

# Decision Tree Embedding by Leaf-Means

Cencheng Shen, Yuexiao Dong, Carey E. Priebe

**Abstract**—Decision trees and random forest remain highly competitive for classification on medium-sized, standard datasets due to their robustness, minimal preprocessing requirements, and interpretability. However, a single tree suffers from high estimation variance, while large ensembles reduce this variance at the cost of substantial computational overhead and diminished interpretability. In this paper, we propose Decision Tree Embedding (DTE), a fast and effective method that leverages the leaf partitions of a trained classification tree to construct an interpretable feature representation. By using the sample means within each leaf region as anchor points, DTE maps inputs into an embedding space defined by the tree's partition structure, effectively circumventing the high variance inherent in decision-tree splitting rules. We further introduce an ensemble extension based on additional bootstrap trees, and pair the resulting embedding with linear discriminant analysis for classification. We establish several population-level theoretical properties of DTE, including its preservation of conditional density under mild conditions and a characterization of the resulting classification error. Empirical studies on synthetic and real datasets demonstrate that DTE strikes a strong balance between accuracy and computational efficiency, outperforming or matching random forest and shallow neural networks while requiring only a fraction of their training time in most cases. Overall, the proposed DTE method can be viewed either as a scalable decision tree classifier that improves upon standard split rules, or as a neural network model whose weights are learned from tree-derived anchor points, achieving an intriguing integration of both paradigms.

**Index Terms**—Decision tree, anchor points, random forest, neural network, supervised learning



## 1 INTRODUCTION

Decision trees are among the most interpretable and foundational models in supervised machine learning. A decision tree recursively partitions the feature space using axis-aligned splits to minimize an impurity measure, such as Gini impurity or mean squared error, producing a hierarchy of rules that map inputs to class labels or responses [1], [2]. Despite their intuitive appeal and interpretability, a single decision tree often suffer from high variance and overfitting.

An ensemble of decision trees, most notably random forest (RF) [3], [4], constructs a collection of trees, each trained on a bootstrap sample of the data and using a random subset of features at each split. The final prediction is obtained by averaging (for regression) or majority voting (for classification) across all trees. This ensemble strategy substantially reduces variance while maintaining low bias, yielding a strong, general-purpose learner that often achieves much better predictive accuracy than a single decision tree and performs well across a wide range of tasks. Since then, numerous works have sought to improve random forest, including approaches based on oblique splits, sparsity, or rotation-based transformations [5], [6], [7].

On the other hand, neural networks (NN) have been the foundational component behind recent progress on complex data regimes such as text, images, and videos. They are composed of multiple layers of affine transformations and nonlinear activations, often supplemented with techniques such as dropout and skip connections [8], [9] to facilitate effective training. Early work established that feed-forward

networks are universal function approximators [10]. With the advent of efficient backpropagation and GPU computing, deep neural networks have become highly scalable to large datasets far beyond the capacity of many traditional machine learning methods, and this scalability has driven state-of-the-art results across vision, speech, and language domains [11], [12]: Convolutional neural networks emerged as the dominant architecture for image processing tasks [13], while more recent transformer-based architectures employing self-attention mechanisms [14] have revolutionized sequence modeling and natural language processing. Deep learning thus represents the prevailing framework in modern machine learning, providing significant advantages in scalability to big data and in inference speed compared to other machine learning families.

Nevertheless, recent empirical studies have shown that random forest and other tree-based ensemble methods outperform neural networks in many standard data settings [15], [16], [17], particularly those involving mixtures of numerical and categorical variables with heterogeneous scales. Random forest require minimal preprocessing, and often work well out of the box without extensive effort devoted to complex architectures, hyperparameter tuning, or regularization. As a result, they tend to generalize better on moderate-sized standard datasets, where deep networks are more susceptible to overfitting or undertraining.

Despite its versatility, random forest suffers from significant computational bottlenecks [18]. First, training hundreds of trees can be computationally expensive for very large datasets or high-dimensional feature spaces. Second, when individual trees grow deep, inference can become considerably more costly. Third, the memory cost can be substantial for large ensembles, and inference latency scales with the number of trees. In addition, although a single decision tree is interpretable, the ensemble as a whole can behave like a

- Cencheng Shen is with the Department of Applied Economics and Statistics, University of Delaware, E-mail: shenc@udel.edu
- Yuexiao Dong is with the Department of Statistics, Operations, and Data Science, Temple University, E-mail: ydong@temple.edu
- Carey E. Priebe is with the Department of Applied Mathematics and Statistics (AMS), the Center for Imaging Science (CIS), and the Mathematical Institute for Data Science (MINDS), Johns Hopkins University, E-mail: cep@jhu.edu

black box.

There also exists many works exploring hybrid strategies combining random forest with other methods, such as logistic regression [19], as well as abundant explorations in various ways to incorporate neuron networks [20], [21], [22], [23], [24], [25], [26]. However, most hybrid RF-NN models remain ad hoc and application-driven, with limited theoretical understanding. Computationally, such hybrids almost always incur increased training time and memory usage, as they often combine the ensemble complexity of forest with the parameter-rich nature of neural networks.

Given a dataset  $(\mathbf{X}, \mathbf{Y})$ , where  $\mathbf{X} \in \mathbb{R}^{n \times p}$  represents  $n$  training samples with  $p$  features and  $\mathbf{Y} \in \{1, \dots, K\}^n$  denotes the corresponding class labels, we propose the decision tree embedding (DTE) method based on a single classification tree. The central idea is to use the sample means of each leaf region as anchor points that transform the input into a new embedding space, yielding  $\mathbf{Z} \in \mathbb{R}^{n \times m}$ . Although the axis-aligned splitting rules of a single decision tree have high variance and thereby degrade inference quality, our key insight is that the sample means within the leaf regions have much lower variance by its aggregating nature. Consequently, these anchor points extracted from a single tree yield a stable and informative embedding that can be readily used for subsequent classification. The resulting formulation can be viewed either as a scalable tree-based classifier, or as a neural network model whose weights are learned from decision trees.

Section 2 presents the main method, along with an ensemble extension using additional bootstrap trees, a subsequent classification step based on linear discriminant analysis, a computational complexity analysis, and qualitative comparisons with random forest and classification neural networks. Section 3 discusses the population-level theoretical properties of DTE, including its preservation of conditional density under mild conditions and a characterization of the resulting classification error. Section 4 visualizes the intrinsic mechanism of DTE using synthetic data, and Section 5 shows that DTE achieves an excellent balance between accuracy and running time, comparing favorably with decision trees, random forest, and shallow neural networks.

