SN Computer Science (2022) 3:313
https://doi.org/10.1007/s42979-022-01147-4

ORIGINAL RESEARCH

A Better Decision Tree: The Max-Cut Decision Tree with Modified PCA Improves Accuracy and Running Time

Jonathan Bodine[1] · Dorit S. Hochbaum[1]

Received: 16 July 2021 / Accepted: 11 April 2022

Abstract

Decision trees are a widely used method for classification, both alone and as the building blocks of multiple different ensemble learning methods. The Max Cut decision tree introduced here involves novel modifications to a standard, baseline variant of a classification decision tree, CART Gini. One modification involves an alternative splitting metric, Max Cut, based on maximizing the distance between all pairs of observations that belong to separate classes and separate sides of the threshold value. The other modification, Node Means PCA, selects the decision feature from a linear combination of the input features constructed using an adjustment to principal component analysis (PCA) locally at each node. Our experiments show that this node-based, localized PCA with the Max Cut splitting metric can dramatically improve classification accuracy while also significantly decreasing computational time compared to the CART Gini decision tree. These improvements are most significant for higher-dimensional datasets. For the example dataset CIFAR-100, the modifications enabled a 49% improvement in accuracy, relative to CART Gini, while providing a 6.8× speed up compared to the Scikit-Learn implementation of CART Gini. These introduced modifications are expected to dramatically advance the capabilities of decision trees for difficult classification tasks.

Keywords Decision tree · Principal component analysis · Maximum cut · Classification

Introduction

One of the major interests of machine learning and data science, in general, is developing methods that can correctly classify new observations. The decision tree is a widely used method for solving these classification tasks, favored for its simplicity and ease of use. The importance of decision trees is further increased by their use as the building block of even more advanced ensemble methods, such as the Random Forest [3]. At their core, decision trees are a series of branching questions, *nodes*, that eventually lead to a specific conclusion—in our case, what class an observation belongs to. The end goal for constructing decision trees is to understand what these branching rules should be, and what conclusion they lead to.

A popular implementation for constructing decision trees is the Classification and Regression Trees (CART) method [4]. CART's popularity is in part due to readily available commercial-grade open-source implementations, such as that provided by Scikit-Learn [16]. In general, the CART methodology works by selecting the feature and respective threshold that is "best" according to a given splitting criterion. Specifically, the *threshold* is the value of a given feature *i*, such that the decision tree asks the question 'is the value of feature *i* less than the threshold value' to separate the observations into two sets (yes/no); and the *splitting criterion* is the rule for evaluating a given feature and respective threshold. The CART methodology proceeds to find the feature and respective threshold for each of the two sets it just defined; this recursive pattern continues until every node is "pure" - that is, it contains a set of samples that all

The authors were supported in part by NSF award No. CMMI-1760102. The second author was also supported in part by AI Institute NSF award 2112533.

This article is part of the topical collection "Knowledge Discovery, Knowledge Engineering and Knowledge Management" guest edited by Joaquim Filipe, Ana Fred, Jan Dietz, Ana Salgado and Jorge Bernardino.

✉ Jonathan Bodine
jonathan.m.bodine@berkeley.edu

Dorit S. Hochbaum
hochbaum@ieor.berkeley.edu

[1] University of California, Berkeley, USA

Published online: 31 May 2022

belong to a single class. This paper explores modifications to two aspects of the CART variant that ultimately lead to the discovery of a decision tree variant with favorable accuracy and running time properties.

The first area of modification on the baseline decision tree is in regards to the splitting criterion. Our splitting procedure addresses a shortcoming of those traditionally used by CART: the partitions generated by the traditional splitting criterions, such as the Gini criterion (see Sect. "Splitting Criterion: Max Cut" for details on the Gini criterion), depend only on separating the different classes while ignoring how similar these observations are in the space of their feature vectors. We believe that this similarity information is meaningful as it can be used to produce thresholds that separate the classes in a way that also separates observations that are dissimilar, resulting in decreased susceptibility to overfitting. Our **Max Cut** splitting criterion takes into account both class membership and feature similarity when deciding on the optimal split (see Sect. "Splitting Criterion: Max Cut" for details).

The second area of modification is in how the features are represented. If the feature vector $\mathbf{x}_i$ is modified through a change of basis, the splits of the decision tree can be changed; therefore, this can have a significant impact on the performance of a decision tree. Here we explore bases alternatives for representing the feature vector. One way to find a natural basis to represent the feature vector is Principal Component Analysis (PCA), first proposed by Karl Pearson F.R.S. [7]. PCA iteratively finds the direction that accounts for the most variance in the data, given that it is orthogonal to all the directions previously found. The use of PCA has been previously studied in conjunction with CART, e.g. [13, 14]; however, prior research focuses on the use of PCA as a dimensionality reducing preprocessing step. We advance on this by proposing a modified use of PCA, throughout the construction of the decision tree, at every node. This is motivated by the idea that distributions could vary in different subspaces of the feature space. Therefore, a meaningful direction overall might not be significant in specific sub-spaces. To account for this, we introduce the following two methods for finding locally meaningful directions **Node Means PCA** and **Node Features PCA** (see Sect. "Feature Representation" for details). Ultimately, after analyzing the two different feature representations, Node Means PCA, which focuses on representing the differences in the classes, demonstrates significant increases in accuracy for high dimensional problems while also decreasing their corresponding runtime.

The benefits of the Max Cut splitting criterion along with Node Means PCA are demonstrated here via an extensive prototypical experimental comparison of our novel **Max Cut Node Means PCA** variant, which combines the Max Cut splitting criterion and the Node Means PCA feature representation, to seven other variants using different combinations of splitting criterions and feature representations. Using python prototypes of the different variants, an experimental comparison is presented here covering more than 20,000 synthetic datasets (Sect. "Synthetic Datasets") with training set sizes ranging between 100 and 300,000 and testing sets fixed at 100,000 observations. First binary classification problems (Sect. "Binary Classification") are considered, followed by 10-class classification problems (Sect. "Multiclass Classification"). Using the synthetic data, the improvements to the baseline CART variant are shown to be statistically significant. Similarly, results are attained for real-world datasets (Sect. "Real-World Datasets"). A preliminary version of these results was given in an earlier version of this paper [2]. These preliminary results demonstrated dramatic improvements in both accuracy and computational time for the Max Cut Node Means PCA variant, with these advantages further pronounced with the dimensionality of the data.

Due to the success in improving accuracy and run time shown for the prototype implementation, we implemented a more advanced implementation of the Max Cut Node Means PCA variant, 'MaxCutTree.' This MaxCutTree implementation is shown here to achieve similar improvements in run times against a commercial-quality implementation of the baseline CART variant, Scikit-Learn. This is tested on 45,000 additional synthetic datasets whose dimensions vary between 51 and 200, with the results replicating those for the prototype—namely increased accuracy and decreased running time for high dimensional datasets. The significance of these results is demonstrated here with the CIFAR-100 dataset [11], which has 100 classes, 3,072 features, and 60,000 total observations (of which 48,000 are used for training). In this case, our MaxCutTree implementation resulted in a 49.4% increase in accuracy relative to the baseline CART while simultaneously providing a 6.8× speed up compared to the commercial-quality implementation.

Description of Variants

Splitting Criterion: Max Cut

Our splitting procedure, Max Cut, takes inspiration from the Maximum Cut problem, which in general is NP-complete [10]. Our use of Maximum Cut is in one dimension, based on distances with respect to a single feature, and restricted to a single threshold, which is polynomial-time solvable. Indeed, in the appendix, we show a linear-time implementation given the sorted observations. The Max Cut criterion is to find the threshold θ maximizing the expression in (1), where $x_{i,j}$ represents the value of feature j for observation i, y_i represent the class of observation i, and $\mathbb{1}_{y_i \neq y_k}$ is equal to 1 if $y_i \neq y_k$ and 0 otherwise.

$$\sum_{\{i|x_{i,j}\leq\theta\}} \sum_{\{k|x_{k,j}>\theta\}} \mathbb{1}_{y_i \neq y_k} \left| x_{i,j} - x_{k,j} \right| \tag{1}$$

Intuitively, the Max Cut criterion finds a partition such that observations in each set are far from the observations belonging to the other set and different classes. Given that the observations can be sorted based on any feature in $O(n \log n)$ and the linear-time implementation of Max Cut, we have an $O(n \log n)$ algorithm for solving this problem on any given feature, where n is the number of observations. It is important to note that this is the same asymptotic time complexity achieved for traditional splitting criterion.

A traditional splitting criterion is the commonly used Gini Impurity, which was provided in the original CART methodology [4]. Gini Impurity is defined in equation 2, where p_c is the fraction of observations that belong to class c.

$$\sum_c p_c(1 - p_c) \tag{2}$$

The Gini Impurity splitting criterion selects the feature and respective threshold such that the average Gini Impurity of the two resulting subsets, weighted by their cardinality, is minimized. We use the Gini impurity splitting criterion as the baseline to which the Max Cut splitting criterion is compared.

Feature Representation

Node Means PCA

We introduce two methods for finding locally meaningful bases. The first method that we consider is referred to as Node Means PCA. Node Means PCA is derived from the idea that it is not the directions that best describe the data, but rather the difference between the classes, that are important. At every decision node, this procedure is performed on the original feature vectors using only the observations that have reached that node to find locally meaningful representations. The first step in this procedure is to think of the dataset in a one-vs-rest context, locally calculating the mean position of each of the possible 'rest' collections, referred to as the 'not-means.' For example, if there are three local points (1, 0, 0), (0, 1, 0), and (0, 0, 1), each belonging to classes A, B, and C respectively, then (0, 0.5, 0.5), (0.5, 0, 0.5), and (0.5, 0.5, 0) would be the resulting not-means, calculated by excluding the first, second, and third points respectively. These newly generated not-means vectors would be passed into the PCA algorithm to find a new basis, and the original features are then transformed into this new basis. It is important to note that this transformation will necessarily result in a dimensionality reduction in the case where the number of classes locally present is less than the dimension of the original feature vector; this dimensionality reduction will result in asymptotic improvements to the splitting time given some weak assumptions (specified later in this section). The features in this new basis are then used as the input features to find the optimal threshold, instead of using the original features. Pseudocode for this variant can be found in Fig. 1 below where U represents the set of all observations at a given node, X and y represent the feature vectors and class associations respectively, $C(y)$ is the set of class values in vector y, $\{i|\exists y_j = i\}$, $U_{-i} = \{u|u \in U, y_u \neq i\}$, and PCAMapping($X$) is a function that returns a mapping T from the original feature space of X to its principal components.

```
procedure meansPCA($U, X, y$):
    begin
        means $\leftarrow \{\}$;
        for each $i \in C(y)$ do
            means $\leftarrow$ means $\cup \{ \frac{1}{|U_{-i}|} \sum_{u \in U_{-i}} \mathbf{x}_u \}$;
        $T \leftarrow$ PCAMapping($means$);
        $X' \leftarrow T(X)$;
        return $X'$;
    end
```

