster S by oversampling.

The balanced dataset contain both original and synthetic instances. These are used to train decision making perceptrons only and are not propagated further down the tree. Only original instances are evaluated using the learned perceptron. Each instance is routed towards the child node corresponding to maximum perceptron output. Progressive data balancing ensures that over-fitting to (near) majority classes is avoided. Also, lower depth and lesser nodes (simpler tree structure) are obtained as oversampled data are not propagated further [21], [22], [25], [31].

C. Tree Termination Criteria

Sequential top-down partitioning of input space is terminated at regions containing instances with uniform class labels. This class purity criteria is generally used in terminating neural trees at leaf nodes. However, terminal partitions (corresponding to leaf nodes) containing very few instances often lead to over-fitting. Otherwise, a tree is generally terminated using threshold on either node purity or maximum tree depth. Consider a node $\mathbf{N}$ at depth $\mathbf{N}_{depth}$ containing $\mathbf{N}_{pts}$ instances with $\mathbf{N}_{maj}$ points belonging to the majority class. Node $\mathbf{N}$ is declared a leaf node based on either of the following conditions.

1) Input data to $\mathbf{N}$ are lesser than a threshold n_{pts}, i.e. $\mathbf{N}_{pts} < n_{pts}$.
2) Fraction of majority points are greater than a purity threshold η_{purity}, i.e. $\frac{\mathbf{N}_{maj}}{\mathbf{N}_{pts}} > \eta_{purity}$
3) Entropy impurity drop $\Delta H(\mathbf{N})$ from $\mathbf{N}$ to its child nodes is lesser than a threshold η_H, i.e. $\mathbf{\Delta H}(\mathbf{N}) < \eta_H$

4) Present Node is at the maximum allowable tree depth $\mathbf{T}_{depth}$, i.e. $\mathbf{N}_{depth} = \mathbf{T}_{depth}$

III. Experiments and Results

The proposed Progressively Balanced Multi-class Neural Tree is benchmarked on ten standard multi-category datasets from UCI [32] machine learning repositories. These datasets are – CONNECT-4 (Detection of win/loss/ties from game status); DNA (Recognizing boundaries between exons and introns in DNA sequence); LETTERDATA (Identification of 26 English Capital Letters), PENDIGITS (Hand written digit recognition), PROTEIN (Secondary structure prediction from Amino acid sequence), SATIMAGE (Classification using multi-spectral features of satellite images), SENSIT VEHICLE, SENSORLESS (motor condition monitoring), SHUTTLE (shuttle attribute classification) and VOWEL (vowel recognition). These multi-category datasets are collected from different domains. They vary in number of features (data dimension), training instances and majority-to-minority class ratios (data imbalance). Details of these datasets are presented in Table I.

Our proposal is compared against three standard multi-class classifiers viz. C4.5 decision trees (C4.5DTree) [1], Support Vector Machines (SVM) with RBF kernel [33] and Multi-layer Perceptron (MLP). The performance of different classifiers on datasets are reported in terms of their average accuracies and classifier structure parameters.

All datasets are initially normalized (along each dimension) to $[0, 1]$ for training. The C4.5 decision trees [1] are trained by minimizing entropy based impurities at splitting nodes. The C4.5 decision trees are terminated using criteria similar to that of neural trees (Sub-section II-C) with parameters set as $n_{pts} = 20$, $\eta_{purity} = 0.05$ and $\mathbf{T}_{depth} = 20$. The SVMs are trained with trade-off parameter set to $C = 1.0$. The RBF kernel bandwidth parameter γ is set to the inverse of the number of features. For example, there are 10 features in VOWEL dataset and $\gamma_{\text{VOWEL}} = 0.1$. The number of support vectors in trained SVMs and average accuracies on test dataset are presented in Table I. For each dataset, multi-layer percpetrons (MLP) are trained with $2 * (d + m)$ nodes in hidden layer. Here, d refers to input data dimension and m is the number of classes. The size of the hidden layer was selected to allow non-linear transformation of input to a higher dimension in a dataset dependent manner. Softmax and RelUs are respectively used as activation functions in output and hidden layers. The average accuracies on test datasets are presented in Table I.