## 2 METHOD

### 2.1 Decision Tree Embedding using A Single Tree

Our main method proceeds as follows:

- **Input:** A labeled dataset  $(\mathbf{X}, \mathbf{Y}) \in \mathbb{R}^{n \times p} \times \{1, \dots, K\}^n$ .
- **Step 1:** Train a standard classification tree that partitions the feature space  $\mathbb{R}^p$  into disjoint regions  $\{\mathcal{R}_1, \mathcal{R}_2, \dots, \mathcal{R}_m\}$ , each corresponding to a leaf region. Let  $S_j = \{i : \mathbf{X}_i \in \mathcal{R}_j\}$  denote the set of sample indices that fall into region  $\mathcal{R}_j$ .
- **Step 2:** Compute the mean for each leaf region:

$$\mu_j = \frac{1}{|S_j|} \sum_{i \in S_j} \mathbf{X}(i, :), \quad j = 1, \dots, m.$$

Collect these leaf means into a matrix  $\mathbf{W} = [\mu_1; \mu_2; \dots; \mu_m] \in \mathbb{R}^{m \times p}$ .

- **Step 3:** Construct the embedding  $\mathbf{Z}$  as

$$\mathbf{Z} = \mathbf{X}\mathbf{W}^\top + \mathbf{1}\mathbf{b}^\top \in \mathbb{R}^{n \times m},$$

where  $\mathbf{1}$  is an  $n \times 1$  vector of ones, and  $\mathbf{b}$  is an  $m \times 1$  intercept vector defined as  $\mathbf{b}(j) = |\mathbf{W}(j, :)|^2$  for  $j = 1, \dots, m$ .

- **Output:** Final embedding  $\mathbf{Z}$ , weight matrix  $\mathbf{W}$ , and intercept vector  $\mathbf{b}$ .

Intuitively, each column of  $\mathbf{Z}$  measures the affinity of samples to a specific leaf region, determined by their inner product with the corresponding leaf mean. Each leaf mean serves as a data-driven anchor point summarizing a local region, and the embedding dimension equals the number of leaves. We refer to this single-tree version as DTE<sub>1</sub>.

### 2.2 Decision Tree Embedding with Additional Bootstrap Trees

The single-tree embedding naturally extends to an ensemble variant, denoted DTE<sub>t</sub> for an ensemble of  $t$  trees. The first tree is always trained on the original dataset, while the remaining  $t - 1$  trees are trained on independent bootstrap samples. Each tree  $s = 1, \dots, t$  yields its own leaf-mean matrix  $\mathbf{W}^{(s)}$  and intercept vector  $\mathbf{b}^{(s)}$ , which are concatenated to form the final embedding.

- **Input:** A labeled dataset  $(\mathbf{X}, \mathbf{Y})$ , and a positive integer  $t$ .
- **Step 1:** Compute the single-tree embedding at  $s = 1$ :

$$[\mathbf{Z}^{(1)}, \mathbf{W}^{(1)}, \mathbf{b}^{(1)}] = \text{DTE}_1(\mathbf{X}, \mathbf{Y}).$$

- **Step 2:** For  $s = 2, \dots, t$ , bootstrap the data to obtain  $(\mathbf{X}^{(s)}, \mathbf{Y}^{(s)})$ , and compute

$$[\mathbf{Z}^{(s)}, \mathbf{W}^{(s)}, \mathbf{b}^{(s)}] = \text{DTE}_1(\mathbf{X}_{trn}^{(s)}, \mathbf{Y}_{trn}^{(s)}).$$

- **Step 3:** Concatenate the results:

$$\mathbf{W} = \begin{bmatrix} \mathbf{W}^{(1)} \\ \mathbf{W}^{(2)} \\ \vdots \\ \mathbf{W}^{(t)} \end{bmatrix}, \quad \mathbf{b} = [\mathbf{b}^{(1)}, \mathbf{b}^{(2)}, \dots, \mathbf{b}^{(t)}],$$

$$\mathbf{Z} = [\mathbf{Z}^{(1)}, \mathbf{Z}^{(2)}, \dots, \mathbf{Z}^{(t)}],$$

- **Output:** Final embedding  $\mathbf{Z}$ , weight matrix  $\mathbf{W}$ , and intercept  $\mathbf{b}$ .

Suppose the  $s$ th tree has  $m^{(s)}$  leaf regions. Then the dimension of  $\mathbf{Z}$  from DTE<sub>t</sub> equals the total number of leaf regions across all trees, i.e.,  $m = \sum_{s=1}^t m^{(s)}$ . While alternative aggregation schemes for combining multiple embeddings are possible, direct concatenation is both simple and information-preserving.

### 2.3 Classification by LDA

The decision tree embedding  $\mathbf{Z}$ , whether constructed from a single tree or an ensemble, can be seamlessly paired with a lightweight linear classifier such as linear discriminant analysis (LDA) for downstream prediction tasks. Given a train/test split, the procedure is as follows:

- **Input:** Training data  $(\mathbf{X}_{trn}, \mathbf{Y}_{trn})$ , testing data  $\mathbf{X}_{tsn}$ , and the number of trees  $t$ .
- **Training:** Compute the training embedding:

$$[\mathbf{Z}_{trn}, \mathbf{W}, \mathbf{b}] = \text{DTE}_t(\mathbf{X}_{trn}, \mathbf{Y}_{trn}).$$

- **Testing Data Embedding:** Project the test data into the embedded space:

$$\mathbf{Z}_{tsn} = \mathbf{X}_{tsn} \mathbf{W}^\top + \mathbf{1} \mathbf{b}^\top.$$

- **Testing:** Train an LDA classifier on  $(\mathbf{Z}_{trn}, \mathbf{Y}_{trn})$ , then apply it to  $\mathbf{Z}_{tsn}$ .
- **Output:** Predicted test labels  $\hat{\mathbf{Y}}_{tsn}$ .

In cross-validation, the predicted labels are compared to ground truth to compute classification accuracy. While other classifiers, such as logistic regression, shallow neural networks, or nearest-neighbor methods, can also be used atop the decision tree embedding, LDA offers an excellent balance of computational efficiency and accuracy.

### 2.4 Computational Complexity

Let  $n$  be the number of samples,  $p$  the feature dimension,  $m$  the number of leaf regions,  $t$  the number of trees, and  $K$  the number of classes in the downstream classifier.

- Training  $t$  classification trees has time complexity  $O(tnp \log n)$ ; when done in parallel, which is feasible since DTE typically uses only a small  $t$ , the cost reduces to  $O(np \log n)$ .
- Computing all leaf means requires aggregating each sample once, for a total cost of  $O(np)$ .
- The matrix multiplication  $\mathbf{Z} = \mathbf{X} \mathbf{W}^\top$  costs  $O(npm)$ .
- LDA costs  $O(nm^2 + m^3)$  for training, and per-sample testing time is  $O(m^2 + K * m)$ .

Since  $p$ ,  $m$ , and  $K$  are typically much smaller than  $n$ , the overall computational complexity is effectively linear in  $n$  for large sample sizes. Compared with random forest, the decision tree embedding requires only a few trees (we found  $t = 1$  or  $t = 3$  to be sufficient), making it substantially faster and more memory-efficient, as most of the cost arises from tree construction.

In practice, a few additional considerations may affect efficiency. For example, if a single tree produces too many leaf regions (large  $m$ ), the classification step by LDA can slow down. This can be mitigated by limiting the tree depth or the number of splits, although we did not apply such constraints in this paper.