Fig. 1 High-level description of node means PCA variant [2]

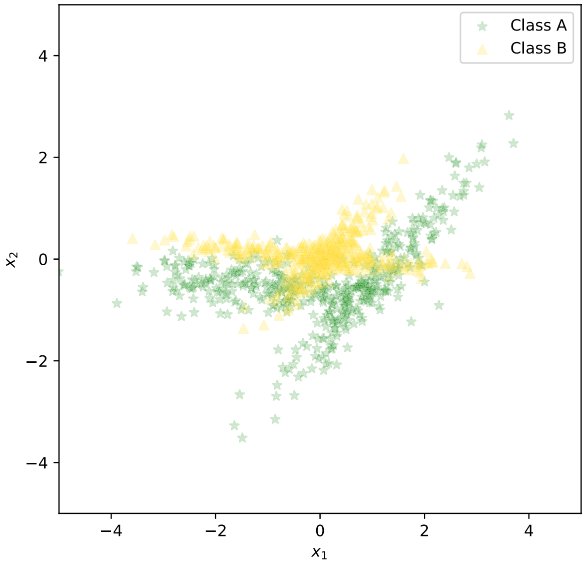

Fig. 2 Example dataset for demonstrating the variants of the algorithm

To provide a visual demonstration of how the Node Means PCA variant works we introduce the following binary

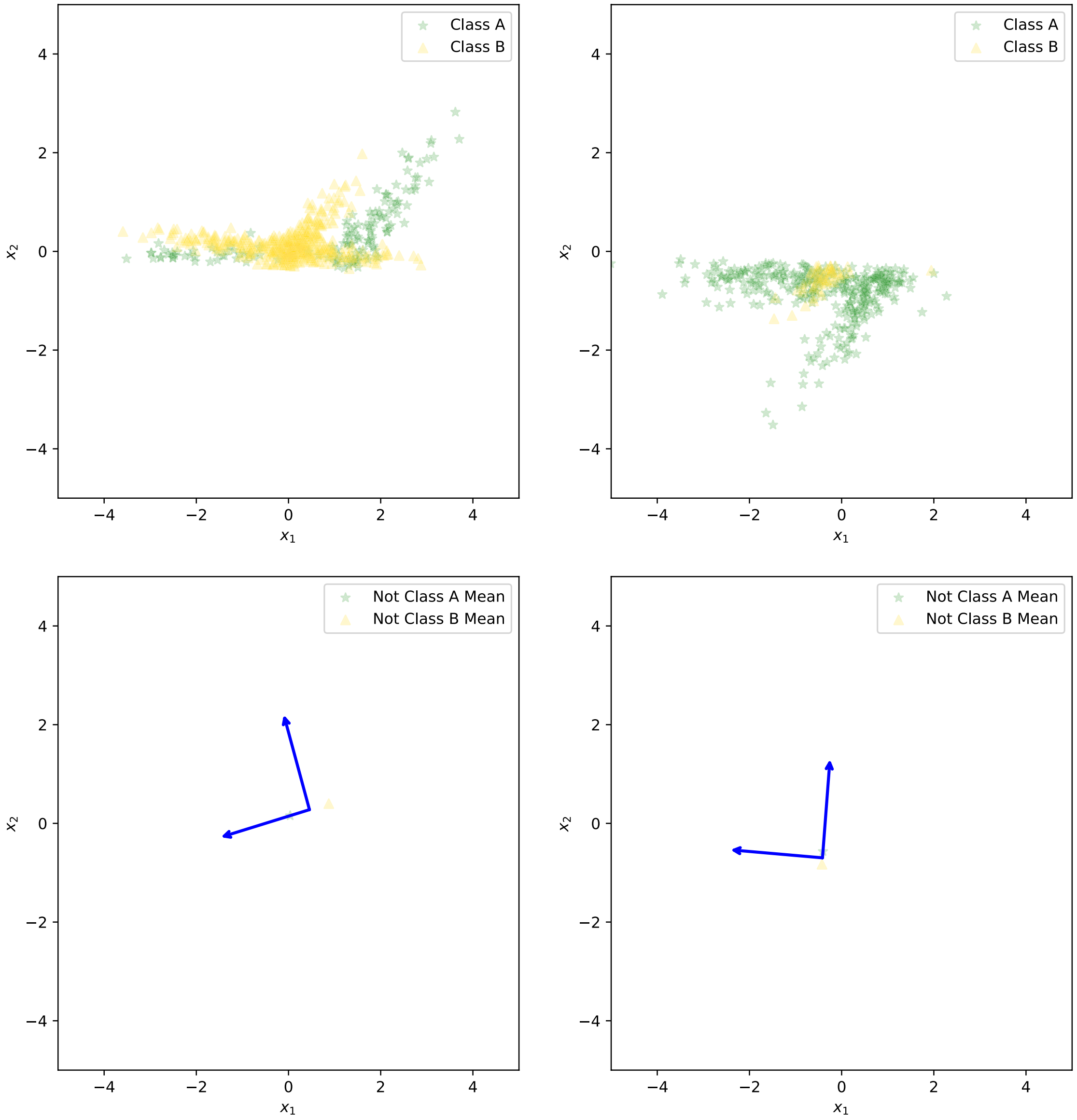

Fig. 3 Demonstration of Node Means PCA calculation on the two sub-problems of a decision tree's second level

classification dataset in two features, plotted in Fig. 2. Using the dataset in Fig. 2 and Node Means PCA in conjunction with the Max Cut splitting criterion at the second level of the decision tree one ends up with the following two local feature representations shown in Figs. 3 and 4 superimposed over the not-means and the data partition implied by the first cut respectively.

Node Features PCA

The second method that we consider is Node Features PCA. Like Node Means PCA, this variant is used at every decision node. The difference is that PCA is performed on the original feature vectors of the observations that have reached that node directly, instead of the not-means, to find the local principal components. These local principal components are then used as the input features to find the optimal threshold, rather than using the original features.

Feature Representation Comparison

We also included two traditional feature representations to understand how our modifications performed in the prototypical comparative study. The first, *Original* feature representation, was simply the feature vector as given. The second, *Pre PCA*, to show that the advantages did not come

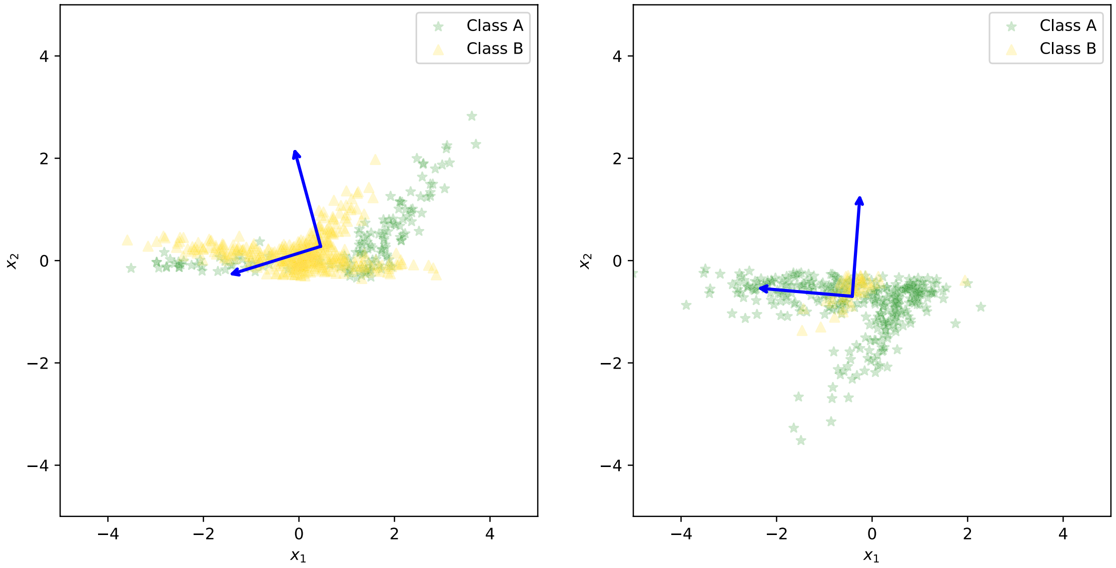

Fig. 4 Node Means PCA representation for each sub problem at depth 2

from simply using PCA, used the principal components generated on the whole training set as the feature representation.

One of the first advantages that can be noticed in regards to Node Means PCA is its asymptotic runtime. As a result of Gini Impurity and Max Cut having the same time complexity, the following results on the time complexity of each variant hold regardless of the choice of splitting criterion. The time complexities, assuming a constant number of classes, for computing the optimal split at each node are given in Table 1, where n is the number of observations, d is the number of input features, and p is the number of principal components considered. From this, we can see that so long as the dimension of the dataset is less than the number of observations, which is a fairly weak assumption, Node Means PCA will have an asymptotic computational advantage.

Decision Tree Variants: Max Cut Node Means PCA

To see the advantages of modifying the splitting criterion or the feature representation in the prototypical comparative study, all the combinations between splitting criterion (Max Cut / Gini) and feature representation (Baseline / Pre PCA / Node Features PCA / Node Means PCA) discussed in Sects. “Splitting Criterion: Max Cut” and 2.2 were studied. This produced a total of eight variants, including the Max Cut Node Means PCA variant. Table 2 provides the nomenclature used to refer to each of these different combinations.

Figure 5 provides a high-level description of a generic decision tree and how the splitting criterion and feature representation specification generates the eight variants.

Table 1 Time complexity of node splitting given a specific feature representation [2]

Feature Representation	Time Complexity
Original / Pre PCA	$O(dn \log n)$
Node Features PCA	$O(n^2 p)$
Node Means PCA	$O(d^2 + n \log n)$

The notation used in Fig. 5 is as follows: U for the set of all observations; X, y for the feature vectors and classes, respectively. Let the “split” parameter function take the value of either Gini or MaxCut, which are functions that return the two subsets implied by the optimal split of their respective objective functions. Let featureType be 1, 2, 3, or 4 for Original, Pre PCA, Node Feature PCA, and Node Means PCA, respectively. Let PCA(X) be the function that return the principal components of X and meansPCA(X) the function from Fig. 1. Let $C(y)$ be the set of class values of vector y, $\{i | \exists y_j = i\}$, and let $U_{-i} = \{u | u \in U, y_u \neq i\}$.