Experiments on neural trees are performed with and without progressive data balancing. Performance of neural trees are evaluated by varying the number of splitting node children in the range of 2–5 and are reported in Table II. As the number of child nodes is a hyper-parameter, we have reported the results of the best performing neural tree while comparing with baseline classifiers in Table I . For all neural trees, we have empirically set the parameters for termination criteria as $n_{pts} = 20$, $\eta_H = 0.05$, $\eta_{purity} = 0.95$ and $\mathbf{T}_{depth} = 20$.

The construction of neural trees is illustrated on two toy datasets. First one has concentric ring-disc structure with three classes (Figure 1). And, the second one has checker board like distribution with three categories (Figure 3).

A. Discussions

The proposed progressively balanced neural tree is observed to provide an accuracy comparable to other multi-class classifiers. On all datasets (except CONNECT-4, PROTEIN, VOWEL), neural tree has shown performance within a margin of 3% from the best performing baseline. However, for all datasets, the neural trees have produced simpler classifier structures. This is evident when one compares the number of Neural Tree Nodes with that of the number of Support Vectors in SVM or the perceptrons in MLP (Table I). We also note that increasing the number of child nodes does not necessarily help in enhancing neural tree performance. A uniform choice of large splitting node children number might result in data insufficiency in partitions leading to over-fitting. Thus, the number of splitting node children must be reduced as one traverses down the tree.

We believe that data balancing plays a major role in fast reduction of impurities through the tree depth. However, the datasets have different distributions and varying extents of overlap. Hence, the effect of data balancing (in its current form) is not consistent. We believe that, such inconsistencies might be playing a significant role in hindering neural tree performance. Oversampling often blows up (near) minority classes with (probably) noisy samples. This solves inter-class imbalance but, enhances intra-class variations. Thus,

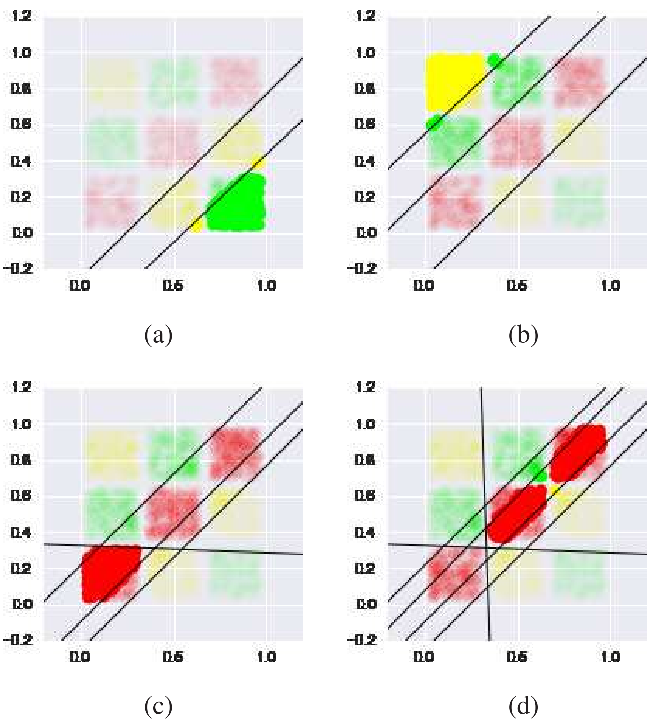

Fig. 3. We have illustrated the splits along a path in the neural tree built on a three-class checker-board dataset. The bright portion indicates the part of data going into a label node (color specifies the class) following successive partitioning by the nodes leading up to the label node.

TABLE I
TABLE SHOWS DETAILS OF 10 BENCHMARK DATASETS FROM UCI MACHINE LEARNING REPOSITORY. TEST SET (AVERAGE) ACCURACIES OF PROPOSED NEURAL TREE WITHOUT DATA BALANCING (NT) AND NEURAL TREE WITH PROGRESSIVE DATA BALANCING (NT+DB) ARE COMPARED WITH C4.5 DECISION TREE, SVM AND MLP. WE NOTE THAT THE NEURAL TREE PROVIDES COMPARABLE PERFORMANCE.