### 2.5 Comparison to Tree and Forest Classification

The DTE method requires building decision trees, but the subsequent classification step fundamentally differs from the splitting rules used in tree-based classifiers. In standard decision trees, small perturbations in the data can lead to

large deviations in the resulting split. Consequently, a single tree often exhibits high variance. Random forest mitigates this by aggregating many such unstable trees, thereby reducing variance through averaging.

In contrast, DTE does not use the split rules at all for subsequent classification. Instead, it uses the sample means of the leaf regions, which has lower variance than the split rules because of sample averaging in the leaf region.

An illustrative example is the following. Suppose a random variable  $X$  satisfies  $X|Y = 1 \in [-1, 0]$ , and  $X|Y = 2 \in [0, 1]$ , uniformly distributed in each respective interval. The perfect split is clearly at 0. A decision tree using limited training samples may place the split at, say, 0.04, which is suboptimal for testing data from the same distribution. A random forest would reduce the deviation: different bootstrap samples may produce splits around 0.02,  $-0.02$ ,  $-0.03$ , etc., and averaging these yields a split location closer to 0 than any single tree.

Now consider the leaf means. With the perfect split at 0, the ideal anchor points are simply the true class means,  $[-0.5, 0.5]$ . Using the imperfect split at 0.04 instead yields empirical leaf means at  $[-0.48, 0.52]$ . Both sets of anchor points linearly separate the two classes, and the discrepancy between them is smaller than the discrepancy between split locations themselves. This illustrates why the embedding is robust against sampling noise with just a single tree.

Therefore, DTE can be viewed as a fast and efficient tree-based classifier that inherently reduces the variance arising from unstable split rules. The variance reduction comes from averaging samples within each leaf region, whereas random forest reduces variance by aggregating many diverse trees, a fundamentally different and more computationally expensive mechanism. Random forest typically requires many trees to stabilize performance, whereas DTE often achieves similar performance with just one or a few trees. In fact, using too many trees can offset the benefits of the embedding during LDA classification. As  $t$  increases, the embedding dimension  $m$  grows proportionally, leading to higher estimation variance and increased running time for LDA.

### 2.6 Relationship to Classification Neural Networks

Once a decision tree is built, DTE becomes a purely linear model. In our proposed classification pipeline, DTE provides a linear embedding with an intercept, followed by LDA, which is another linear transformation with an intercept. Consequently, the overall model is equivalent to a shallow two-layer neural network: DTE serves as the hidden layer with  $m$  neurons, and LDA acts as the output layer. In particular, the leaf regions and their associated anchor points can be viewed as a sufficient transformation in the sense of [27].

Because model inference in DTE with LDA involves only basic matrix operations similar to a trained neural network, it inherently avoids the potentially expensive tree traversal process used in random forest, where inference time increases with both tree depth and the number of trees.

Therefore, DTE can be viewed as a neural network model, except that the number of neurons  $m$  and the weight matrices are learned from the structure of a decision tree,

rather than from random initialization followed by back-propagation. It is often the case that constructing a single classification tree is significantly faster than running gradient-based training. While this does not guarantee superiority over back-propagation, when the decision tree provides a good partition of the feature space, DTE is expected to perform favorably.

### 3 THEORY

While the previous sections considered the sample-based version of the method, working with sample data  $\mathbf{X}$  and the embedding  $\mathbf{Z}$ , in this section we study the corresponding random-variable formulation. Specifically, we analyze the population-level behavior of the method in terms of random variables  $X$  and  $Y$ . Under the classical i.i.d. assumption, each sample pair  $(\mathbf{X}(i, :), \mathbf{Y}(i))$  is drawn independently from the joint distribution of  $(X, Y)$  for  $i = 1, \dots, n$ .

Our focus is exclusively on classification, so  $X$  takes values in  $\mathbb{R}^p$  and  $Y$  is a discrete label in  $\{1, \dots, K\}$ . We denote their supports by  $\mathcal{X}$  and  $[K]$ , respectively.

#### 3.1 Bayes-homogeneous Partition and Conditional Probability Preservation

We first establish that the decision tree embedding preserves the conditional class distribution whenever the tree's leaf regions are Bayes-homogeneous, defined as follows.

**Definition 1.** Given random variables  $(X, Y) \in \mathcal{X} \times [K]$ , a set of regions  $\{\mathcal{R}_1, \dots, \mathcal{R}_m\}$  partitioning the support  $\mathcal{X}$  is said to be Bayes-homogeneous, if the conditional distribution of  $Y$  given  $X$  is constant within each region. That is, for every region  $\mathcal{R}_j$  and all pairs of points  $x, x' \in \mathcal{R}_j$

$$P(Y | X = x) = P(Y | X = x')$$

The set of regions is said to be  $\epsilon$ -Bayes-homogeneous if, their conditional distributions satisfy

$$\|P(Y | X = x) - P(Y | X = x')\|_1 \leq \epsilon,$$

where  $\epsilon$  is the maximum deviation across all regions.

This definition is flexible enough to allow arbitrary  $\epsilon$ , making it universally applicable. In practice, a well-behaved decision tree is expected to yield a set of leaf regions with a small  $\epsilon$ . Our first theorem shows that the decision tree embedding approximately preserves the conditional class distribution up to this same bound, and the preservation becomes exact in the ideal case  $\epsilon = 0$ .

Note that we use the notation  $P(Y | X)$  to denote the entire conditional distribution of  $Y$  given  $X$ , i.e.,

$$P(Y | X) = (P(Y = 1 | X), \dots, P(Y = K | X)),$$

a vector in the probability simplex. When referring to a specific class  $c$ , we explicitly write  $P(Y = c | X)$ .

**Theorem 1.** Let  $(X, Y)$  be a random pair with  $X \in \mathbb{R}^p$  and  $Y \in [K]$ . Let  $\{\mathcal{R}_1, \dots, \mathcal{R}_m\}$  be the set of leaf regions from a classification tree, and let  $Z = X\mathbf{W}^\top + \mathbf{1b}^\top$  denote its Decision Tree Embedding, where

$$\mu_j = \mathbb{E}[X | X \in \mathcal{R}_j], \quad \mathbf{b}_j = -\|\mu_j\|^2, \quad \mathbf{W} = [\mu_1^\top, \dots, \mu_m^\top]^\top.$$

If the partition  $\{\mathcal{R}_j\}$  is  $\epsilon$ -Bayes-homogeneous, then

$$\|P(Y | X) - P(Y | Z)\|_1 \leq \epsilon.$$

In particular, when  $\epsilon = 0$ , the DTE embedding is sufficient for  $Y$ ; that is,

$$P(Y | X) = P(Y | Z) \quad \text{a.s.}$$

Therefore, DTE is able to approximately preserve the conditional label probabilities. Each leaf region  $\mathcal{R}_j$  defines an anchor point  $\mu_j$ , whose corresponding coordinate in  $Z$  serves as a similarity-based weight. In the ideal case, DTE is fully sufficient, providing a lossless transformation of the underlying class-conditional structure.

#### 3.2 Population Classification Error Bound

Our next theorem derives a classification error bound.

