Deep Oblique Decision Tree (DODT): A Novel Approach for Multi-class Classification

Tianye Ding
Khoury College of Computer Sciences, Northeastern University

Chengyi Kuang
Khoury College of Computer Sciences, Northeastern University

Yuqiao Su
Khoury College of Computer Sciences, Northeastern University

Zekai Shen
Khoury College of Computer Sciences, Northeastern University

Abstract
We present a novel machine learning architecture, the *Deep Oblique Decision Tree* (DODT), which combines the strengths of decision trees and deep learning techniques to address multi-class classification problems. The proposed DODT model integrates the interpretability and hierarchical structure of decision trees with the learning capacity and feature representation capabilities of deep neural networks. Our architecture employs oblique decision boundaries [1] to enhance the flexibility and expressiveness of the decision tree's splitting criteria, resulting in improved classification performance.

Keywords: classification, decision tree, deep learning, neural network

1 Introduction

Multi-class classification problems are ubiquitous in various fields, including image recognition, natural language processing, and bioinformatics. Existing algorithms often face challenges in achieving high prediction accuracy while maintaining computational efficiency.

Traditional algorithms like KNN and Decision Trees have been widely used for multi-class classification. However, these methods often suffer from low prediction accuracy and/or high computational cost, especially when dealing with large datasets with numerous features.

To address these challenges, we propose a novel approach called Deep Oblique Decision Tree (DODT). The DODT combines the strengths of deep learning and decision tree algorithms to achieve high prediction accuracy while maintaining computational efficiency. Specifically, the DODT algorithm uses a deep neural network to extract features from the input data, followed by a decision tree model that leverages the extracted features to make multi-class

predictions.

Our experiments on various datasets demonstrate that the DODT outperforms traditional classifiers in terms of both accuracy and computational time. Additionally, the DODT exhibits robustness against imbalanced data and noisy features, making it a reliable and versatile method for multi-class classification in a wide range of domains.

2 Data Analysis

We employed the "Wine_Quality _Data" dataset [2], sourced from Kaggle. This dataset encompasses an extensive range of wine's chemical properties such as acidity, sugar, pH, and alcohol content, as well as the wine's color. Although the dataset predominantly consists of continuous data representing various chemical properties, it also includes a single discrete feature, the wine's color (red or white), which can provide valuable insights into the factors influencing wine quality. The dataset contains approximately 6.5k rows, providing ample flexibility to divide it into separate test and training sets.

However, the dataset presents certain challenges and limitations that may affect its utility for machine learning tasks. The class distribution is imbalanced, with most samples centered around average quality scores and extreme quality scores being underrepresented. This imbalance can hinder classification algorithms' performance, particularly for minority classes. Additionally, the dataset's focus on continuous data, with only one discrete feature (color), might limit the advantages of having a mix of continuous and discrete data. Furthermore, the dataset provides no explanation of how the quality score is determined. If the score was assigned by tasters, it would inherently involve a degree of subjectivity, since individual preferences and experiences can influence their evaluations.

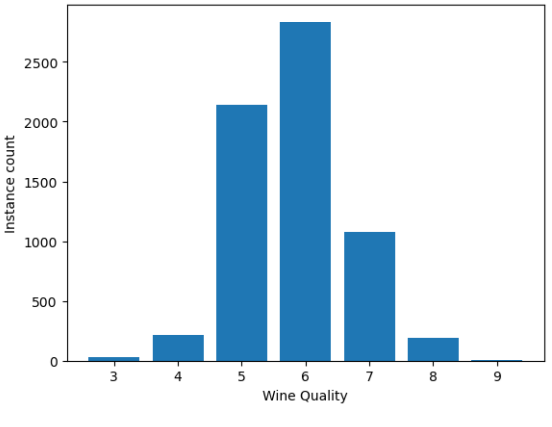


Figure 1: The distribution of wine quality.