		CONNECT-4	DNA	LETTERDATA	PENDIGITS	PROTEIN	SATIMAGE	SENSIT VEHICLE	SENSORLESS	SHUTTLE	VOWEL
dataset info	Num classes	3	3	26	10	3	6	3	11	7	11
	Input Dimension	126	180	16	16	357	37	50	48	9	10
	Instances	67555	2000	15,000	7,494	17,766	4435	78,823	58,509	43,500	528
	Maj:Min	6.89	2.20	1.10	1.08	2.16	2.40	2.15	1	3537.7	1
NT	Accuracy	0.78	0.92	0.81	0.95	0.62	0.85	0.74	0.89	0.95	0.48
	Nodes	16	5	461	36	64	5	31	71	7	95
NT+DB	Accuracy	0.78	0.93	0.83	0.95	0.63	0.86	0.74	0.98	0.99	0.5
	Nodes	13	5	389	31	24	5	37	65	5	88
C4.5	Accuracy	0.76	0.91	0.83	0.87	0.50	0.81	0.661	0.99	0.99	0.38
	Nodes	9743	31	1443	285	3371	233	13481	853	49	95
SVM	Accuracy	0.93	0.94	0.81	0.14	0.67	0.65	0.76	0.98	0.99	0.6
	Support Vectors	4282	612	7697	7440	3603	502	7160	14764	942	381
MLP	Accuracy	0.83	0.90	0.82	0.92	0.70	0.82	0.76	0.99	0.99	0.10
	Nodes	264	378	136	72	726	98	112	140	46	64

TABLE II
TEST SET (AVERAGE) ACCURACY OF PROPOSED NEURAL TREE WITHOUT DATA BALANCING (NT) AND NEURAL TREE WITH PROGRESSIVE DATA BALANCING (NT+DB) ON 5 BENCHMARK DATASETS. IT IS OBSERVED THAT DATA BALANCING GENERALLY LEADS TO SIMPLER TREE STRUCTURES. HOWEVER, INCREASING NUMBER OF SPLITTING NODE CHILDREN DOES NOT NECESSARILY IMPROVE PERFORMANCE.

Child Nodes	Tree	Parameters	DNA	LETTERDATA	PENDIGITS	PROTEIN	SATIMAGE
2	NT	Acc	0.91	0.76	0.83	0.62	0.836
		Nodes	5	115	19	54	13
	NT+DB	Acc	0.89	0.73	0.95	0.63	0.84
		Nodes	5	109	35	36	15
3	NT	Acc	0.92	0.79	0.94	0.61	0.85
		Nodes	4	193	40	54	31
	NT+DB	Acc	0.92	0.80	0.95	0.63	0.86
		Nodes	4	169	31	24	37
4	NT	Acc	0.92	0.81	0.95	0.63	0.75
		Nodes	5	249	51	64	17
	NT+DB	Acc	0.93	0.83	0.95	0.63	0.85
		Nodes	5	389	33	699	37
5	NT	Acc	0.92	0.81	0.95	0.61	0.82
		Nodes	11	461	36	170	16
	NT+DB	Acc	0.92	0.81	0.95	0.60	0.83
		Nodes	11	356	31	125	41

one must reach a suitable balance between oversampling and undersampling in cases of highly skewed class distributions.

IV. CONCLUSION

A progressively balanced neural tree classifier is proposed for applications involving multiples classes. Multi-output single layer perceptrons with soft-max activations are used at splitting nodes. These perceptrons are learned from input data by minimizing an entropy impurity loss function. Input data at each node are balanced to prevent formation of biased perceptrons. The neural tree partitions the input space in a sequential top-down manner till pure partitions are reached. Such pure partitions correspond to leaf nodes. Majority class labels of leaf nodes are used as tree predictions. The proposed approach is benchmarked on ten datasets from UCI machine learning repository against three standard classifiers. Neural tree is observed to provide almost similar accuracies with lower classifier complexity.

