Neural Decision trees

Final report

March 2019

Team members: Mahmud Allahverdiyev (MSc.1), Temirlan Seilov (MSc.1), Anton Razzhigaev (MSc.1), Alsu Sagirova (MSc.1)

1 Abstract

In this report, Neural Decision trees (NDT) - a novel approach which combines two important modelling in Machine Learning - Decision trees and Neural Networks, is discussed and the the methods to realize NDTs from the current literature will be addressed. The benchmarks of our implementation and direct comparison of individual and hybrid models, has been included in the results section.

2 Introduction

Decision trees (DT) have extensively being applied to wide range of problems in Computer Vision and Machine Learning. The ensembles of DT, Random Forests are the result of bagging procedure applied to the data. These trees are easy to interpret and parallelization over modern computing architecture is possible. Although the trees are not a good fit for online learning, due to favourable performance, they have been used in image classification and several other high-dimensional problems.

At the same time, Deep Neural Networks (DNN) have successfully outperformed conventional classification and regression models in different settings. While they are acting with a black box mechanism, they are capable of learning arbitrary functions and complex structures.

Let's compare these two modelling techniques in details. Decision trees make splits over the data until a stopping criterion is met[5]. The splits are based on a particular pair of a feature and a threshold; several split - impurity criterions, including entropy and gini have been proposed. With extremely large trees and reasonably big values of parameters, it is possible to overfit the data, thus ensembles of trees reduce variance dramatically while not affecting bias too much. Tree splits can be viewed as axis parallel which means we are limited in learning functions in this linear splitting fashion. Moreover, training trees with the lowest loss is NP-hard and the loss is not differentiable in the classical

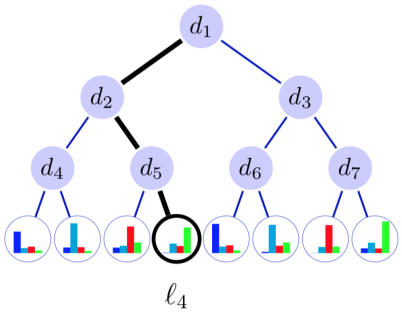

Figure 1: DT routing with learnable parameters

setting. Neural Networks have a completely distinct architecture. They are very adapt to complex structures, representation learning and other non-trivial tasks. Training NNs is much slower and computationally expensive compared to DT models. Another difference is that NNs can quickly absorb a new sample, thus suitable for online learning circumstances.

As we can see, both approaches have their own strength, yet visible issues. In the following sections of the report, the combined model - Neural Decision Trees will be discussed.

3 NDT construction

NDT based idea was first described in [4]. In their approach, the input data is fed into a [deep] neural network and the outputs of the fully-connected layer play the role of splitting decision trees. They introduce the NDT structure as following. Let's name decision nodes and leaf nodes of the DT as $\mathcal{N}$ and $\mathcal{L}$, respectively. Each node $l \in \mathcal{L}$ represents probability distribution for the outputs $\mathcal{Y}$. In contrast to deterministic splits in DTs, they parametrized the decision mechanism by establishing parameters Θ such that decision node $n \in \mathcal{N}$ is assigned the decision function $d_n(\cdot;\Theta) : \mathcal{X} \to [0,1]$. In other words, Θ defines the routing probabilities for inner nodes of the tree. [4] provides the output probabilities for the sample x over the tree T via:

$$P_T[y|x,\Theta,\pi] = \sum_{l\in\mathcal{L}} \pi_{ly}\mu_l(x|\Theta) \tag{1}$$

where π_{ly} denotes the probability of the sample x reaching leaf l on the class y. To satisfy differentiability and gradient based optimization, an activation function, i.e. sigmoid is applied to decide the tree to expand to the left or the right subtree:

$$d_n(x;\Theta) = \sigma(f_n(x;\Theta)) \tag{2}$$

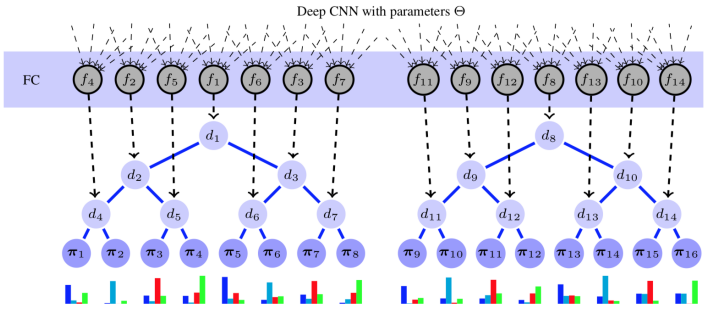

Figure 2: NDT realization with CNN + DT

To learn the parameters Θ, GD with respect to Θ is applied:

$$\Theta^{(t+1)} = \Theta^{(t)} - \eta \frac{\partial R}{\partial \Theta}(\Theta^{(t)}, \pi; \mathcal{B}) \tag{3}$$

where $\mathcal{B}$ denotes the mini-batch used in the update and the GD of loss $\mathcal{L}$

$$\frac{\partial L}{\partial \Theta}(\Theta, \pi; x, y) = \sum_{n \in \mathcal{N}} \frac{\partial L(\Theta, \pi; x, y)}{\partial f_n(x; \Theta)} \frac{\partial f_n(x; \Theta)}{\partial \Theta} \tag{4}$$