To provide an intuitive visual understanding of how these eight variants function differently, we present the partitions generated for the example dataset in Fig. 2 at depth two (Fig. 6) and fully fit (Fig. 7) for each of the variants. To further provide an understanding of how these variants work, the reader is referred to the provided animation (Online Resource 1) of their fitting. This visual representation provides insights into why Max Cut Node Means PCA works so well, since its partition lines more naturally follow

Table 2 Variants' nomenclature [2]

Feature Type	Split Criteria	
	Gini	Max Cut
(1) Original	Gini Features	Max Cut Features
(2) Pre PCA	Gini Pre PCA Features	Max Cut Pre PCA Features
(3) Node Features PCA	Gini Node Features PCA	Max Cut Node Features PCA
(4) Node Means PCA	Gini Node Means PCA	Max Cut Node Means PCA

```
procedure decisionTree($U, (X,y), split, featureType$):
    begin
        if type = 2 do $X \leftarrow \mathrm{PCA}(X)$;
        $list \leftarrow \{(U,X,y)\}$;
        while $list \neq \emptyset$ do
            pop $(U,X,y)$ from list;
            if type = 1,2 do $X' \leftarrow X$;
            else if type = 3 do $X' \leftarrow \mathrm{PCA}(X)$;
            else do $X' \leftarrow \mathrm{meansPCA}(U,X,y)$;
            $P_1, P_2 \leftarrow split(U,X',y)$;
            if $|C(y(P_1))| \neq 1$ do list $\leftarrow$ list $\cup \{(P_1, X(P_1), y(P_1))\}$;
            if $|C(y(P_2))| \neq 1$ do list $\leftarrow$ list $\cup \{(P_2, X(P_2), y(P_2))\}$;
    end
```

Fig. 5 High-level description of decision tree fitting [2]

the lines of the data as compared to the baseline CART implementation.

Prototypical Comparative Study

Prototype Implementations

To make as fair a time-based comparison as possible between the eight different variants, we implemented prototypes of all eight in the Python programming language [17] to be used in our preliminary comparative study. Both the Gini Impurity and Max Cut optimal threshold calculation along a given feature were implemented as NumPy [15] vector operations to avoid the use of slower Python for loops, improving their runtime. Moreover, both utilize efficient $O(n \log n)$ implementations. The Scikit-Learn [16] package was used to perform PCA operations. We remark that both the Gini Impurity and Max Cut metric were implemented so that in the event of a tie, they favored thresholds that separated the observations into sets of similar cardinality, and the threshold was chosen to be the mean value between the two closest points that should be separated. We also note that we used, for the stopping rule, either the node having one class present or no split existing such that both sides of the branch contain at least one observation. Other than the necessarily different subroutines, to make time-based comparisons as consistent as possible, no difference existed between the different prototypes. Sect. "Advanced Implementation: MaxCutTree" will further verify runtime analysis derived by the prototype implementations by recreating runtime results for the Max Cut Node Means PCA variant compared to a commercial implementation when using a more advanced implementation.

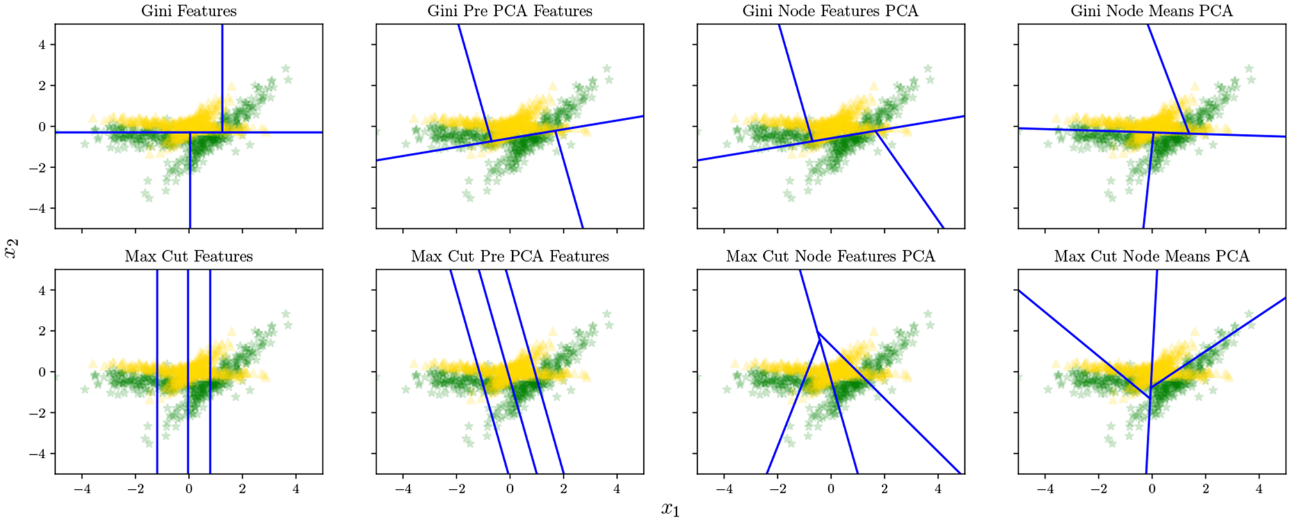

Fig. 6 Space partitions at depth 2 of each variant

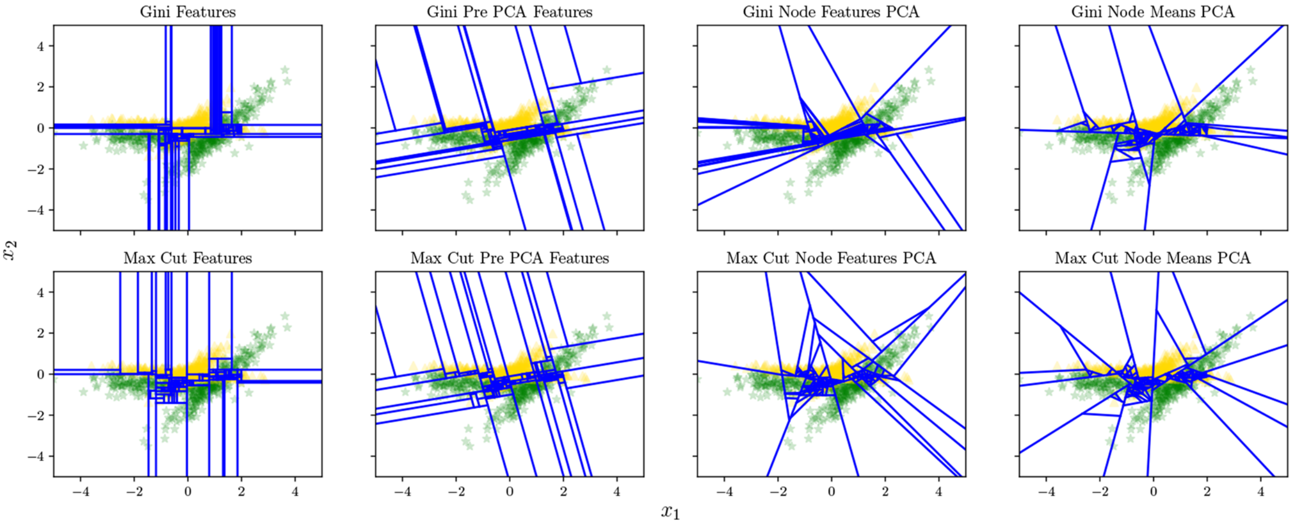

Fig. 7 Space partitions at full depth of each variant

Synthetic Datasets

We first consider the performance of the eight variants on synthetic datasets, which provide key advantages compared to real-world datasets. Foremost, synthetic datasets allow for generating an arbitrary number of independent datasets. This allowed us to draw statistically significant conclusions from the 20,000 plus datasets we generated. We were able to generate an arbitrarily large number of training and test examples with datasets ranging from 100,100 to 400,000 observations. This provides greater insights into the variants' performances on larger datasets than could have been achieved using only real-world data. Lastly, synthetic datasets provide fine-grained control of the datasets' different characteristics.

Experimental Design

The synthetic datasets were generated using the Scikit-Learn datasets.make_classification function [16]. This function generated one cluster for each of the classes. These clusters were then centered around the vertices of a hyper-cube of a specified dimension (one of the characteristics that we would modify). Random points would then be generated from a normal distribution around each cluster's center, and these points would be assigned to that cluster's corresponding class, except in 1% of cases where the class is randomly assigned. The features were then randomly linearly combined within each cluster to generate covariance. The option of having redundant or repeated features was not used in our experiments. We use the option of adding an arbitrary number of random features to the feature vector as a second characteristic that we control.

We observed how the following four factors affected accuracy, wall clock time, and the number of leaves in the tree throughout our experiments. The factors controlled are:

1. The number of training examples. We chose to fix the size of the testing set at 100,000 data points. For instance, if the training set consisted of 10,000 points, we would generate a dataset of 110,000 data points where 10,000 points would be randomly selected for training and the remaining 100,000 would be for testing.
2. The number of classes.
3. The number of informative features, referred to as the Dimension of Data in our results.
4. The number of meaningless (noisy) features, referred to as the Noise Dimension in our results.

We evaluated the variants' results on 30 independently generated datasets for each combination of parameter settings explored. For each of these datasets, the variants were presented with the same training/testing examples, which were standardized (zero mean and unit variance) using the Scikit-Learn StandardScaler [16] fit to the training examples. Standardization was used to make the distances in each feature more comparable, which will be important in the PCA analysis as well as finding the Max Cut. Note that both the Gini Impurity method and the orientation of the original feature vector are invariant under this transformation, indicating that the baseline variant's performance should also be invariant.