This work has experimented with entropy impurity functions only. An immediate future extension aims at formulating neural trees with other impurity functions based on Gini index and Hellinger distance based measures. The present data balancing approach emphasizes on oversampling. This often disturbs the intra-class variation patterns in (near) minority classes thereby leading to over-fitting. We believe that data balancing strategy can be improved by striking a balance between the extent of oversampling and undersampling while maintaining both inter-class and intra-class balance. This work has also not explored the flexibility of accommodating multiple child nodes. A data dependent choice of number of child nodes at different depths may lead to lesser number of splitting nodes for large datasets.

REFERENCES

[1] S. R. Safavian and D. Landgrebe, "A survey of decision tree classifier methodology," *IEEE Transactions on Systems, Man, and Cybernetics*, vol. 21, pp. 660 – 674, 1991.

[2] A. P. Muniyandi, R. Rajeswari, and R. Rajaram, "Network anomaly detection by cascading k-means clustering and c4.5 decision tree algorithm," *Procedia Engineering*, vol. 30, no. Supplement C, pp. 174 – 182, 2012, international Conference on Communication Technology and System Design 2011.

[3] H. Jantan, A. R. Hamdan, and Z. A. Othman, "Human talent prediction in hrm using c4.5 classification algorithm," *International Journal on Computer Science and Engineering*, vol. 2, 2010.

[4] J. R. Quinlan, "Induction of decision trees," *Machine learning*, vol. 1, p. 81106, 1986.

[5] T. sien Lim, W. yin Loh, and Y.-S. Shih, "An empirical comparison of decision trees and other classification methods," Tech. Rep., 1998.

[6] T. K. Ho, "The random subspace method for constructing decision forests," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 20, no. 8, pp. 832–844, Aug. 1998.

[7] A. Sakar and R. J. Mammone, "Growing and pruning neural tree networks," *IEEE Trans. Comput.*, vol. 42, no. 3, pp. 291–299, Mar. 1993.

[8] P. E. Utgoff, "Perceptron trees: A case study in hybrid concept representations," in *Proceedings of the Seventh AAAI National Conference on Artificial Intelligence*, ser. AAAI'88. AAAI Press, 1988, pp. 601–606.

[9] M. Arun Kumar and M. Gopal, "A hybrid svm based decision tree," *Pattern Recogn.*, vol. 43, no. 12, pp. 3977–3987, Dec. 2010.

[10] G. L. Foresti and G. Pieroni, "Exploiting neural trees in range image understanding," *Pattern Recognition Letters*, vol. 19, no. 9, pp. 869 – 878, 1998.

[11] Y. Chen, B. Yang, J. Dong, and A. Abraham, "Time-series forecasting using flexible neural tree model," *Information Sciences*, vol. 174, no. 3, pp. 219 – 235, 2005.

[12] A. Bifet, G. Holmes, B. Pfahringer, and E. Frank, *Fast Perceptron Decision Tree Learning from Evolving Data Streams*. Berlin, Heidelberg: Springer Berlin Heidelberg, 2010, pp. 299–310.

[13] R. Balestriero, "Neural decision trees," *CoRR*, vol. abs/1702.07360, 2017.

[14] J.-E. Strömberg, J. Zrida, and A. Isaksson, "Neural trees : Using neural nets in a tree classifier structure," Linkping University, Automatic Control, Tech. Rep. 1157, 1991.

[15] R. Kannao and P. Guha, "Generic tv advertisement detection using progressively balanced perceptron trees," in *Proceedings of the Tenth Indian Conference on Computer Vision, Graphics and Image Processing*, ser. ICVGIP '16. New York, NY, USA: ACM, 2016, pp. 8:1–8:8.

[16] C. Micheloni, A. Rani, S. Kumar, and G. L. Foresti, "A balanced neural tree for pattern classification," *Neural Networks*, vol. 27, no. Supplement C, pp. 81 – 90, 2012.