3 Deep Oblique Decision Tree

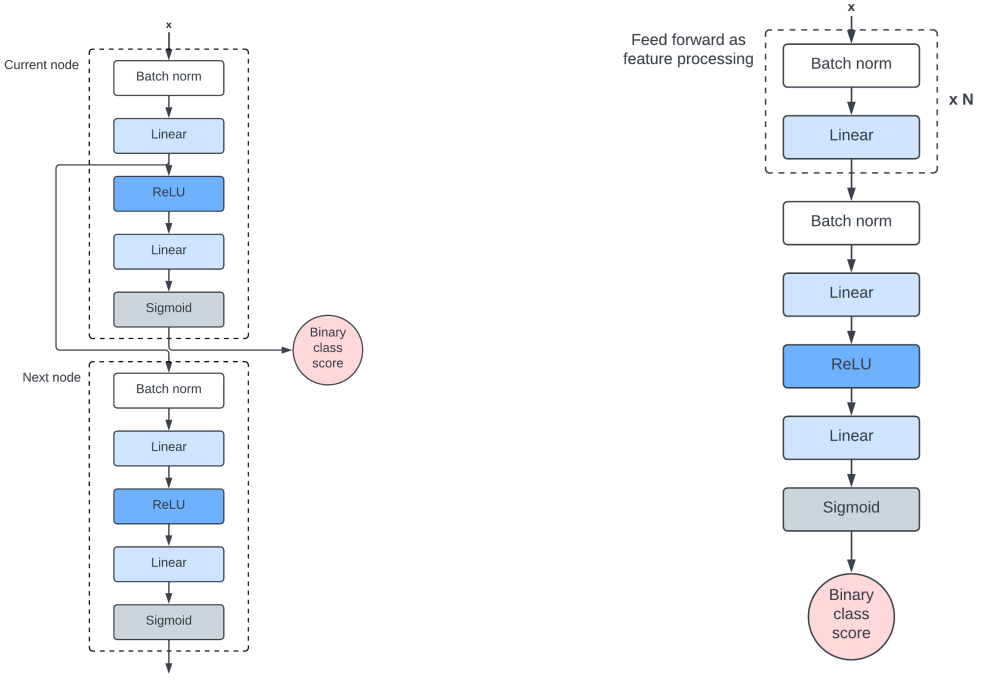
In our DODT model, we departed from the traditional decision tree approach of using a linear split and instead utilized a simple feed-forward neural network (Figure 2(a)) for splitting and feature vector processing. Rather than relying on a linear heuristic for splitting, our model maximizes the proportion of the target class in one split while minimizing it in the other. To train the network, we choose the class presented with the highest frequency within the current partition to be encoded as 1's and the rest to be encoded as 0's. While a single feed-forward model within a node may not completely filter out the target class, it will learn and process feature vectors by assigning higher weights to features that are more closely associated with the target class and lower weights to others. These weights can be further refined by the feed-forward models within subsequent nodes, ultimately achieving an accurate split. As a result, our partition process follows a coarse-to-fine pattern when navigating down a branch.

After training the feed-forward network for the specified number of epochs, the node performs a final run using the network and calculates a probability score for each entry to try and split the dataset. If the model is unable to split the dataset into two subsets, we assume that the feed-forward network within the current node is not deep enough to learn the underlying, more complex features of the dataset. In this case, the model will not perform a split in the node, and the entire processed dataset will be passed on to the next node, which effectively adds depth to the deep learning model (Figure 2(b)).

The feed-forward network's hyperparameters include the dimensions of the hidden layers (ff_dim), which remain consistent throughout the DODT architecture, the number of epochs to be trained (num_epochs), the learning rate of the optimizer ($learning\_rate$), the momentum strength of stochastic gradient descent, and the strength of L2-regularization (reg). The network structure comprises an input batch normalization layer that uses z-score, a linear layer with an input dimension of $input\_dim$, and an output dimension of ff_dim. In the root node, $input\_dim$ is the number of dimensions of the raw feature vectors and is the same as ff_dim in the rest of the architecture. The linear layer is followed by a ReLU activation layer, which adds non-linearity to the network, and another linear layer that takes ff_dim inputs and produces a final feature score. Finally, a sigmoid activation layer computes the probability of the instance being the class encoded as 1. To speed up the convergence rate, the linear layers' weight initialization uses Xavier initialization, which initializes the weights through a normal distribution of $\mathcal{N}(0, \frac{1}{n})$, where n denotes the number of dimensions of the input features. And the complete DODT architecture also has hyperparameters for maximum depth (max_depth) and target impurity ($target\_impurity$) similar to the traditional decision tree.

The tree stops expanding when it reaches the given max_depth or $target\_impurity$ on the current partition. The final class for the current node is determined by the majority of class instances within the partition, following the traditional decision tree approach.