Relying on multiple independent datasets, we can calculate the significance of our results for each data generation setting using a one-tailed Paired Sample T-Test. For each variant, j, and data generation setting we consider

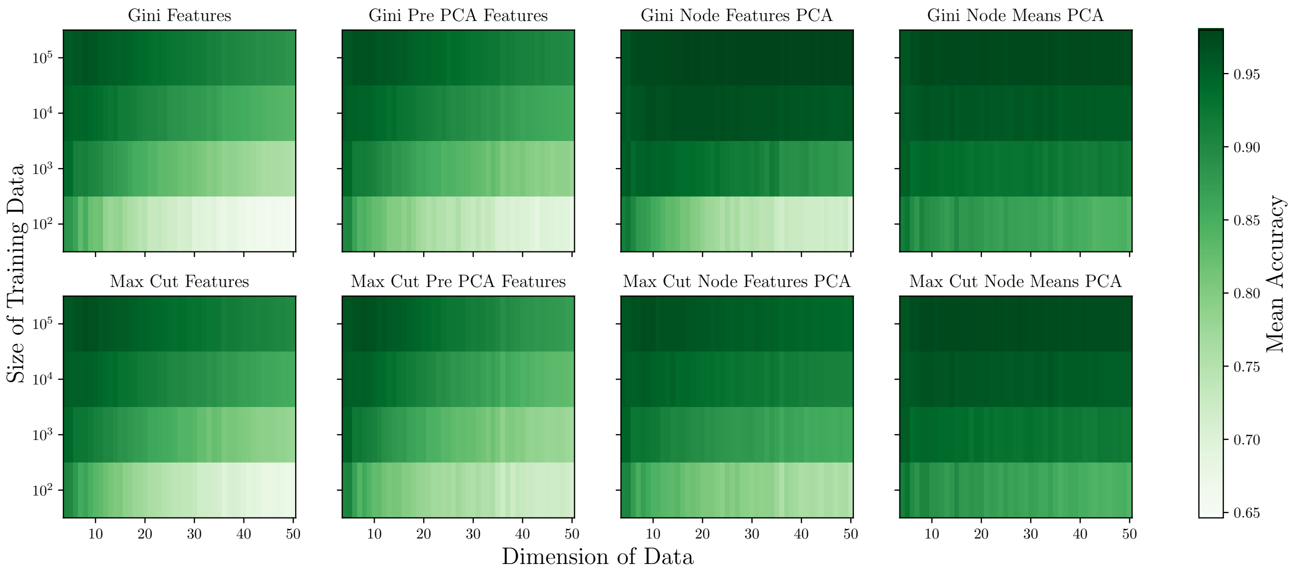

Fig. 8 Mean accuracy of binary classification on synthetic data, evaluated on test sets of 100,000 [2]

14 alternative hypotheses $\{A_i\}_{i=1}^{14}$ and corresponding null hypotheses $\{N_i\}_{i=1}^{14}$. Specifically, let $\{A_i\}_{i=1}^{7}$ represent the alternative hypotheses that variant j outperformed, in a pairwise comparison, each of the other seven variants. Conversely, let $\{A_i\}_{i=8}^{14}$ represent the alternative hypotheses that variant j underperformed, in a pairwise comparison, each of the other seven variants. We then used the Holm-Bonferroni Method [9] to correct for multiple hypotheses testing, testing for significance at the 5% level. In the case where all $\{N_i\}_{i=1}^{7}$ were rejected, we labeled that data setting on the accuracy significance graph as “Best”. In the case where any of $\{N_i\}_{i=8}^{14}$ were rejected we labeled that data setting on the accuracy significance graph as “Not Best”. Else we labeled the region as “Undecidable.”

All of these experiments were run on server nodes that each contained two Intel Xeon E5-2670 v2 CPUs for a total of 20 cores (no hyper-threading was used). Each decision tree was constructed using a single core and thread, such that up to 20 experiments were running simultaneously on a single node.

Binary Classification

The first set of experiments served to evaluate how the accuracy and run time of each variant varied with the size of the training set and the number of informative features. For this experiment, we set the noise dimension to zero. The results of these experiments are summarized in Figs. 8 and 9.

Examining Figs. 8 and 9, it is apparent that using locally meaningful feature representations (Node Features PCA / Node Means PCA) dramatically improves the accuracy when compared to the baseline, as well as the variant with PCA applied in preprocessing. Figure 9 makes it clear that, with statistical significance, the baseline variant is not the best performing variant on average. On the other hand, Fig. 9 indicates that the most promising variants are Gini Node Features PCA, Gini Node Means PCA, and Max Cut Node Means PCA

We first consider Gini Node Means PCA vs. Max Cut Node Means PCA. When looking at Figs. 8 and 9, no significant trends can be seen when comparing the two, except that Max Cut Node Means PCA seems to perform better more often for smaller training sets. We then examine the accuracy ratio of Max Cut Node Means PCA / Gini Node Means PCA for each of the 5,640 datasets in this experiment. From this analysis, we found the mean ratio to be 1.0003 and the sample standard deviation to be 0.0087, implying a 95% confidence interval for the mean ratio of (1.00008, 1.00053). This implies that Max Cut Node Means PCA had a slight advantage over the Gini Node Means PCA when considering the average ratio. Therefore, we decided to compare Gini Node Features PCA vs. Max Cut Node Means PCA.

When considering Gini Node Features PCA vs. Max Cut Node Means PCA, the most prominent pattern that emerges is that Gini Node Features PCA performs better for large training sets (based on accuracy) and Max Cut Node Means PCA performs better for smaller training sets. This pattern of Max Cut Node Means PCA performing better for harder datasets can be seen in Fig. 10. It is important to note that this outperformance is almost entirely attributed to the choice of feature vector type as the same pattern emerges if Gini Node means PCA was used instead of Max Cut Node Means PCA. In the cases where 1000 training samples were

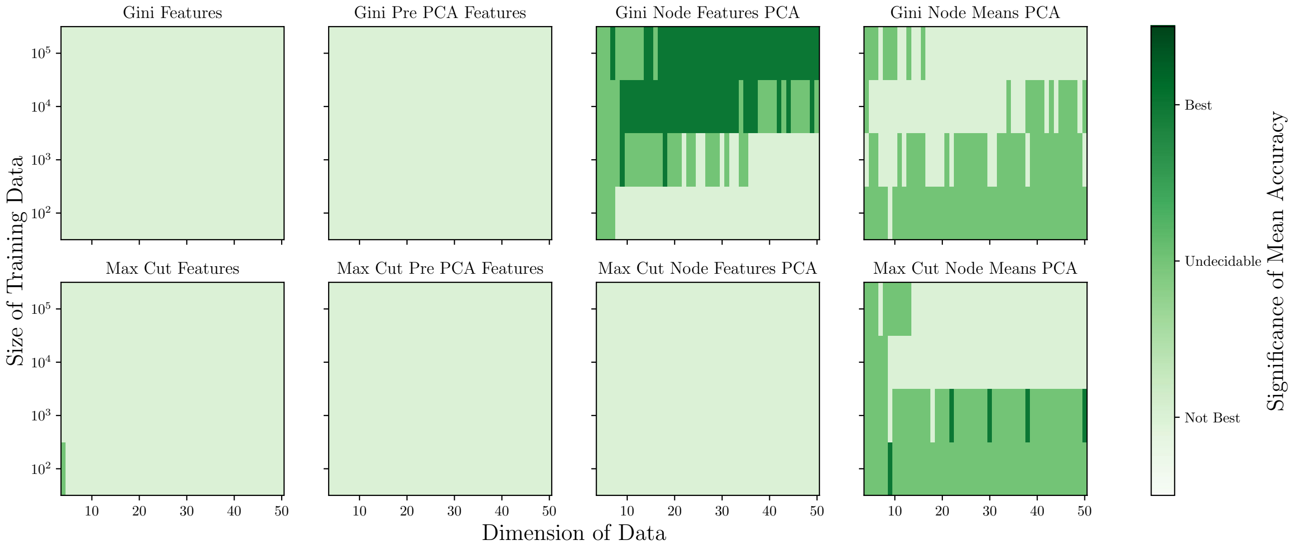

Fig. 9 Significance of mean accuracy of binary classification on synthetic data, evaluated on test sets of size 100,000 [2]

used, Node Means PCA, after a slight dip for low dimensional data, increasingly outperforms Node Features PCA as the dimensionality increases. One reason for this is that Node Means PCA uses averaging, thereby reducing noise. As a result of the dimensionality reduction, Node Means PCA searches fewer basis vectors than Node Features PCA. In fact, in the binary case, there are only two classes; hence the Node Means PCA variant searches only two basis vectors to select a threshold.

Figure 11 is the histogram of all 5640 ratios of the Max Cut Node Means PCA Accuracy divided by the Gini Node Features PCA Accuracy. It is evident from the right skew in Fig. 11 that when Node Means PCA provides better accuracy, it does so considerably. On the other hand, when it provides less accuracy, it does so with only a minor loss. To further demonstrate this point, conditioned on the ratio being greater than 1 (which happens 49.6% of the time), the mean is 1.075, and the max is 1.430. When the ratio is conditioned to be less than 1 (which happens 50.4% of the time), the mean is 0.989, and the minimum is 0.915. This shows that Node Means PCA has the potential of a significant improvement in accuracy with only a minor risk of accuracy loss.

Figure 12 shows the relative runtime for each of the different variations. When focusing on these experimental results for Node Means PCA vs. Node Features PCA, we see that Node Means PCA has significantly better properties concerning computational cost. Max Cut Node Means PCA took between 27 and 98% less time than Gini Node Features PCA to fit and between 21 and 98% less time than the baseline variant. The computational efficiency of Node Means PCA is partially explained by the computational efficiency of the split as provided in Table 1. However, another factor is the size of the decision tree, for which the number of leaves

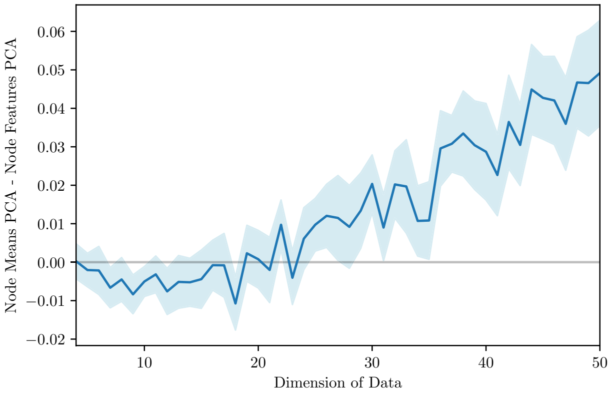

Fig. 10 Difference between Max Cut Node Means PCA Accuracy and Gini Node Features PCA Accuracy with 95% confidence interval (shaded) using the Training Sets of size 1000 [2]

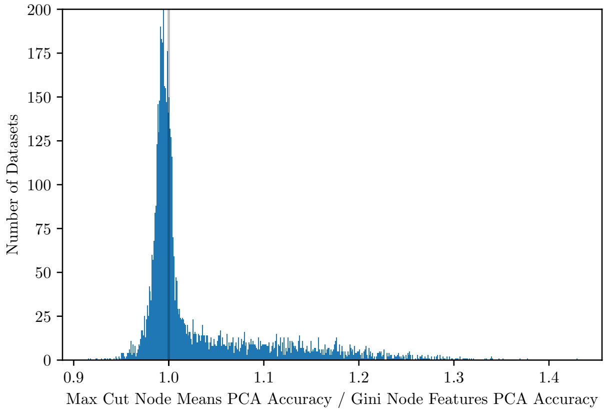

Fig. 11 Histogram of Max Cut Node Means PCA Accuracy, normalized with respect to the accuracy of Gini Node Features PCA Accuracy [2]