**Theorem 2.** Under the same notation as Theorem 1, and further assume that each point in a leaf region  $\mathcal{R}_j$  is closer (in Euclidean norm) to its own region mean  $\mu_j$  than to any other region mean  $\mu_{j'}$ .

For each class  $c \in [K]$ , let

$$\mathcal{J}_c = \{j : \arg \max_{c'} P(Y = c' | X \in \mathcal{R}_j) = c\}$$

be the index set of leaves whose majority label is  $c$ . We then consider the indicator classifier on the DTE embedding:

$$g(Z) = \arg \max_{c \in \{1, \dots, C\}} \gamma_c^\top Z, \quad (\gamma_c)_j = \begin{cases} 1, & j \in \mathcal{J}_c, \\ 0, & \text{otherwise.} \end{cases}$$

For each  $j$ , let

$$l_j = 1 - \max_c P(Y = c | X \in \mathcal{R}_j)$$

be the impurity level at each region. Then the classification error of the indicator rule  $g$  equals

$$L_g = \sum_{j=1}^m P(X \in \mathcal{R}_j) l_j.$$

In particular, if every leaf region is pure, we have  $l_j = 0$  and  $L_g = 0$ , such that the DTE embedding achieves perfect classification with the indicator rule.

Therefore, when the decision tree performs well so that the resulting leaf partition  $\{\mathcal{R}_1, \dots, \mathcal{R}_m\}$  is pure and the regions are sufficiently well-clustered, meaning each point is closer to its own leaf mean than to any other leaf mean, we have  $P(Y = c | X \in \mathcal{R}_j) = 1$  for some class  $c$ . In this case, the classification induced by the DTE embedding is also perfect.

We remark that the indicator classification rule considered in Theorem 2 is not necessarily optimal; rather, it is chosen for analytical simplicity and to yield a conservative error bound. In practice, the linear discriminant analysis classifier used in the sample version of our method is more flexible and typically achieves better performance, as it leverages the empirical covariance structure of  $Z$  to produce better decision boundaries in finite-sample settings.



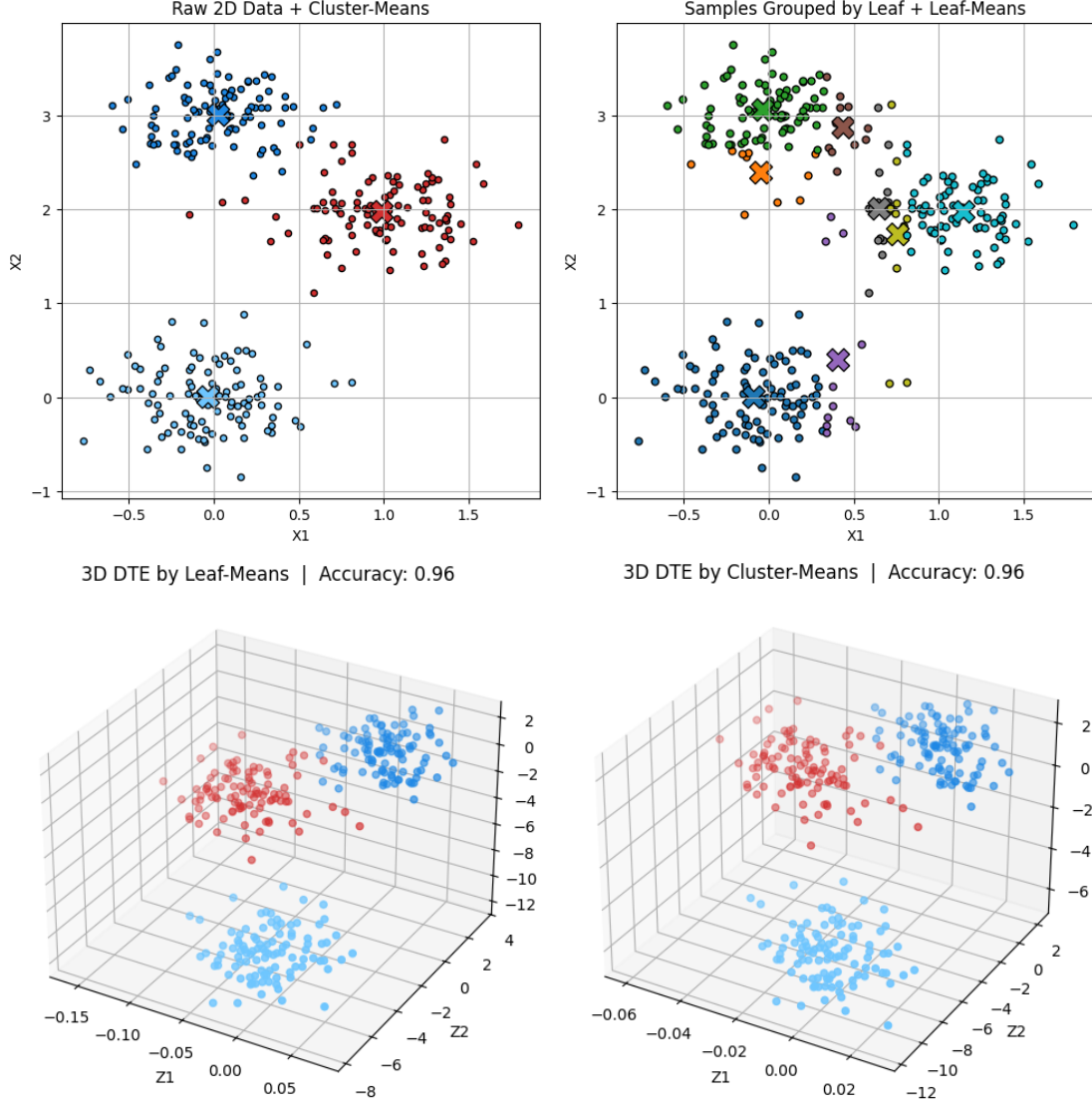


Fig. 1. Top left: simulated training data in the original two-dimensional space. The two clusters of class 1 are shown in light and dark blue, respectively, while class 2 is shown in red. Empirical cluster means are indicated by bolded markers in matching colors. Top right: leaf assignments and leaf means of the fitted decision tree. Bottom left: the three-dimensional DTE embedding  $\mathbf{Z}$  obtained using the tree-derived leaf means. Samples are colored by their true cluster labels. Bottom right: the oracle embedding  $\mathbf{Z}^*$ , obtained by replacing the leaf means with the true cluster means.

#### 4 SIMULATION

In this section, we consider a simple simulation setting to provide visual intuition for how DTE works. We generate synthetic two-dimensional data drawn from two classes with three total clusters. For class 1, the data are generated equally like from either of the following two distributions:

$$X_{1a} \sim \mathcal{N}\left(\begin{bmatrix} 0 \\ 0 \end{bmatrix}, 0.3^2 I_2\right), \quad X_{1b} \sim \mathcal{N}\left(\begin{bmatrix} 0 \\ 3 \end{bmatrix}, 0.3^2 I_2\right).$$

For class 2, the data are generated from a single distribution:

$$X_2 \sim \mathcal{N}\left(\begin{bmatrix} 1 \\ 2 \end{bmatrix}, 0.3^2 I_2\right).$$

The class priors are  $\pi_1 = 2/3$  and  $\pi_2 = 1/3$ . This mixture model has a Bayes-optimal accuracy of approximately 0.993.