When passing data to the next node, whether it's partitioned or unpartitioned, the pro-



(a) Feed-forward as oblique splitting

(b) Feed-forward as deep learning

Figure 2: An overview of our feed-forward neural network and the deep learning architecture. The computed binary class probability scores will be used to partition the processed feature vectors from the first linear layer (a). Gradient backpropagation will only be applied within the current feed-forward network. When navigating along a branch, the feed-forward layers will function like deep learning neural networks (b).

cessed feature vectors will be the output from the first linear layer before being fed to the ReLU activation layer. This is because we want the processed feature vectors to be passed into the next node without necessarily going through ReLU. Negative weight values assigned to certain features may be associated with the 0-encoded classes, and the non-linearity can be achieved through the non-linear split within the current node.

4 Method

4.1 Feature selection

In our dataset, the features display a low degree of correlation, prompting us to incorporate all of them during the model training process. Despite our efforts to implement Principal

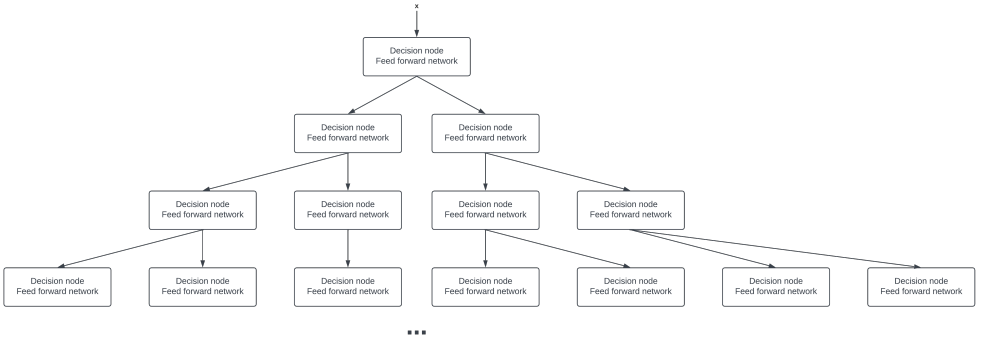


Figure 3: Illustration of an example resultant DODT structure with successful dataset partitions and possible deep neural networks.

Component Analysis (PCA), the outcome was a decrease in accuracy. DODT, however, has the ability to uncover underlying features stemming from various data combinations through the use of neural networks. Consequently, feature selection may not be particularly beneficial in this context.

4.2 Models and Rational

We investigate the performance of Deep Oblique Decision Trees (DODT) as a solution to multi-class classification problems. As a comparison, we consider two baseline models: K-Nearest Neighbors (KNN) and traditional Decision Trees (DT). KNN, although adaptive, serves as a naive approach due to its sensitivity to irrelevant features, resulting in a low baseline model. On the other hand, Decision Trees, a popular multi-class classification algorithm, offer reasonable accuracy and the ability to perform automatic feature selection, positioning it as a medium baseline model. The proposed DODT method integrates neural networks into the decision tree framework, aiming to enhance its performance by leveraging the benefits of both techniques. To evaluate the efficacy of DODT, we compare its performance against KNN and traditional Decision Trees, examining how well it addresses the limitations of these baseline models while providing improved accuracy and generalization in multi-class classification tasks.

4.3 hyperparameter Tuning

As we did not use any GPU-supported library and all model source codes were written in plain Python, optimizing performance was a major challenge during hyperparameter tuning. To overcome this obstacle, we employed parallelism and specifically leveraged multi-processing

techniques. This approach allowed us to train several models simultaneously and significantly reduced the time required for tuning DODT, DT, and KNN models. In an ideal scenario, multi-threading would be the preferred option due to its lower memory requirements and higher efficiency compared to multi-processing. However, the presence of the Python Global Interpreter Lock (GIL) restricts the execution of Python byte code to only one thread at a time. Consequently, multi-threading does not provide any performance improvement. To address this issue, multi-processing can be used as each process operates with its own GIL, allowing for parallelism and efficient utilization of system resources.

4.4 Cross-validation

For each type of model, they were trained on all possible, reasonable combinations of hyperparameters. For example, the data was first split into a training and testing set, with the training set comprising 70% of the original data. Then, for each hyperparameter combination, the models were trained on 80% of the training set and validated on the remaining 20% of the data. The model with the highest validation accuracy was selected for testing.

5 Analysis

The performance comparison of the DODT against KNN and traditional DT demonstrates the benefits of integrating neural networks with decision trees. The test accuracies of the three algorithms are as follows:

1. KNN: Test accuracy of 45.0% with $k = 6$ as the optimal hyperparameter. KNN's relatively lower test accuracy suggests that it might not be well-suited for the specific problem at hand, possibly due to its sensitivity to irrelevant features or noise in the data.