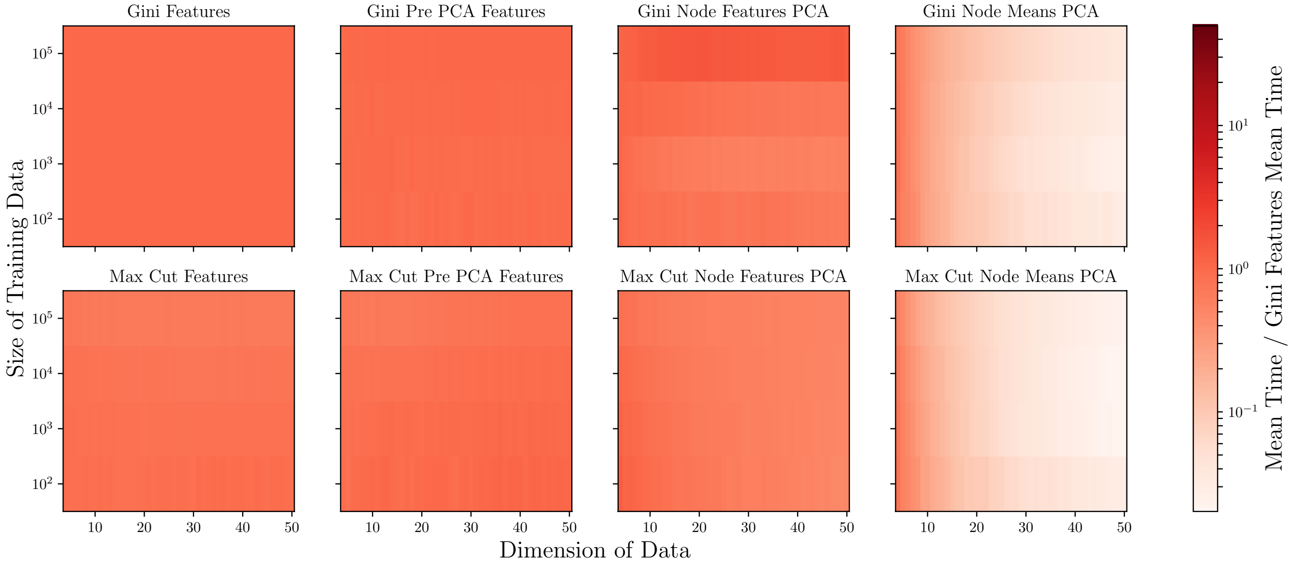

Fig. 12 Log ratio of mean CPU time for each of the variants with respect to the CPU time of the Gini Features baseline, for binary classification decision trees trained on synthetic data

Table 3 Mean Number of Leaves in Decision Tree [2]

Variant	Mean # of Leaves
Baseline	1129
Gini Node Features PCA	418
Gini Node Means PCA	429
Max Cut Node Means PCA	448

is a proxy. The mean number of leaves across all datasets is provided in Table 3.

Table 3 shows that the average number of leaves is significantly less for Gini Node Features PCA, Gini Node Means PCA, and Max Cut Node Means PCA compared to the Baseline variant. This reduction can help account for the significant observed computational efficiency of the Node Means PCA vs. the Baseline. Therefore, considering the costs and benefits of computational time, accuracy, and robustness to higher-dimensional problems, the Node Means PCA variant

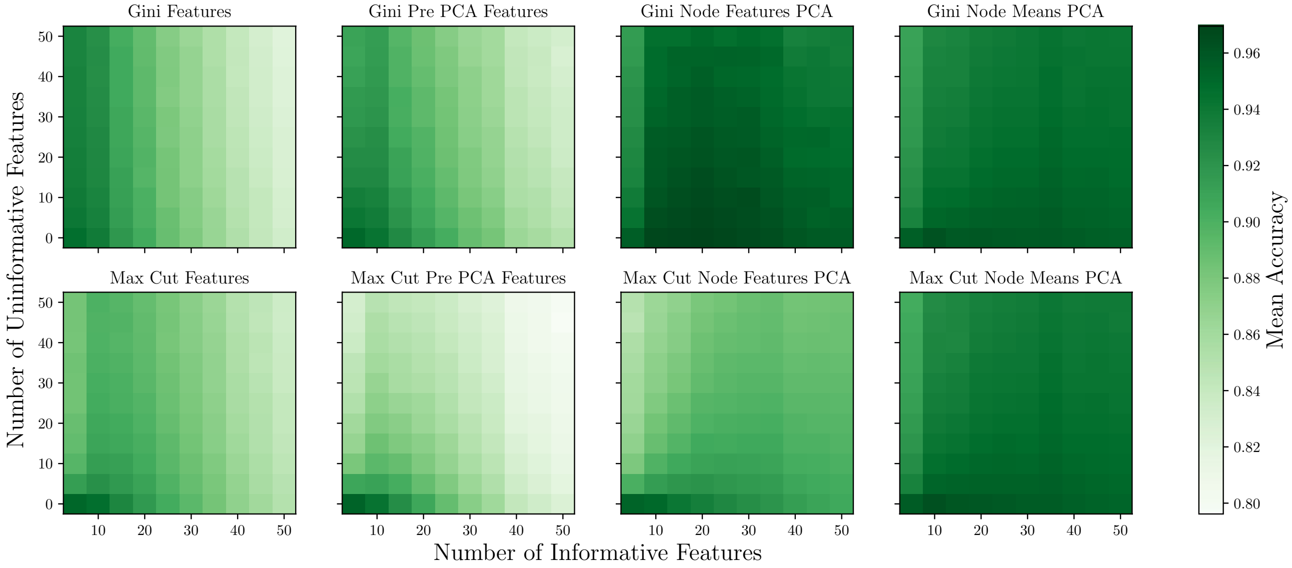

Fig. 13 Suitability of Mean accuracy to the addition of noise features for binary classification on synthetic data, with training set of size 10,000 and test sets of size 100,000

has the best performance, with a slight preference towards the Max Cut metric over Gini Impurity.

The second experiment considers the addition of noise features to see if the Localized PCA variants are negatively affected. For this experiment, the training set sizes were fixed to 10,000 and then for each combination of meaningful dimension in $\{5, 10, 15, ..., 50\}$ and noise dimension in $\{0, 5, 10, ..., 50\}$, we generated and analyzed 30 datasets. The results of this experiment can be seen in Fig. 13.

The major result of analyzing the addition of noise features is that local features continue to provide significant benefits (seen in the darker shading of Fig. 13) except for a few minor exceptions. Specifically, when the real dimension was five and the noise dimension was greater than or equal to 15. This represents the cases where the noise features account for 75% or more of the available features and there are a small number of informative features. We do note that the Gini based variants tend to outperform the Max Cut variants with the noise; however, the Max Cut Node Means PCA variant still provides, in the presence of noise, significant benefits compared to the baseline variant while not significantly under-performing any of the other variants. Therefore, this experiment shows that the Max Cut Node Means PCA variant is advantageous to use even when the noisiness of each feature is uncertain/unknown.

Through the analysis of synthetic binary data, the Max Cut Node Means PCA variant provides significant advantages. Precisely, the Max Cut Node Means PCA variant captures most of the accuracy benefits of using a unique feature representation at each node while also decreasing the running time. Together, these two benefits demonstrate the significant advancement to decision trees that the new Max Cut Node Means PCA variant provides.

Multiclass Classification

The next experiment investigates the impact of using data with ten classes rather than just two. Referring to the same steps as in Sect. “Binary Classification”, one can see how different training set sizes, informative dimension, and noise dimension can affect the variant’s performance. The main results derived from varying the training data size and the underlying dimension of the data can be seen in Figs. 14, 15, and 16. An important note is that these plots do not use a perfect logarithmic scale for the y-axis. The values for each horizontal bar are 10^2, 10^3, 10^4, 10^5, 2×10^5, and 3×10^5 (note that the last two are not 10^6 and 10^7 as one might initially suspect).

As seen in Fig. 15, the baseline Gini Features variant is “Not Best” for all tested synthetic dataset parameters. The prevalence of these “Not Best” markings indicates with statistical significance that the baseline is not the best variant on average, much like in the binary case (Sect. “Binary Classification”). The “Best” markings indicate, from an accuracy standpoint, that Max Cut Nodes Means PCA was the best option in most cases.

Moreover, Fig. 16 shows that Max Cut Nodes Means PCA provides significant computational advantages for high dimensional data (taking up to 93% less time). It can, however, take up to 22% longer when looking at low dimensional problems, which is not a fundamental issue. This is since lower-dimensional problems are computationally faster, so the increased relative time does not translate to significant realized computational time.

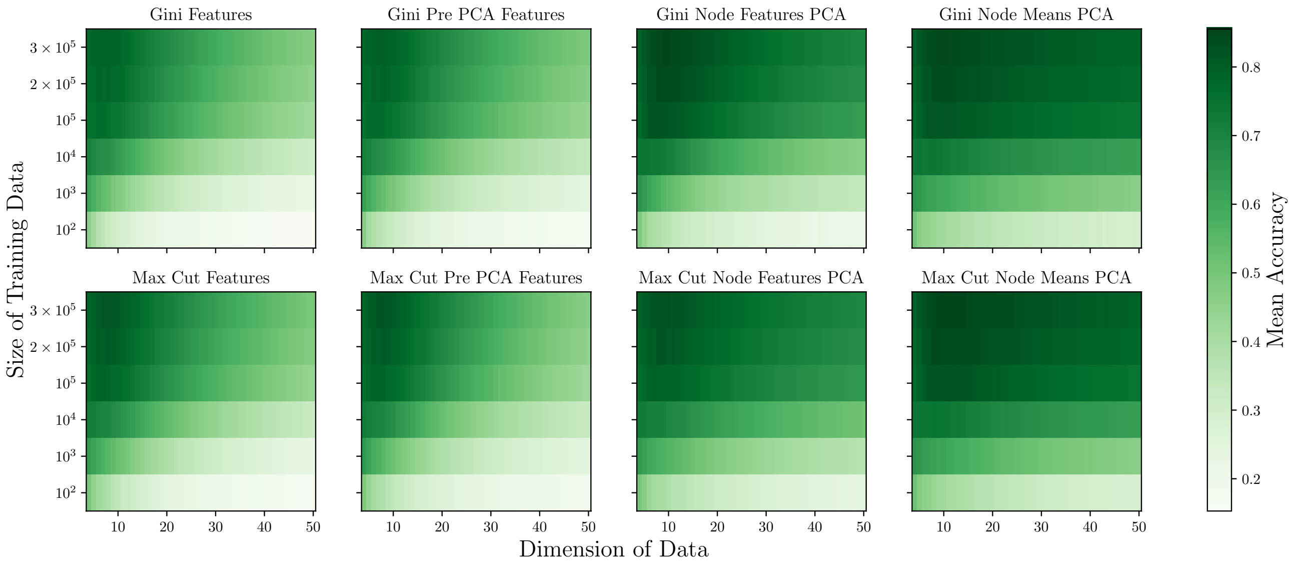

Fig. 14 Mean accuracy of 10-class classification on synthetic data, evaluated on test sets of size 100,000 [2]