[17] P. Kontschieder, M. Fiterau, A. Criminisi, and S. R. Bulò, "Deep neural decision forests," in *Proceedings of the Twenty-Fifth International Joint Conference on Artificial Intelligence*, ser. IJCAI'16. AAAI Press, 2016, pp. 4190–4194.

[18] N. V. Chawla, "C4.5 and imbalanced data sets: investigating the effect of sampling method, probabilistic estimate, and decision tree structure," in *In Proceedings of the ICML03 Workshop on Class Imbalances*, 2003.

[19] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," *CoRR*, vol. abs/1412.6980, 2014.

[20] M. Abadi, A. Agarwal, P. Barham, E. Brevdo, Z. Chen, C. Citro, G. S. Corrado, A. Davis, J. Dean, M. Devin, S. Ghemawat, I. Goodfellow, A. Harp, G. Irving, M. Isard, Y. Jia, R. Jozefowicz, L. Kaiser, M. Kudlur, J. Levenberg, D. Mané, R. Monga, S. Moore, D. Murray, C. Olah, M. Schuster, J. Shlens, B. Steiner, I. Sutskever, K. Talwar, P. Tucker, V. Vanhoucke, V. Vasudevan, F. Viégas, O. Vinyals, P. Warden, M. Wattenberg, M. Wicke, Y. Yu, and X. Zheng, "TensorFlow: Large-scale machine learning on heterogeneous systems," 2015.

[21] D. A. Cieslak and N. V. Chawla, "Learning decision trees for unbalanced data," in *Joint European Conference on Machine Learning and Knowledge Discovery in Databases*. Springer, 2008, pp. 241–256.

[22] P. Lenca, S. Lallich, T.-N. Do, and N.-K. Pham, "A comparison of different off-centered entropies to deal with class imbalance for decision trees," in *Pacific-Asia Conference on Knowledge Discovery and Data Mining*. Springer, 2008, pp. 634–643.

[23] S. Lomax and S. Vadera, "A survey of cost-sensitive decision tree induction algorithms," *ACM Computing Surveys (CSUR)*, vol. 45, no. 2, p. 16, 2013.

[24] S. Cateni, V. Colla, and M. Vannucci, "A method for re-sampling imbalanced datasets in binary classification tasks for real-world problems," *Neurocomputing*, vol. 135, pp. 32–41, 2014.

[25] T. Jo and N. Japkowicz, "Class imbalances versus small disjuncts," *SIGKDD Explor. Newsl.*, vol. 6, no. 1, pp. 40–49, 2004.

[26] N. V. Chawla, "C4. 5 and imbalanced data sets: investigating the effect of sampling method, probabilistic estimate, and decision tree structure," in *ICML*, vol. 3, 2003.

[27] G. M. Weiss, "The effect of small disjuncts and class distribution on decision tree learning," Ph.D. dissertation, Rutgers, The State University of New Jersey, 2003.

[28] C. Drummond, R. C. Holte *et al.*, "C4. 5, class imbalance, and cost sensitivity: why under-sampling beats over-sampling," in *Workshop on learning from imbalanced datasets II*, vol. 11, 2003.

[29] B. W. Yap, K. A. Rani, and Rahman, "An application of oversampling, undersampling, bagging and boosting in handling imbalanced datasets," in *Advanced Data and Information Engg.* Springer, 2014, pp. 13–22.

[30] G. E. Batista, R. C. Prati, and M. C. Monard, "A study of the behavior of several methods for balancing machine learning training data," *ACM Sigkdd Explorations Newsletter*, vol. 6, no. 1, pp. 20–29, 2004.

[31] S. Kotsiantis, D. Kanellopoulos, P. Pintelas *et al.*, "Handling imbalanced datasets: A review," *Transactions on Computer Sci. and Engg.*, vol. 30, no. 1, pp. 25–36, 2006.

[32] M. Lichman, "UCI machine learning repository," 2013.

[33] B. Schoelkopf, K. Sung, C. Burges, F. Girosi, P. Niyogi, T. Poggio, and V. Vapnik, "Comparing support vector machines with gaussian kernels to radial basis function classifiers," Cambridge, MA, USA, Tech. Rep., 1996.