A training set  $(\mathbf{X}, \mathbf{Y})$  of size  $n = 100$  is sampled from this model. The top left panel of Figure 1 visualizes the training data: light and dark blue points correspond to the

two clusters in class 1, and red points correspond to class 2. The empirical cluster means are shown as well. We then fit a single classification decision tree using a minimum leaf size of 10. The top right panel of Figure 1 colors each sample according to its leaf assignment and marks the corresponding leaf means.

Using this fitted tree, we compute the associated decision tree embedding. The matrix  $\mathbf{W}$  contains the leaf means, and the embedding  $\mathbf{Z}$  is 8-dimensional because the tree has eight leaf regions. The first three coordinates of  $\mathbf{Z}$  are visualized in the bottom left panel of Figure 1, showing that the two classes are well separated by the resulting embedding.

In this simulation, the ideal separating regions can be viewed as the true generating clusters. Thus, we also construct a hypothetical oracle embedding  $\mathbf{Z}^*$ , obtained by applying the same algorithm but replacing the leaf means with the true cluster means, which serve as optimal anchor points. The resulting embedding is displayed in the bottom

right panel of Figure 1. The close visual agreement between the two bottom panels indicates that the single-tree DTE closely approximates the geometry of the oracle embedding. In particular, although the split rules themselves exhibit high variance relative to the optimal split, the resulting anchor points and the induced embedding exhibit very low variance.

To further assess separation quality, we compute the classification error using DTE with LDA on a test set of size 100 generated from the same model. We perform this both for the actual embedding  $\mathbf{Z}$  and the oracle embedding  $\mathbf{Z}^*$ , respectively. Our method yields a remarkably well-separated embedding: an LDA classifier trained on  $(\mathbf{Z}, \mathbf{Y})$  achieves an accuracy of 0.96, which is identical to that obtained from  $(\mathbf{Z}^*, \mathbf{Y})$ .

This shows that the leaf means serve as effective anchor points for a geometry-preserving embedding, and that the resulting decision tree embedding yields high-quality representations even with a single tree. Although the tree-induced partition deviate significantly from the oracle cluster structure, the DTE method remains well-behaved and robust to sampling variability in the tree partitions.

## 5 REAL DATA

Table 1 compares DTE with single decision tree, random forest, and a two-layer neural network on 20 public datasets from the UCI Machine Learning Repository [28], Kaggle, and MATLAB built-ins. The datasets span diverse domains with varying sizes, from text to grayscale images to standard feature data. Categorical features are converted via one-hot encoding.

All experiments were conducted on a standard desktop (6-core Intel i7 CPU, 64GB RAM) using MATLAB 2024a, with no GPU or parallelization. We used various MATLAB built-in functions with default parameters in most cases: For DTE-1, we built a single tree using `fitctree` (`numBins` = 30, `minLeafSize` = 10) and applied LDA using `fitcdiscr` with pseudolinear discriminant type. DTE-3 uses two additional trees built on independent bootstrap samples. The baseline decision tree uses `fitctree` with default settings. Random forest use `TreeBagger` with 50 trees, and the neural network uses `fitcnet` with 100 hidden neurons and standardization enabled.

Classification errors were averaged over 10 replicates of 5-fold splits (80% train / 20% test). The source and size of each dataset are listed in Table 1. The main findings are as follows:

- DTE-1 vs. decision tree: DTE-1 is equal or better on 19/20 datasets, and significantly better on 16/20.
- DTE-3 vs. DTE-1: DTE-3 matches or improves upon DTE-1 on all datasets and is significantly better on 6/20.
- DTE-3 vs. neural network: DTE-3 is better in 13/20 cases, with 9 significant.
- DTE-3 vs. random forest: DTE-3 is significantly better on 5 datasets, while random forest is significantly better on 6; otherwise performance is comparable.

Here, we deem a method significantly better when the difference in error rates exceeds three standard deviations, using the larger standard deviation of the two methods.

Figure 2 reports running times. Across all datasets, DTE-1 and the single decision tree are the fastest methods because they build only one tree. DTE-3 is roughly three times slower (since tree building is repeated without parallelization), but still substantially faster than random forest. Despite MATLAB’s optimized implementation, the 50-tree random forest is about 9× slower than DTE-1 and 4× slower than DTE-3 on median. The neural network is typically faster than the forest but slower than DTE-3 (about 3× on median), with occasional slow convergence. As noted in the complexity analysis, DTE by LDA can be slower when the number of leaf regions  $m$  is too large, as seen in the FacePIE dataset, due to the quadratic complexity of LDA in  $m$ .

Overall, the results highlight the advantages of the proposed method. DTE-1 is consistently more accurate than a single decision tree, even though it uses the same tree. DTE-3 often yields additional gains while remaining efficient. DTE is competitive with random forest and shallow neural networks in accuracy, while offering substantially lower computation time in most cases.

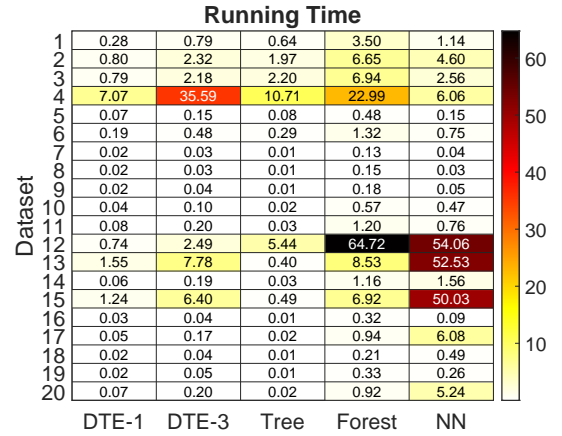


Fig. 2. This figure shows the running time, in seconds, for each method and each dataset, including training and testing. We report the average running time for each dataset, averaged over all cross validation runs.

## 6 CONCLUSION

In this paper, we introduce the decision tree embedding, which can be viewed as a combination of neural networks and decision trees, inheriting advantages from both. The method performs a linear transformation and offers scalable inference like a neural network, while its weights are derived from tree-based leaf means, providing strong interpretability and an adaptive, data-driven choice of hidden-layer size without relying on back-propagation. We provide theoretical characterization, simulation-based visualizations illustrating its mechanism, and numerical experiments demonstrating its effectiveness.