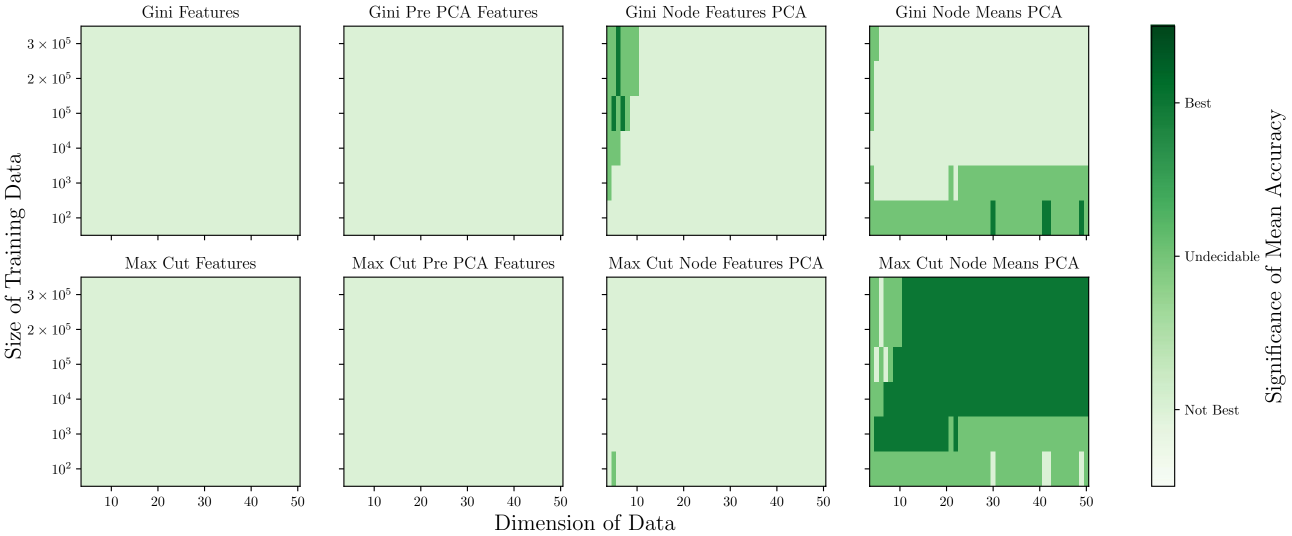

Fig. 15 Significance of mean accuracy of 10-class classification on synthetic data, evaluated on test sets of size 100,000 [2]

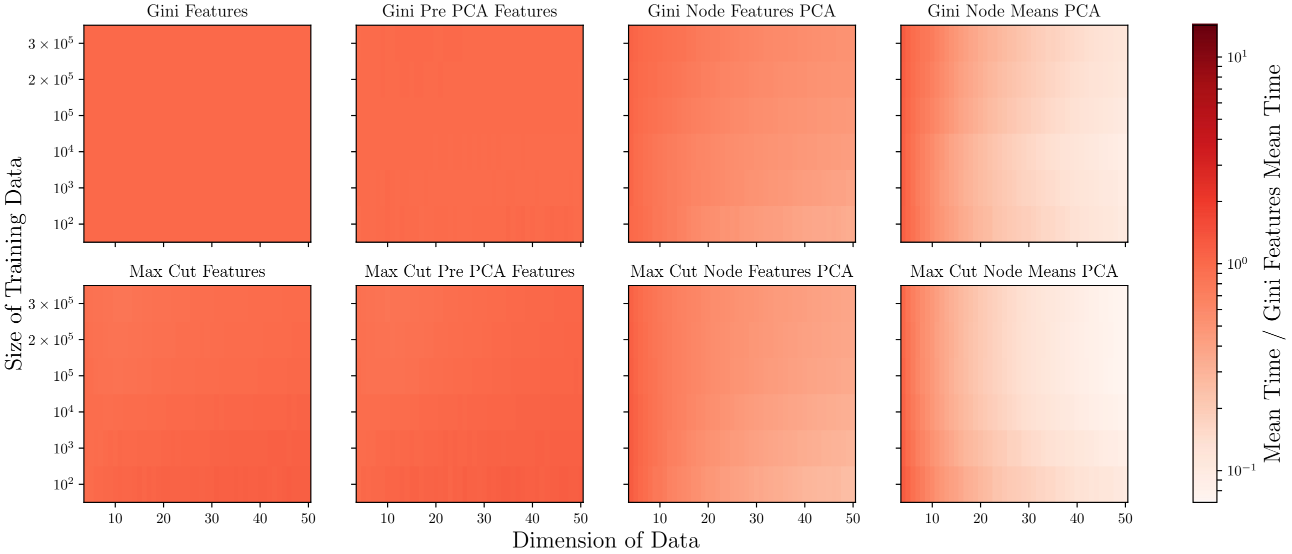

Fig. 16 Log ratio of mean CPU time with respect to Gini Features baseline, for 10-class classification decision tree construction on synthetic data

Upon close inspection of Fig. 15, the markings indicate that the Gini Node Features variant may be the best option in terms of accuracy for larger training sets and fewer features. This is the same pattern that emerged in the binary case, and Fig. 17 allows for a closer comparison of these two variants, focusing specifically on the 300,000 case. It demonstrates that while Gini Node Features may outperform in the fewer dimensional, larger training set case, Max Cut Node Means PCA rapidly and dramatically outperforms Gini Node Features as the number of features increases.

When the comparison is made between the Max Cut Node Means PCA variant and the Gini Node Means PCA variant, Fig. 18 helps demonstrates the marginal but statistically significant improvement by utilizing the Max Cut metric. Specifically, the 95% confidence interval for the mean ratio between the accuracy of the Max Cut Node Means PCA variant and the Gini Node Means PCA variant is (1.0060, 1.0071). Thus, we can conclude that the use of Max Cut does appear to improve accuracy on average.

When analyzing how the performance of the variants, reported in Fig. 19, was affected by the inclusion of noise

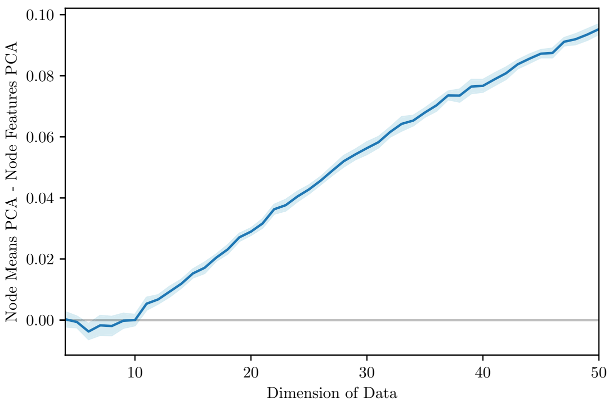

Fig. 17 Difference between Max Cut Node Means PCA Accuracy and Gini Node Features PCA Accuracy with 95% confidence interval (shaded) using the Training Sets of size 300,000 [2]

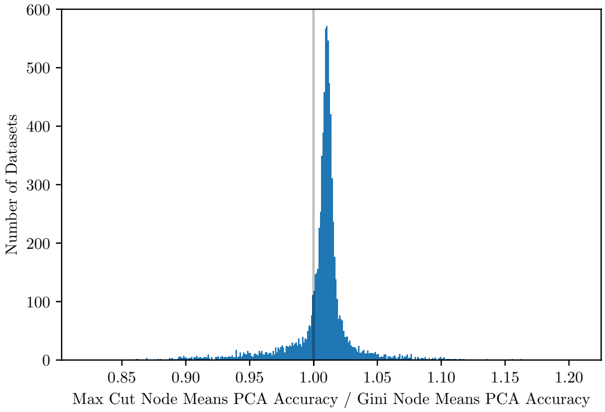

Fig. 18 Histogram of Max Cut Node Means PCA Accuracy, normalized with respect to the accuracy of Gini Node Means PCA Accuracy [2]

features, the advantages of using Node Means PCA are apparent in most cases. Like in the binary case, the baseline variant only has better performance in cases with very high percentages of noise features (greater than 50%) and few meaningful features. Since the improved accuracy of Max Cut Node Means PCA is still held for higher-dimensional problems, even when the percentage of noise features exceeded 50%, we believe that the Max Cut Node Means PCA variant is broadly applicable.

Synthetic Dataset Results Summary

Extensive testing indicates that our localized PCA variants provide dramatic improvements over the baseline variant in all but the most extreme noise conditions. A thorough analysis of these localized PCA variants found that from an accuracy standpoint, Node Means PCA should be preferred for increasingly difficult classification tasks, particularly in higher-dimensional cases where there are more classes and/or fewer training examples. Moreover, when Node Means PCA was not the best option, it was typically only outperformed by a small margin, and when it was the best option, it could dramatically improve results. Empirically, Node Means PCA results in significantly faster computation times than the baseline and Node Features variants. Thus, Node Means PCA provides dramatic improvements in both accuracy and run time.

We further compared the Max Cut and Gini Impurity metrics and found that when the Max Cut metric was used in conjunction with the Node Means PCA variant; it can increase a decision tree's performance on average. Based on our experimentation on more than 20,000 diverse datasets, we generally recommend that Max Cut Node Means PCA be considered as an alternative to the traditional baseline variant. We have thus shown that for synthetic datasets, Max Cut Node Means PCA yields a significant and dramatic improvement over the baseline variant in accuracy and computational efficiency. Next, we present the performance of our variant for real-world datasets.

Real-World Datasets

We proceed to explore the effects of the eight different variants on real-world datasets, considering a total of six different datasets. We used the Iris dataset, Wine Quality dataset [5], and the Indian Liver Patient dataset (ILPD), all three taken from the UCI Machine Learning Repository [6]. The Wine Quality dataset is separated into two sets, red and white wines, and in our experimentation, we consider each of these sets as a single combined dataset and as individual sets. The wine's classification was included as another feature in the combined dataset, one if red, zero otherwise. We also used the MNIST dataset [12], as well as the CIFAR-10 and CIFAR-100 datasets [11]. Table 4 provides a brief overview of each of these datasets. Note that the ILPD dataset lists only 579 samples compared to the 583 in the dataset; this is since we removed samples with missing values.

These experiments on real-world data were run on server nodes that each contained two Intel Xeon E5-2670 v2 CPUs for a total of 20 cores (no hyper-threading was used). Each decision tree was constructed using a full server node utilizing parallel processing due to the NumPy vector operations. The total CPU time was used for these experiments rather than recording wall clock time as in the previous synthetic dataset experiments.