2. DT: Test accuracy of 52.7% with optimal hyperparameters including entropy-based impurity function, maximum tree depth of 5, a minimum of 2 instances per leaf node, and a target impurity of 0.0. The Decision Tree classifier shows a marked improvement in accuracy compared to KNN, indicating its ability to capture the underlying structure of the data and perform automatic feature selection.

3. DODT: Test accuracy of 60.2% with optimal hyperparameters such as ff_dim of 15, momentum of 0.9, max_depth of 20, $target\_impurity$ of 0.1, $learning\_rate$ of 1×10^{-6}, 500 training epochs, and reg of 0.1. DODT's superior performance highlights the advantages of combining decision trees with neural networks, resulting in a more powerful and versatile model that can better capture the underlying patterns in the data.

The DODT algorithm outperforms both KNN and traditional Decision Tree models in terms of test accuracy. Its superior performance can be attributed to the integration of neural networks within the decision tree framework, enabling it to address the limitations of KNN and DT while delivering improved performance. This comparison suggests that the DODT

model is the suitable choice for multi-class classification problems, providing a more robust solution by leveraging the strengths of both decision trees and neural networks.

6 Discussion

While gradient backpropagation is only performed within each feed-forward network, this approach addresses issues such as gradient vanishing and explosion when propagating through deep neural networks, while also improving computational efficiency. The DODT architecture is flexible and can be adapted to complex and imbalanced datasets, but it is also prone to overfitting, for example, a model with ff_dim of 12, max_depth of 15, and $target\_impurity$ of 0.1 achieves a training accuracy of 82% on the wine quality dataset but only 50% accuracy on validation and testing. The hyperparameter tuning process presents trade-off dilemmas that make it difficult to fine-tune the model. Limiting the max_depth of the architecture to prevent overfitting upon the decision tree structure also limits the deep learning structure's capability to learn complex and implicit features, and vice versa.

For potential future improvements, there are several areas we could explore for both the architecture and hyperparameters. On the architecture side, one possible improvement would be to refine how the model determines if it has reached the max_depth. For instance, we could avoid incrementing the current depth when the dataset remains unpartitioned when passing to the next node. This could allow for various neural network depths from a high-level view. Another potential modification would be to add a dropout layer to our feed-forward network to further restrict the deep neural network from overfitting. Additionally, we could experiment with adding residual connections across nodes on the same branch to enable a full network gradient flow, though this could also lead to gradient mixing on a branching node. On the hyperparameter side, we could introduce more optional flexibilities to the hyperparameter tuning process to give more manual control over the architecture. For example, we could consider exploring different activation functions or regularization techniques or allowing for more fine-tuning of the learning rate or batch size. Overall, there are many potential avenues for improving the architecture and hyperparameters of the DODT model, and further exploration is needed to determine the most effective strategies.

7 Conclusion

In conclusion, this study has demonstrated the effectiveness of the Deep Oblique Decision Tree (DODT) algorithm compared to traditional machine learning models, such as K-Nearest Neighbors (KNN) and Decision Trees (DT), on a dataset exhibiting diverse characteristics and a highly imbalanced target class distribution. The DODT algorithm successfully combines the strengths of both decision trees and neural networks to achieve outstanding performance, making it a suitable choice for multi-class classification problems.

The adaptability of the DODT algorithm to various data distributions and its ability to effectively address imbalanced class distributions emphasize its versatility and potential for delivering improved performance across a wide range of domains and complex problem settings.

Potential future research directions include investigating the applicability of DODT to different types of data and problems with diverse levels of class imbalance, exploring potential enhancements and optimizations to further augment its performance, and comparing DODT with other robust multi-class classification algorithms to gain deeper insights into its unique strengths and limitations.

8 Author Contributions

Tianye Ding: Proposed the network architecture and implementation of the forward-backward passes within the feed-forward network with other deep learning techniques.
Chengyi Kuang: Constructed the self-referential tree structure of DODT. Proposed and built the generic multi-process hyperparameter tuning framework. Performed hyperparameter tuning on KNN.
Yuqiao Su: Applied DODT hyperparameter tuning on the data set, contributed to functions in the code base, participated in essay writing, and completed the poster.
Zekai Shen: Performed hyperparameter tuning on decision tree. Contributed to features and functions of the code base and helped debugging process. Participated in poster and report writing.

9 References

Breiman, Leo. Classification and Regression Trees. Wadsworth International Group, 1984.

Khaled, Ghassen. Wine_Quality_Data. February 2023, Kaggle, www.kaggle.com/datasets/ghassenkhaled/wine-quality-data?resource=download.