There are several promising directions for further development. At present, classification performance may deteriorate when  $t$  becomes large, as the resulting increase in dimension  $m$  leads to higher estimation variance in LDA. This suggests an opportunity to introduce an additional dimension reduction layer before classification, potentially allowing the method to consistently benefit from larger

Data	$(n, p, K, m)$	DTE-1	DTE-3	Tree	Forest	S-NN
Cora [29]	(2708,1433,7,79)	27.3 ± 0.5	25.6 ± 0.3	34.9 ± 0.6	<b>24.6</b> ± 0.3	25.2 ± 0.5
Citeseer [30]	(3312,3703,6,104)	29.9 ± 0.3	28.7 ± 0.3	39.9 ± 0.5	<b>27.9</b> ± 0.4	28.0 ± 0.6
Isolet [31]	(7797,617,26,140)	5.9 ± 0.1	5.5 ± 0.1	19.6 ± 0.3	6.0 ± 0.1	<b>4.4</b> ± 0.2
FacePIE [32]	(11554,1024,68,681)	5.2 ± 0.1	5.2 ± 0.1	45.3 ± 0.5	3.0 ± 0.1	<b>2.9</b> ± 0.1
FaceYale 32*32 [33]	(165,1024,15,14)	30.5 ± 1.7	<b>18.3</b> ± 1.3	54.4 ± 3.0	24.3 ± 1.4	23.1 ± 1.8
FaceYale 64*64 [33]	(165,4096,15,15)	20.8 ± 1.5	<b>10.7</b> ± 1.7	37.2 ± 4.2	19.3 ± 1.7	18.6 ± 1.9
Iris [34]	(150,4,3,3)	<b>1.9</b> ± 0.3	2.1 ± 0.2	5.2 ± 1.1	5.0 ± 0.7	5.5 ± 1.6
Wine [35]	(178,13,3,4)	8.2 ± 1.2	2.4 ± 0.7	10.3 ± 1.2	<b>2.0</b> ± 0.6	2.4 ± 0.6
Wisc Cancer [36]	(699,9,2,6)	4.2 ± 0.2	4.2 ± 0.1	6.0 ± 0.5	<b>3.3</b> ± 0.3	4.9 ± 0.5
Colon [37]	(62,2000,2,3)	13.0 ± 1.4	<b>12.0</b> ± 1.0	27.1 ± 3.9	18.9 ± 4.5	21.4 ± 4.2
Golub [38]	(72,7129,3,3)	8.7 ± 2.4	6.6 ± 1.8	9.9 ± 2.3	<b>6.5</b> ± 1.8	9.9 ± 4.0
Credit Card [39]	(284807,30,2,16)	0.06 ± 0.00	0.06 ± 0.00	0.08 ± 0.00	<b>0.05</b> ± 0.00	0.07 ± 0.00
Bank Marketing [40]	(45211,51,2,471)	10.0 ± 0.0	10.0 ± 0.0	11.6 ± 0.1	<b>9.5</b> ± 0.1	11.9 ± 0.2
Student Success [41]	(4424,36,3,120)	24.1 ± 0.1	24.1 ± 0.1	30.3 ± 0.6	<b>23.0</b> ± 0.4	30.6 ± 0.4
Adult Income [42]	(32561,108,2,483)	16.6 ± 0.1	16.6 ± 0.1	17.2 ± 0.1	<b>13.5</b> ± 0.1	18.8 ± 0.1
German Credit	(1000,24,2,30)	23.6 ± 0.4	<b>23.4</b> ± 0.4	30.4 ± 1.2	23.5 ± 0.6	29.6 ± 0.8
Abalone	(4177,10,2,94)	22.4 ± 0.1	22.4 ± 0.1	26.8 ± 0.5	<b>21.3</b> ± 0.4	24.6 ± 0.7
Wholesale	(440,7,1,17)	28.5 ± 0.2	<b>28.4</b> ± 0.3	41.6 ± 2.4	30.4 ± 0.4	45.4 ± 1.8
Wine White [43]	(4898,11,2,29)	12.4 ± 0.1	12.4 ± 0.1	12.4 ± 0.4	<b>8.9</b> ± 0.1	11.1 ± 0.6
Wine Red [43]	(1599,11,2,135)	19.8 ± 0.1	19.8 ± 0.1	18.5 ± 0.5	<b>12.0</b> ± 0.2	15.4 ± 0.5

TABLE 1

Average classification error rates (in percentage) with standard deviations across all real data experiments. For each experiment, the best-performing method is highlighted in bold. The number of leaf regions  $m$  are based on a single decision tree on the full data.

ensembles, or to employ an alternative classifier that is less susceptible to increases in dimensionality. Another direction is to explore trees built using alternative criteria, such as regression trees, other impurity measures, or structured splitting rules. Finally, because the formulation closely resembles a neural network, the method could be incorporated into more complex architectures or applied to domain-specific tasks.