We considered each variant without scaling the features and with standardizing the features as a pre-processing step and then reported the best result. To evaluate the performance of each variant, we used one of two methods: either 10 × 10 cross-validation (as in the case of Iris, ILPD,

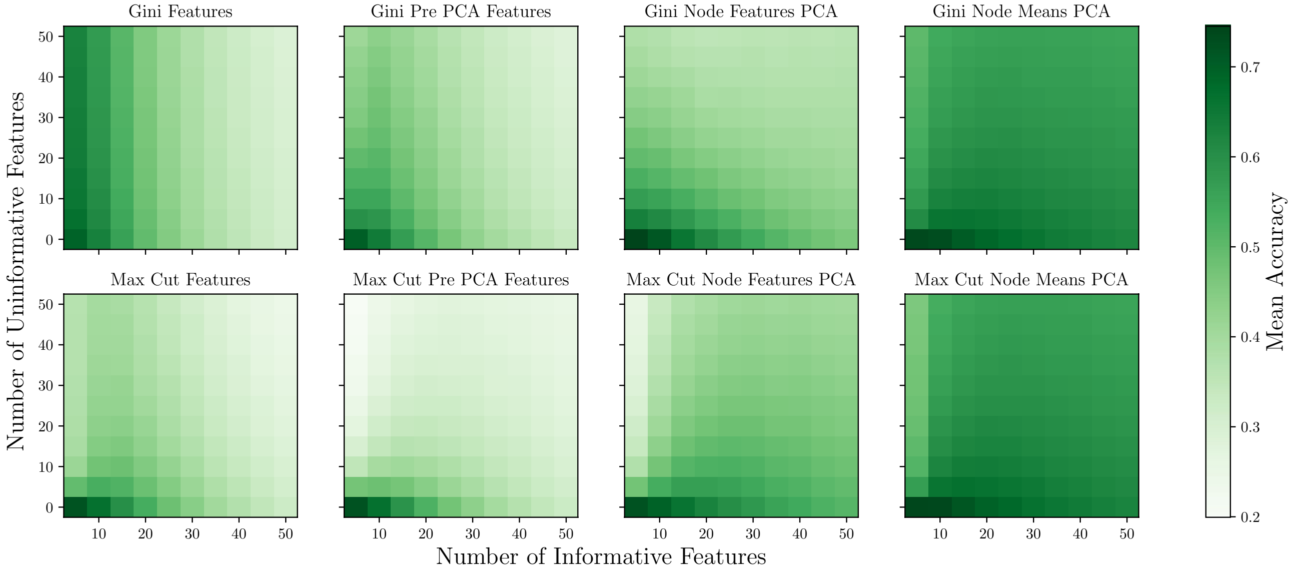

Fig. 19 Mean accuracy robustness for 10-class classification on synthetic data, with training set of size 10,000 and test sets of size 100,000

Table 4 Dataset Characteristics [2]

Dataset	Samples	Features	Classes
Iris	150	4	3
ILPD	579	10	2
Wine-Red	1599	11	6
Wine-White	4898	11	7
Wine-Both	6497	12	7
MNIST	70,000	784	10
CIFAR-10	60,000	3072	10
CIFAR-100	60,000	3072	100

Wine-Red, Wine-White, and Wine-Both), or an 80%–20% train-test data split (as in the case of MNIST, CIFAR-10, and CIFAR-100). The decision not to use 10 × 10 cross-validation for the MNIST, CIFAR-10, and CIFAR-100 datasets resulted from the significant computational requirements for computing these trees. Moreover, due to the large size of the datasets, the variance in accuracy should be lower than that in smaller datasets, implying that the 10 × 10 cross-validation would not be as beneficial. The results of these experiments are reported in Table 5, with the mean value provided for the Iris, ILPD, Wine-Red, Wine-White, and Wine-Both datasets.

The results of the real-world analysis verify that the improvements in accuracy are similar to those in the synthetic datasets. We found that in all but one of the problems analyzed, the Node Means PCA modification was the best option to use. For the ILPD dataset, the only dataset in which a Node Means PCA variant did not achieve the highest accuracy, Node Means PCA achieved the second-highest accuracy. Moreover, the highest accuracy for the ILPD dataset was achieved by the Max Cut Pre PCA Features variant, which should have benefited from the unique structure of the dataset, namely the natural clustering of male and female observations when the first three principal components are analyzed [8]. Defining performance as the percentage improvement in accuracy relative to the baseline variant (utilizing the better-performing metric between Max Cut and Gini), this modification had improvements of 1.6%, 3.7%, 1.9%, 1.9%, 1.9%, 6.3% 33.3%, and 49.4% for each of the datasets.

Moreover, Max Cut was the best choice in six out of eight problems. Defining performance as the percentage improvement of the best Max Cut variant compared to the best performing Gini variant, Max Cut had improvements of 1.1%, 0.9%, -0.3%, 0.0%, 0.5%, 0.2% 2.0%, and 6.9%. Finally, Max Cut Node Means PCA was the best choice in five out of eight problems and was tied for the best in one problem. Defining performance as the percentage improvement of accuracy compared to the baseline variant, Max Cut Node Means PCA represented an improvement of 1.6%, 2.0%, 1.0%, 1.9%, 1.9%, 6.3%, 33.3%, and 49.4%.

Table 5 shows that Max Cut Node Means PCA is invariably the fastest variant for larger problems. Although Max Cut Node Means PCA took more time for small datasets, we do not believe this discounts its overall performance. Regardless of the variant chosen, the decision trees are fit in a negligible amount of time for the smaller datasets relative to the larger datasets. Moreover, Max Cut Node Means PCA still provides the previously mentioned accuracy benefits. For the larger datasets, such as CIFAR-100, the Max Cut Node Means PCA variant reduced the required CPU time by 94% compared to

Table 5 Results of Real-World Dataset Experiments [2]

	Iris			ILPD		
	Accuracy	Relative Time	Scaled	Accuracy	Relative Time	Scaled
Gini Features	0.945	1.000	False	0.652	01.000	False
Gini Pre PCA Features	0.941	1.000	True	0.671	15.872	False
Gini Node Features PCA	0.944	1.309	True	0.660	18.134	False
Gini Node Means PCA	0.950	0.894	False	0.676	07.147	True
Max Cut Features	0.947	1.032	True	0.649	01.333	True
Max Cut Pre PCA Features	0.950	0.915	False	**0.682**	16.168	True
Max Cut Node Features PCA	0.941	1.213	False	0.653	17.136	False
Max Cut Node Means PCA	**0.960**	**0.798**	False	0.665	**00.467**	True
	Wine-Red			Wine-White		
	Accuracy	Relative Time	Scaled	Accuracy	Relative Time	Scaled
Gini Features	0.626	01.000	False	0.623	1.000	False
Gini Pre PCA Features	0.636	03.229	True	0.615	1.216	True
Gini Node Features PCA	0.619	11.690	True	0.615	8.301	True
Gini Node Means PCA	**0.638**	06.707	True	**0.635**	3.788	True
Max Cut Features	0.629	**00.799**	True	0.627	**0.548**	True
Max Cut Pre PCA Features	0.632	03.308	True	0.628	1.076	True
Max Cut Node Features PCA	0.636	12.479	True	0.634	8.740	True
Max Cut Node Means PCA	0.631	05.539	True	**0.635**	4.449	True
	Wine-Both			MNIST		
	Accuracy	Relative Time	Scaled	Accuracy	Relative Time	Scaled
Gini Features	0.624	1.000	False	0.869	1.000	False
Gini Pre PCA Features	0.617	1.214	False	0.823	1.558	True
Gini Node Features PCA	0.608	9.122	True	0.863	2.612	False
Gini Node Means PCA	0.633	4.231	True	0.922	0.186	False
Max Cut Features	0.623	**0.648**	True	0.847	0.744	False
Max Cut Pre PCA Features	0.626	1.046	True	0.866	1.091	False
Max Cut Node Features PCA	0.633	9.755	True	0.897	1.552	False
Max Cut Node Means PCA	**0.636**	4.251	True	**0.924**	**0.093**	False
	CIFAR-10			CIFAR-100		
	Accuracy	Relative Time	Scaled	Accuracy	Relative Time	Scaled
Gini Features	0.262	1.000	False	0.083	1.000	False
Gini Pre PCA Features	0.246	1.162	False	0.073	1.259	False
Gini Node Features PCA	0.290	1.173	False	0.090	1.010	True
Gini Node Means PCA	0.342	0.085	False	0.116	0.132	False
Max Cut Features	0.243	1.313	False	0.077	0.626	False
Max Cut Pre PCA Features	0.271	1.503	False	0.090	0.637	False
Max Cut Node Features PCA	0.309	1.034	True	0.109	0.491	False
Max Cut Node Means PCA	**0.349**	**0.075**	False	**0.124**	**0.061**	True

The best value in each category is indicated in bold the baseline variant. Thus, Max Cut Node Means PCA provides significant computational and accuracy advantages.

Advanced Implementation: MaxCutTree

To compare our running time results with commercial-quality implementations of decision trees, we implemented the Max Cut Node Means PCA as a python package "MaxCutTree" using Cython [1]. We then compare the CPU time required to train a decision tree using MaxCutTree to the commercial quality Scikit-Learn implementation of CART. We first compare our new implementation by repeating the first experiment from both Sects. "Binary Classification" and "Multiclass Classification" with the added extension of including problems where the dimension of the feature vector is up to 200. The results of this are presented in Figs. 20 and 21 where we mark the areas where the MaxCutTree implementation outperforms with '/' markings.

Figure 20 shows that our MaxCutTree implementation provides runtime advantages compared to a commercial baseline for sufficiently large or high dimensional datasets. This pattern of the computational advantages increasing with the dimension of the feature vector is the same as can be seen in the runtime analysis for the prototype implementations (see Figs. 12 and 16). Specifically, our MaxCutTree implementation outperforms the commercial implementation in both the Binary and Multiclass case for the largest problems, the upper right-hand corner, which are also the problems that take the longest to fit. Moreover, the computational improvements provided by our implementation are precisely where the MaxCutTree variant provides the most improvements to accuracy, as shown in Fig. 21 where MaxCutTree can achieve an accuracy over 3× that of the Scikit–Learn variant (note as MaxCutTree and Scikit–Learn are only specific implementations of the Max Cut Node Means PCA and Gini Features variants respectively, the accuracy results are the same as for the prototype implementations).

These computational speed-ups are attained for real-world datasets as well: Table 6 provides the speed-up factor of MaxCutTree for the three largest real-world dataset problems presented in Sect. "Real-World Datasets". Therefore, replicating the pattern seen in the prototype implementation analysis when compared against a commercial implementation providing further credence to the run time advantages of the Max Cut Node Means PCA variant. Further, it is anticipated that even more advanced optimizations such as memory management, once fully implemented, will further decrease the size of the problems for which computational advantages are realized against commercial implementations.

Conclusions

This paper shows the benefits of two important modifications to the CART methodology for constructing classification decision trees. We first introduced a Max Cut based metric for determining the optimal binary split as an alternative to the Gini Impurity method (see appendix for an $O(n \log n)$ implementation of Max Cut along a single feature). We then introduced Node Means PCA to determine locally meaningful directions for considering splits with a focus on discriminating between classes. Together these modifications form our novel Max Cut Node Means PCA variant. We first make a theoretical comment on how these modifications are reflected in the asymptotic runtime of node splitting and observe that Node Means PCA improves upon the traditional method.

