The Max-Cut Decision Tree: Improving on the Accuracy and Running Time of Decision Trees

Jonathan Bodine and Dorit S. Hochbaum
University of California, Berkeley, U.S.A.

Keywords: Decision Tree, Principal Component Analysis, Maximum Cut, Classification.

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 involves novel modifications to a standard, baseline model of classification decision tree, precisely CART Gini. One modification involves an alternative splitting metric, Maximum Cut, which is 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 is to select the decision feature from a linear combination of the input features constructed using Principal Component Analysis (PCA) locally at each node. Our experiments show that this node-based, localized PCA with the novel splitting modification can dramatically improve classification, while also significantly decreasing computational time compared to the baseline decision tree. Moreover, our results are most significant when evaluated on data sets with higher dimensions, or more classes. For the example data set CIFAR-100, the modifications enabled a 49% improvement in accuracy, relative to CART Gini, while reducing CPU time by 94% for comparable implementations. These introduced modifications will dramatically advance the capabilities of decision trees for difficult classification tasks.

1 INTRODUCTION

Decision trees are a widely used method for classification, both alone and as the building blocks of multiple different ensemble learning methods. A standard approach for the construction of decision trees utilizes the Classification and Regression Trees (CART) method (Breiman et al., 1984). This method constructs decision trees by selecting a threshold value on some feature to separate, or split, the observations into two branches, based on some evaluation method. This process continues recursively until a preset stopping criterion is reached. Throughout our analysis, we define the stopping rule as either the node having only one class present, or no split existing such that both sides of the branch contain at least one observation.

One of the standard methods for evaluating the different splits is Gini Impurity, which was provided in the original CART methodology (Breiman et al., 1984). Gini Impurity is defined in equation 1, where p_c is the fraction of observations that belong to class c.

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

Define a *threshold* as a value in range of one of the features' values such that the observations are partitioned into two subsets, the samples whose values for that feature are less than the threshold, and those whose values are greater than the threshold. Then, a threshold is selected such that the average Gini Impurity of the resulting two subsets, weighted by their cardinality, is minimized. We will use this methodology as the *baseline* to compare our novel splitting procedure against.

Our splitting procedure is motivated by a shortcoming of Gini Impurity. The partitions generated by Gini Impurity depend only on a threshold value while ignoring the fact that some partitions are to two sets that are very dissimilar in terms of the distances between their feature vectors, whereas others are closer in terms to this distance. The more dissimilar the partition, the more meaningful it is. Given two options for a threshold that both partition the observations into the same ratios and same sizes, Gini Impurity would show no preference, regardless of the distance between these classes. We believe that this distance information is meaningful and preferable, as it produces thresholds that separate the classes further apart in space, resulting in decreased susceptibility to overfitting.

Bodine, J. and Hochbaum, D.
The Max-Cut Decision Tree: Improving on the Accuracy and Running Time of Decision Trees.
DOI: 10.5220/0010107400590070
In *Proceedings of the 12th International Joint Conference on Knowledge Discovery, Knowledge Engineering and Knowledge Management (IC3K 2020) - Volume 1: KDIR*, pages 59-70
ISBN: 978-989-758-474-9

Our splitting procedure takes inspiration from the Max Cut problem, which in general is NP-complete (Karp, 1972). Our use of Max Cut is in one dimension, based on distances with respect to a single feature, and restricted to a single threshold. It is therefore polynomial-time solvable. Indeed, in the appendix, we show a linear-time implementation given the sorted observations. The Max Cut criterion is to find θ maximizing equation 2, where $x_{i,j}$ represents the value of feature j for observation i, y_i represent the class of observation i, and θ represent the threshold value.

$$\sum_{\{i|x_{i,j}\leq\theta\}} \sum_{\{k|x_{k,j}>\theta\}} \mathbb{1}_{y_i \neq y_k} \left| x_{i,j} - x_{k,j} \right| \qquad (2)$$

Given the fact 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 of the feature vector, where n is the number of observations. It is important to note that this is the same asymptotic time complexity achieved when implementing the Gini Impurity method.

Modifying how the features are represented can have a significant impact on the performance of a decision tree. If the feature vector $\mathbf{x}_i$ is modified through a change of basis, the splits of the decision tree can be changed. This paper seeks to find a good basis to represent the feature vector. One common way to find a natural basis to represent the feature vector in is Principal Component Analysis (PCA), first proposed by Karl Pearson F.R.S. (F.R.S., 1901). 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 as a preprocessing step when utilizing decision trees is not new. However, we propose that it should instead be used 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 Features PCA and Node Means PCA (see section 2 for details).

To understand how these changes affect a decision tree's performance, we performed extensive experimental analysis on 8 different decision tree algorithms listed in Table 2, each of which uses different combinations of the methods discussed in section 2. Moreover, we implemented all of the algorithms in python to understand the relative performance between the algorithms without having different levels of code optimization effecting the results. This does mean our baseline code is not as fast as commercially available versions; however, by choosing our own implementation for the baseline we are able to minimize the differences between the code for the different implementations. This highlights differences in performance caused by the changes to the algorithm and not different implementation choices.