## REFERENCES

- [1] J. R. Quinlan, *C4.5: Programs for Machine Learning*. San Mateo, CA: Morgan Kaufmann, 1993. 1
- [2] L. Breiman, J. Friedman, R. Olshen, and C. Stone, *Classification and Regression Trees*. Belmont, CA: Wadsworth International Group, 1984. 1
- [3] L. Breiman, “Bagging predictors,” *Machine Learning*, vol. 24, no. 2, pp. 123–140, 1996. 1
- [4] —, “Random forests,” *Machine Learning*, vol. 45, no. 1, pp. 5–32, 2001. 1
- [5] T. M. Tomita, J. Browne, C. Shen, J. Chung, J. L. Patsolic, B. Falk, J. Yim, C. E. Priebe, R. Burns, M. Maggioni, and J. T. Vogelstein, “Sparse projection oblique random forests,” *Journal of Machine Learning Research*, vol. 21, no. 104, pp. 1–39, 2020. 1
- [6] J. J. Rodriguez, L. I. Kuncheva, and C. J. Alonso, “Rotation forest: A new classifier ensemble method,” *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 28, no. 10, pp. 1619–1630, 2006. 1
- [7] B. H. Menze, B. M. Kelm, R. Masuch, U. Himmelreich, and F. A. Hamprecht, “On oblique random forests,” in *Proceedings of the Joint DAGM and OAGM Symposium*. Springer, 2011, pp. 453–462. 1
- [8] N. Srivastava, G. E. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov, “Dropout: A simple way to prevent neural networks from overfitting,” *Journal of Machine Learning Research*, vol. 15, pp. 1929–1958, 2014. 1
- [9] K. He, X. Zhang, S. Ren, and J. Sun, “Deep residual learning for image recognition,” in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016, pp. 770–778. 1
- [10] K. Hornik, “Approximation capabilities of multilayer feedforward networks,” *Neural Networks*, vol. 4, no. 2, pp. 251–257, 1991. 1
- [11] Y. LeCun, Y. Bengio, and G. Hinton, “Deep learning,” *Nature*, vol. 521, no. 7553, pp. 436–444, 2015. 1
- [12] I. Goodfellow, Y. Bengio, and A. Courville, *Deep Learning*. MIT Press, 2016. 1
- [13] Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner, “Gradient-based learning applied to document recognition,” *Proceedings of the IEEE*, vol. 86, no. 11, pp. 2278–2324, 1998. 1
- [14] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, L. Kaiser, and I. Polosukhin, “Attention is all you need,” in *Advances in Neural Information Processing Systems (NeurIPS)*, 2017, pp. 5998–6008. 1
- [15] R. Caruana and A. Niculescu-Mizil, “An empirical comparison of supervised learning algorithms,” in *Proceedings of the 23rd International Conference on Machine Learning (ICML)*. ACM, 2006, pp. 161–168. 1
- [16] M. Fernández-Delgado, E. Cernadas, S. Barro, and D. Amorim, “Do we need hundreds of classifiers to solve real world classification problems?” *Journal of Machine Learning Research*, vol. 15, no. 1, pp. 3133–3181, 2014. 1
- [17] L. Grinsztajn, E. Oyallon, and G. Varoquaux, “Why do tree-based models still outperform deep learning on typical tabular data?” in *Advances in Neural Information Processing Systems (NeurIPS)*, 2022. 1
- [18] D. R. Ferreira, A. E. Lazzaletti, and A. S. Ferreira Junior, “Computational and memory cost analysis of ensemble tree classifiers,” *IEEE Access*, vol. 9, pp. 121–136, 2021. 1
- [19] X. He, J. Pan, O. Jin, T. Xu, B. Liu, T. Xu, Y. Shi, A. Atallah, R. Herbrich, S. Bowers, and J. Q. Candela, “Practical lessons from predicting clicks on ads at facebook,” in *Proceedings of the 20th ACM Conference on Knowledge Discovery and Data Mining (KDD)*, 2014. 2
- [20] P. Kotschieder, M. Fiterau, A. Criminisi, and S. Rota Buló, “Deep neural decision forests,” in *Proceedings of the IEEE International Conference on Computer Vision (ICCV)*, 2015. 2
- [21] S. Wang, J. Tang, and H. Liu, “Using a random forest to inspire a neural network and improving on it,” in *IEEE International Conference on Data Mining Workshops (ICDMW)*. IEEE, 2016, pp. 1049–1056. 2
- [22] G. Biau, E. Scornet, and J. Welbl, “Neural random forests,” *Statistical Analysis and Data Mining: The ASA Data Science Journal*, vol. 12, no. 5, pp. 397–407, 2019. 2
- [23] O. Togunwa, J. Afolabi, and S. Fasina, “A deep hybrid model for maternal health risk classification,” *Frontiers in Artificial Intelligence*, vol. 6, p. 1213436, 2023. 2
- [24] N. Konstantinov, D. Grigoryev, and A. Babenko, “Neural attention forests,” *arXiv preprint arXiv:2304.05980*, 2023. 2
- [25] Y. Qiu, M. Xu, and B. Yu, “Neural networks meet random forests,” *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, vol. 86, no. 5, pp. 1435–1460, 2024. 2
- [26] S. Nickzamid and S. A. Gandab, “Hybrid random forest and convolutional neural network framework for hyperspectral image classification,” *arXiv preprint arXiv:2502.00232*, 2025. 2

- [27] C. Shen and Y. Dong, "A graph sufficiency perspective for neural networks," *arXiv preprint arXiv:2507.10215*, 2026. 3
- [28] M. Kelly, R. Longjohn, and K. Nottingham, "The uci machine learning repository," <https://archive.ics.uci.edu>, 2023. 6
- [29] A. K. McCallum, K. Nigam, J. Rennie, and K. Seymore, "Automating the construction of internet portals with machine learning," *Information Retrieval*, vol. 3, pp. 127–163, 2000. 7
- [30] C. L. Giles, K. D. Bollacker, and S. Lawrence, "Citeseer: An automatic citation indexing system," in *Proceedings of the Third ACM Conference on Digital Libraries*, 1998, pp. 89–98. 7
- [31] M. Fanty and R. Cole, "Spoken letter recognition," in *Advances in Neural Information Processing Systems*, vol. 3, 1991. 7
- [32] T. Sim, S. Baker, and M. Bsat, "The CMU pose, illumination, and expression database," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 25, no. 12, pp. 1615–1618, 2003. 7
- [33] P. N. Belhumeur, J. P. Hespanha, and D. Kriegman, "Eigenfaces vs. fisherfaces: Recognition using class specific linear projection," *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, vol. 19, no. 7, pp. 711–720, 1997. 7
- [34] R. A. Fisher, "The use of multiple measurements in taxonomic problems," *Annals of Eugenics*, vol. 7, no. 2, pp. 179–188, 1936. 7
- [35] S. Aeberhard, D. Coomans, and O. de Vel, "Comparative analysis of statistical pattern recognition methods in high dimensional settings," *Pattern Recognition*, vol. 27, no. 8, pp. 1065–1077, 1994. 7
- [36] W. Street, W. Wolberg, and O. Mangasarian, "Nuclear feature extraction for breast tumor diagnosis," in *Proceedings of SPIE - The International Society for Optical Engineering*, 1993. 7
- [37] U. Alon, N. Barkai, D. A. Notterman, K. Gish, S. Ybarra, D. Mack, and A. J. Levine, "Broad patterns of gene expression revealed by clustering analysis of tumor and normal colon tissues probed by oligonucleotide arrays," *Proceedings of the National Academy of Sciences*, vol. 96, no. 12, pp. 6745–6750, 1999. 7
- [38] T. R. Golub, D. K. Slonim, P. Tamayo, C. Huard, M. Gaasenbeek, J. P. Mesirov, H. Coller, M. L. Loh, J. R. Downing, M. A. Caligiuri, C. D. Bloomfield, and E. S. Lander, "Molecular classification of cancer: class discovery and class prediction by gene expression monitoring," *Science*, vol. 286, no. 5439, pp. 531–537, 1999. 7
- [39] A. Dal Pozzolo, G. Boracchi, O. Caelen, C. Alippi, and G. Bontemp, "Credit card fraud detection: a realistic modeling and a novel learning strategy," *IEEE Transactions on Neural Networks and Learning Systems*, vol. 29, no. 8, pp. 3784–3797, 2018. 7
- [40] S. Moro, R. Laureano, and P. Cortez, "Using data mining for bank direct marketing: An application of the CRISP-DM methodology," in *Proceedings of the European Simulation and Modelling Conference*, 2011, pp. 117–121. 7
- [41] M. V. Martins, D. Tolledo, J. Machado, L. M. T. Baptista, and V. Realinho, "Early prediction of student's performance in higher education: A case study," in *Trends and Applications in Information Systems and Technologies*, vol. 1365. Springer, 2021. 7
- [42] R. Kohavi, "Scaling up the accuracy of naive-bayes classifiers: a decision-tree hybrid," in *Proceedings of the Second International Conference on Knowledge Discovery and Data Mining*, 1996, pp. 202–207. 7
- [43] P. Cortez, A. Cerdeira, F. Almeida, T. Matos, and J. Reis, "Modeling wine preferences by data mining from physicochemical properties," *Decision Support Systems*, vol. 47, no. 4, pp. 547–553, 2009. 7