Our extensive experimental analysis included more than 20,000 synthetically generated datasets with training sizes ranging from 100 to 300,000 and the number of informative features ranging between 4 and 50. We considered both binary and 10-class classification tasks. These experiments demonstrate the significant improvements in increased accuracy and decreased computational time provided by utilizing the Max Cut Node Means PCA variant. Furthermore, we show that the accuracy improvements become even more substantial as the dimension of the data and/or the number of classes increases and that the runtime improves with the dimension and size of the datasets.

Analysis was also done on real-world datasets. Our results indicate that the Max Cut Node Means PCA variant remains advantageous even in real-world applications. For example, we show that in the case of CIFAR-100 (100 classes, 3,072 features, and 48,000 training points out of 60,000 total), our variant results in a 49.4% increase in accuracy performance compared to the baseline CART variant, while simultaneously providing a 6.8× speed up compared to the Scikit-Learn implementation when using our MaxCutTree implementation. The Max Cut Node Means PCA variant (MaxCutTree) helps bring decision trees into the world of big, high-dimensional data. Our experiments demonstrate the significant improvements that the Max Cut Node Means PCA variant has on constructing classification decision trees for these types of datasets. Further research on how these novel decision trees affect the performance of ensemble methods may lead to even greater advancements in the area.

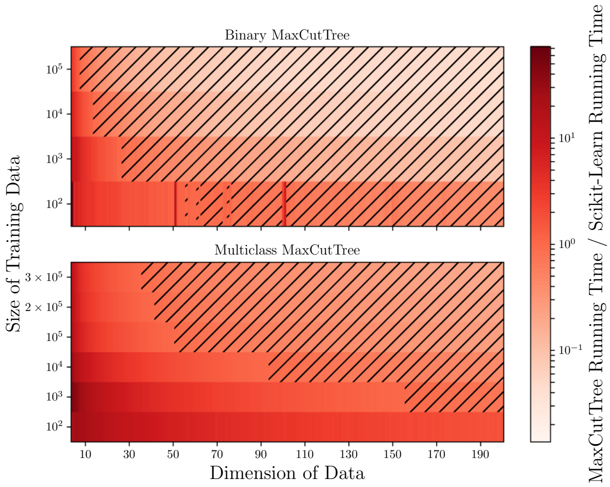

Fig. 20 Log ratio of mean CPU time with respect to Scikit-Learn, for MaxCutTree Binary and 10-class classification decision tree construction on synthetic data

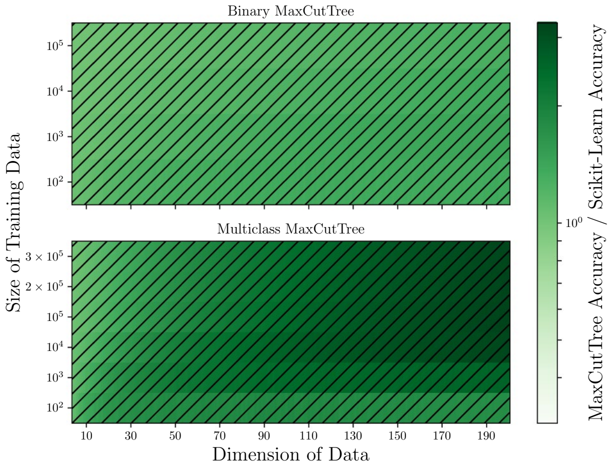

Fig. 21 Log ratio of mean accuracy with respect to Scikit-Learn, for MaxCutTree Binary and 10-class classification decision tree construction on synthetic data

Table 6 MaxCutTree vs. Scikit-Learn

	MNIST	CIFAR-10	CIFAR-100
MaxCutTree	**0:02.91**	**0:20.65**	**0:57.89**
Scikit-Learn	0:09.39	2:06.37	6:33.78
Speed-Up Factor	3.2×	6.1×	6.8×

The best value in each category is indicated in bold

Appendix

Implementation of Max Cut Metric in $O(n \log n)$:

The first step is to sort the observations in ascending order such that $x_{i,j} < x_{k,j} \; \forall i < k$, which is known to be implemented in $O(n \log n)$ time. There are a total of n possible splits to consider; let the value achieved by the split between $x_{i,j}$ and $x_{k,j}$ be referred to as θ_i. We will now show that given θ_{i-1}, θ_i can be computed in constant time. It is evident that the following equality holds:

$$\begin{aligned}
\theta_i = \theta_{i-1} &- \sum_{t<i} \mathbb{1}_{y_i \neq y_t} \left| x_{i,j} - x_{t,j} \right| \\
&+ \sum_{t>i} \mathbb{1}_{y_i \neq y_t} \left| x_{i,j} - x_{t,j} \right| \\
= \theta_{i-1} &- \sum_{t<i} \mathbb{1}_{y_i \neq y_t} \left(x_{i,j} - x_{t,j} \right) \\
&+ \sum_{t>i} \mathbb{1}_{y_i \neq y_t} \left(x_{t,j} - x_{i,j} \right) \\
= \theta_{i-1} &+ \sum_{t<i} \mathbb{1}_{y_i \neq y_t} x_{t,j} - \mathbb{1}_{y_i \neq y_t} x_{i,j} \\
&+ \sum_{t>i} \mathbb{1}_{y_i \neq y_t} x_{t,j} - \mathbb{1}_{y_i \neq y_t} x_{i,j} \\
= \theta_{i-1} &+ \sum_{t} \mathbb{1}_{y_i \neq y_t} x_{t,j} - \mathbb{1}_{y_i \neq y_t} x_{i,j} \qquad \text{Since } \mathbb{1}_{y_i \neq y_i} = 0 \; \forall i \\
= \theta_{i-1} &+ \left(\sum_{t} \mathbb{1}_{y_i \neq y_t} x_{t,j} \right) - x_{i,j} \left(\sum_{t} \mathbb{1}_{y_i \neq y_t} \right)
\end{aligned}$$

Since $\sum_t \mathbb{1}_{y_i \neq y_t} x_{t,j}$ and $\sum_t \mathbb{1}_{y_i \neq y_t}$ are only dependent on the class of observation i, these can be calculated once for each class c and recorded as S_c and N_c respectively. Then, $\theta_i = \theta_{i-1} + S_{y_i} - x_{i,j} N_{y_i}$Therefore, the complexity of the split per feature is $O(n \log n)$.

Acknowledgements This research used the Savio computational cluster resource provided by the Berkeley Research Computing program at the University of California, Berkeley (supported by the UC Berkeley Chancellor, Vice Chancellor for Research, and Chief Information Officer).

Funding The authors were supported in part by NSF award No. CMMI-1760102. The second author was also supported in part by AI institute NSF award 2112533.

Data availability The package used to generate the synthetic datasets in Sect. "Synthetic Datasets", all of the datasets in Sect. "Real-World Datasets", and the package used to generate the extra synthetic datasets in Sect. "Advanced Implementation: MaxCutTree" are freely available online. The random seeds used to generate the synthetic datasets will be provided upon request.

Declarations

Conflict of interest: The authors declare that they have no conflict of interest.

Code availability: All the Python packages referenced are freely available online. The implementations of the variants in Sect. "Prototypical Comparative Study" are custom and available upon request. The implementation of MaxCutTree in Sect. "Advanced Implementation: MaxCutTree" is available on GitHub at the following url: https://github.com/hochbaumGroup/Max-cut-decision-tree

References

1. Behnel S, Bradshaw R, Citro C, Dalcin L, Seljebotn DS, Smith K. Cython: The best of both worlds. Comput Sci Eng. 2011;13(2):31–9.
2. Bodine J, Hochbaum DS. The max-cut decision tree: Improving on the accuracy and running time of decision trees. In: Proceedings of the 12th International Joint Conference on Knowledge Discovery, Knowledge Engineering and Knowledge Management - KDIR,, pp. 59–70. INSTICC, SciTePress (2020). https://doi.org/10.5220/0010107400590070
3. Breiman L. Random forests. Mach Learn. 2001;45(1):5–32. https://doi.org/10.1023/a:1010933404324.
4. Breiman L, Friedman J, Stone C, Olshen R. Classification and Regression Trees. The Wadsworth and Brooks-Cole statistics-probability series. Taylor & Francis (1984). https://books.google.com/books?id=JwQx-WOmSyQC
5. Cortez P, Cerdeira A, Almeida F, Matos T, Reis J. Modeling wine preferences by data mining from physicochemical properties. Decis Support Syst. 2009;47:547–53.
6. Dua D, Graff C. UCI machine learning repository (2017). http://archive.ics.uci.edu/ml
7. Liii FRSKP. on lines and planes of closest fit to systems of points in space. Lond Edinburgh Dublin Philosophical Mag J Sci. 1901;2(11):559–72. https://doi.org/10.1080/14786440109462720.
8. Hochbaum DS, Baumann P. Sparse computation for large-scale data mining. In: 2014 IEEE International Conference on Big Data (Big Data), pp. 354–363 (2014). https://doi.org/10.1109/BigData.2014.7004252
9. Holm S. A simple sequentially rejective multiple test procedure. Scandinavian J Stat. **6**(2), 65–70 (1979). http://www.jstor.org/stable/4615733
10. Karp RM. Reducibility among combinatorial problems. In: R.E. Miller, J.W. Thatcher, J.D. Bohlinger (eds.) Complexity of computer computations, pp. 85–103. Plenum Press, New York (1972). https://doi.org/10.1007/978-1-4684-2001-2_9.
11. Krizhevsky A. Learning multiple layers of features from tiny images. Tech. rep. (2009)
12. LeCun Y, Cortes C, Burges C. Mnist handwritten digit database. ATT Labs [Online]. http://yann.lecun.com/exdb/mnist **2** (2010)
13. Li M. Application of cart decision tree combined with pca algorithm in intrusion detection. In: 2017 8th IEEE International Conference on Software Engineering and Service Science (ICSESS), pp. 38–41 (2017). https://doi.org/10.1109/ICSESS.2017.8342859
14. Nway OA. Comparative study of principal component analysis (pca) based on decision tree algorithms. Int J Adv Sci Res Eng. **4**(6), 122–126 (2018). https://doi.org/10.31695/ijasre.2018.32767
15. Oliphant TE. A guide to NumPy, vol. 1. Trelgol Publishing USA (2006)
16. Pedregosa F, Varoquaux G, Gramfort A, Michel V, Thirion B, Grisel O, Blondel M, Prettenhofer P, Weiss R, Dubourg V, Vanderplas J, Passos A, Cournapeau D, Brucher M, Perrot M, Duchesnay E. Scikit-learn: Machine learning in Python. J Mach Learn Res. 2011;12:2825–30.
17. Van Rossum G, Drake FL. Python 3 Reference Manual. Scotts Valley, CA: CreateSpace; 2009.

Publisher's Note Springer Nature remains neutral with regard to jurisdictional claims in published maps and institutional affiliations.