Overall, [4] summarizes their algorithm in the following fashion in Figure 2. The initial parameters Θ are randomly initialized in the beginning of the training. At each epoch, firstly, the π values of bottom layer - DT nodes, are calculated, then the training data is split into mini-batches for SGD update of Θ in (3). In this setting, the output units of the deep network determine the routing in the DT, rather than delivering the final predictions. The backpropagation can be understood in the way that gradient terms from the bottom level nodes of the DTs are calculated from their children nodes: for a node $n \in \mathcal{N}$ the accumulated gradient terms is $A_n = A_{nl} + A_{nr}$. This step required traversing the tree only once in bottom to top fashion.

[2] introduces NDTs as a way of resolving the issue between generalization and interpretability. Instead of getting an explanatory view from Neural Networks, they put forward the idea of training decision trees based on neural networks as in the previous approach. They deploy decision trees where each inner node acts contains a filter and the leaves carry the learned distributions. The probability of choosing the left or the right to go down with the sample x and the sigmoid function σ, is formulated as:

$$p_i(x) = \sigma(w_i x + b_i) \tag{5}$$

Their model can be compared to the mixture of experts models [3]; however, here the updates refrain the trees from always producing the same distribution. They present two ways of getting final probabilities: deriving the prediction via the leaf with the largest path probability and averaging over the whole set of

leaves of DT. σ in (5) allows the trees to make splits rather than axis-aligned ones defined by the features of the input.

Another novel architecture was introduced at [1] where NDTs consist of decision trees where each node is an independent multi-layer perceptron which can result in nonlinear splits due to activation functions. Interestingly, they indicate the interconnections with NDTs and Hashing Neural Networks (HNN) which contain a sigmoid at the final layer rather than a softmax. The mechanism of the model much resembles Locality Sensitive Hashing which aims to send same or similar objects to the exact bins; in this manner, splitting criteria can be optimized through iterative optimization routines rather than heuristic methods.

4 Challenges & Results

Our results are the following. [4] mentions their experiments on ImageNet with the top error 6.38% via an ensemble of 7 dNDF.NETs. [2] introduced an extra penalty term which was meant to prevent internal nodes from sending the data to only one of the left or right child. In our implementation, the same issue was also observed when training binary classifier. The approach taken to overcome the problem was similar to the one mentioned in [2]; it is possible to force the tree node to split the data in a specified ratio or according to a predetermined threshold based on prediction probabilities. Still, this procedure brought other difficulty with overfitting in moderate tree depths $d \geq 5$ on MNIST. Our baseline for MNIST with MLP embedded decision trees is 92.37% and the CNN architecture got 99.2%, while single MLP classifier with one hidden layer gets 87.4% test accuracy. On the other side, DNDF implementation produced 99.3%. The tree ensemble models from sklearn, RandomForestClassifier, DecisionTreeClassifier and ExtraTreesClassifier had $95\%, 93\%$ and 90.2% accuracy, respectively (without GridSearch and extensive parameter tuning). On the "malicious websites" dataset, our MLP implementation gets 88% test accuracy while the models from the libraries produced 95% maximum accuracy.

5 Team roles

The tasks for the project was divided among the team members during the first two weeks for the project stage. Temirlan and Anton have done the implementation and testing part for the DNDF model while Mahmud and Alsu managed the tasks on MLP-embedded DT and single CNN- DT models, respectively. Till the project presentation week, the team established 4 meetings to discuss the project state and assign new tasks to each of the team members. During the final week of the project, the results of the models implemented, have been thoroughly analyzed and the work has been finalized according to the requirements given by the course instructors and the feedback from the technical assistants.

6 Conclusion

In this report, we summarized main findings of [1], [2], [4]. They showed how to combine two different modelling techniques Decision Trees and Neural Networks to use their strength asynchronously. Different settings of these methods have been analyzed and the results have been replicated over artificial and real-life datasets.

References

[1] Randall Balestriero. Neural decision trees. *arXiv preprint arXiv:1702.07360*, 2017.

[2] Nicholas Frosst and Geoffrey Hinton. Distilling a neural network into a soft decision tree. *arXiv preprint arXiv:1711.09784*, 2017.

[3] Michael I. Jordan and Robert A. Jacobs. Hierarchical Mixtures of Experts and the EM Algorithm. *Neural Computation*, 2008.

[4] Peter Kontschieder, Madalina Fiterau, Antonio Criminisi, and Samuel Rota Bulo. ICCV15_DeepNDF_main. *The IEEE International Conference on Computer Vision (ICCV)*, 2015.

[5] S. Rasoul Safavian and David Landgrebe. A Survey of Decision Tree Classifier Methodology. *IEEE Transactions on Systems, Man and Cybernetics*, 1991.