## APPENDIX

### 7 PROOFS

**Theorem 1.** Let  $(X, Y)$  be a random pair with  $X \in \mathbb{R}^p$  and  $Y \in [K]$ . Let  $\{\mathcal{R}_1, \dots, \mathcal{R}_m\}$  be the set of leaf regions from a classification tree, and let  $Z = X\mathbf{W}^\top + \mathbf{1b}^\top$  denote its Decision Tree Embedding, where

$$\mu_j = \mathbb{E}[X \mid X \in \mathcal{R}_j], \quad \mathbf{b}_j = -\|\mu_j\|^2, \quad \mathbf{W} = [\mu_1^\top, \dots, \mu_m^\top]^\top.$$

If the partition  $\{\mathcal{R}_j\}$  is  $\varepsilon$ -Bayes-homogeneous, then

$$\|P(Y \mid X) - P(Y \mid Z)\|_1 \leq \varepsilon.$$

In particular, when  $\varepsilon = 0$ , the DTE embedding is sufficient for  $Y$ ; that is,

$$P(Y \mid X) = P(Y \mid Z) \quad \text{a.s.}$$

*Proof.* Let  $\{\mathcal{R}_1, \dots, \mathcal{R}_m\}$  be the leaf partition of  $\mathcal{X}$ . The map  $X \mapsto \mathcal{R}(X) \in \{1, \dots, m\}$  is measurable with respect to the  $\sigma$ -field generated by the partition, i.e.

$$\sigma(\mathcal{R}(X)) = \sigma(\{X \in \mathcal{R}_j\}_{j=1}^m).$$

By definition of the decision tree embedding,  $Z$  is a measurable function of  $\mathcal{R}(X)$ ; hence the generated  $\sigma$ -fields coincide:

$$\sigma(Z) = \sigma(\mathcal{R}(X)).$$

Therefore,

$$P(Y \mid Z) = P(Y \mid \mathcal{R}(X)). \quad (1)$$

For each  $j$ , define

$$P(Y = c \mid X \in \mathcal{R}_j) := \frac{P(Y = c, X \in \mathcal{R}_j)}{P(X \in \mathcal{R}_j)}.$$

Bayes  $\varepsilon$ -homogeneity means that for all  $x, x' \in \mathcal{R}_j$ ,

$$\|P(Y \mid X = x) - P(Y \mid X = x')\|_1 \leq \varepsilon_j,$$

and define the global bound  $\varepsilon = \max_j \varepsilon_j$ .

Fix any  $x \in \mathcal{R}_j$ . Since  $P(Y \mid \mathcal{R}_j)$  is the average of  $P(Y \mid X = x')$  over  $x' \in \mathcal{R}_j$ , convexity of the  $L^1$  norm implies

$$\|P(Y \mid X = x) - P(Y \mid \mathcal{R}_j)\|_1 \leq \varepsilon_j \leq \varepsilon.$$

Using Equation 1 and that  $\mathcal{R}(X) = j$  if and only if  $X \in \mathcal{R}_j$ , we have almost surely

$$\|P(Y \mid X) - P(Y \mid Z)\|_1 = \|P(Y \mid X) - P(Y \mid \mathcal{R}(X))\|_1 \leq \varepsilon.$$

When  $\varepsilon = 0$ , the conditional distribution  $P(Y \mid X = x)$  is constant on each  $\mathcal{R}_j$ , and thus  $P(Y \mid X) = P(Y \mid Z)$  almost surely. Hence  $Z$  is a sufficient statistic for  $Y$  relative to  $X$ .  $\square$

**Theorem 2.** Under the same notation as Theorem 1, and further assume that each point in a leaf region  $\mathcal{R}_j$  is closer (in Euclidean norm) to its own region mean  $\mu_j$  than to any other region mean  $\mu_{j'}$ .

For each class  $c \in [K]$ , let

$$\mathcal{J}_c = \{j : \arg \max_{c'} P(Y = c' \mid X \in \mathcal{R}_j) = c\}$$

be the index set of leaves whose majority label is  $c$ . We then consider the indicator classifier on the DTE embedding:

$$g(Z) = \arg \max_{c \in \{1, \dots, C\}} \gamma_c^\top Z, \quad (\gamma_c)_j = \begin{cases} 1, & j \in \mathcal{J}_c, \\ 0, & \text{otherwise.} \end{cases}$$

For each  $j$ , let

$$l_j = 1 - \max_c P(Y = c \mid X \in \mathcal{R}_j)$$

be the impurity level at each region. Then the classification error of the indicator rule  $g$  equals

$$L_g = \sum_{j=1}^m P(X \in \mathcal{R}_j) l_j.$$

In particular, if every leaf region is pure, we have  $l_j = 0$  and  $L_g = 0$ , such that the DTE embedding achieves perfect classification with the indicator rule.

*Proof.* Fix any  $x \in \mathcal{R}_j$ . Its DTE embedding is given by

$$Z_{j'}(x) = x^\top \mu_{j'} - \|\mu_{j'}\|^2, \quad j' = 1, \dots, m.$$

We first show that the  $j$ th coordinate is the maximal one. Using the identity

$$x^\top \mu_{j'} - \|\mu_{j'}\|^2 = -\frac{1}{2}\|x - \mu_{j'}\|^2 + \frac{1}{2}\|x\|^2,$$

we obtain

$$Z_j(x) - Z_{j'}(x) = \frac{1}{2}(\|x - \mu_{j'}\|^2 - \|x - \mu_j\|^2). \quad (2)$$

Since  $\mu_j$  is the mean of the leaf  $\mathcal{R}_j$ , all points  $x \in \mathcal{R}_j$  satisfy

$$\|x - \mu_j\|^2 \leq \|x - \mu_{j'}\|^2, \quad \forall j' \neq j, \quad (3)$$

because each leaf of a decision tree corresponds to the region of points split so as to be closest (in squared Euclidean distance) to its own mean. Combining Equation 2 and 3 yields

$$Z_j(x) \geq Z_{j'}(x) \quad \forall j' \neq j. \quad (4)$$

Now consider the indicator classifier

$$g(Z) = \arg \max_{c \in [K]} \gamma_c^\top Z, \quad (\gamma_c)_j = 1_{\{j \in \mathcal{J}_c\}}. \quad (5)$$

Since  $\gamma_c^\top Z(x)$  selects the maximum among coordinates belonging to class  $c$ , and Equation 4 shows that the maximal coordinate of  $Z(x)$  is the  $j$ th one, it follows that

$$g(Z(x)) = c_j,$$

where  $c_j$  is the majority class in region  $\mathcal{R}_j$ .

Thus,  $g$  predicts a constant label on each leaf region. The misclassification probability is therefore

$$\mathbb{P}(g(Z(X)) \neq Y) = \sum_{j=1}^m P(X \in \mathcal{R}_j) P(Y \neq c_j \mid X \in \mathcal{R}_j). \quad (6)$$

By definition of the impurity

$$l_j = 1 - \max_c P(Y = c \mid X \in \mathcal{R}_j) = P(Y \neq c_j \mid X \in \mathcal{R}_j),$$

and substituting into Equation 6 gives

$$L_g = \sum_{j=1}^m P(X \in \mathcal{R}_j) l_j.$$

If each leaf is pure, then  $l_j = 0$  for all  $j$ , so  $L_g = 0$ , and the DTE embedding is perfectly linearly separable under the indicator classifier.  $\square$