An analysis was conducted on over 20,000 synthetic data sets (section 3.1) with training set sizes ranging between 100 and 300,000 and testing sets fixed at 100,000 observations. We first considered binary classification problems (section 3.1.2), followed by multiclass classification problems (section 3.1.3). On the synthetic data we validated that the modification produced statistically significant improvements to the baseline Gini Impurity CART model. Similarly, these results were then validated on real-world data sets (section 3.2). These modifications were found to provide dramatic improvements in both accuracy and computational time, and these advantages further increased with the dimensionality of the data or the number of classes (section 4). The significance of these results is demonstrated here with the CIFAR-100 dataset (Krizhevsky, 2009), which has 100 classes, 3,072 dimensions, and 60,000 total observations (of which 48,000 are used for training). In this case, compared to the baseline CART Gini model, our Max Cut Node Means PCA algorithm (see section 2) resulted in a 49.4% increase in accuracy relative to the baseline while simultaneously reducing the CPU time required by 94%.

2 DESCRIPTION OF ALGORITHMS

We introduce two methods for finding locally meaningful bases. The first method that we consider is **Node Features PCA**. At every decision node, PCA is performed on the original feature vectors using only the observations that have reached that node in order 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.

The second method, referred to as **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. Like Node Features PCA, this algorithm is used at every decision node; the difference is in what points get fed into the PCA algorithm. Specifically, the first step is to think of the data set in a one-vs-rest context, locally calculating the mean position of each of

the possible 'rest' collections. For example, if there are 3 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 means, calculated by excluding the first, second, and third points respectively. These newly generated 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. The features in this new basis are then used as the input features to find the optimal threshold, instead of using the original features.

A high level description of a generic decision tree and how the specification of the split method and the feature type of basis selected generates the algorithms proposed, see Figure 1. The notation used in Fig 1 is 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 Max-Cut, 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, Global PCA, Local PCA, and Local Means PCA respectively. Let PCA(X) and PCAMapping(X) be functions that return the principal components of X or the function that maps X to its principal components respectively. Let $C(y)$ be the set of class values of vector y, $\{i|\exists y_j = i\}$, and let $U_{-i} = \{u|u \in U, y_u \neq i\}$.

In our analysis, we consider a total of 8 different algorithms, which we implemented in the Python programming language (Van Rossum and Drake, 2009). In order to improve the run time of these algorithms, both the Gini Impurity and Max Cut optimal threshold calculation for a given feature were implemented as NumPy (Oliphant, 2006) vector operations to avoid the use of slower Python for loops. Moreover, both utilize efficient $O(n \log n)$ implementations. We remark that both the Gini Impurity and Max Cut metric were implemented so that in the event of a tie, they favored more balanced splits, and the split was chosen to be the mean value between the two closest points that should be separated. Other than these subroutines, in order to make time-based comparisons as consistent as possible, no difference existed between the implementations of Gini Impurity and Max Cut based trees. The scikit-learn (Pedregosa et al., 2011) package was used to perform PCA.

As a result of Gini Impurity and Max Cut having the same time complexity, the following results on the time complexity of each algorithm hold regardless of the choice of evaluation criterion. The time complexities for computing the optimal split at each node are given in Table 1, where n is the number of


```
procedure decisionTree(U, (X,y), split, featureType):
    begin
        if type = 2 do X ← PCA(X);
        list ← {(U,X,y)};
        while list ≠ ∅ do
            pop (U,X,y) from list;
            if type = 1,2 do X' ← X;
            else if type = 3 do X' ← PCA(X);
            else do X' ← meansPCA(U,X,y);
            P1,P2 ← split(U,X',y);
            if |C(y(P1))| ≠ 1 do list ← list ∪ {(P1,X(P1),y(P1))};
            if |C(y(P2))| ≠ 1 do list ← list ∪ {(P2,X(P2),y(P2))};
    end

procedure meansPCA(U,X,y):
    begin
        means ← {};
        for each i ∈ C(y) do
            means ← means ∪ {1/|U_-i| Σ_{u∈U_-i} x_u};
        T ← PCAMapping(means);
        X' ← T(X);
        return X';
    end
```

Figure 1: High-Level description of decision tree fitting and means PCA algorithms.

observations, d is the number of input features, and p is the number of principal components considered.

Table 1: Time Complexity of Node Splitting.

Algorithm	Time Complexity
Baseline	$O(dn\log n)$
Node Features PCA	$O(n^2p)$
Node Means PCA	$O(d^2+n\log n)$

In order to be able to evaluate the relative performance of the algorithms we ran them all with the same implementation. Our baseline implementation is obviously slower than that of open source packages (such as scikit-learn (Pedregosa et al., 2011)), which have been optimized and are not implemented in Python, yet it provides a cleaner basis for comparison. The code that we used has not been made publicly available as it does not represent the levels of optimization needed for a production grade package. Our code, however, is made available upon request.

Each of the algorithms is determined by:

1. Split criterion, which is Gini or Max Cut.
2. Which feature vector type to choose: Original, Global PCA, Local PCA, or Local Means PCA.

Table 2 provides the nomenclature we use to refer to each of these different combinations. Each of these algorithms was put through the same rigorous experimentation, using both synthetic and real-world data sets.

3 A COMPARATIVE STUDY

3.1 Synthetic Data Sets

We first consider the performance of the 8 algorithms on synthetic data sets, which provide key advantages compared to real-world data sets. 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. Moreover, 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 gave us greater insights into the algorithms' performances on larger datasets than we could have achieved using only real-world data. Lastly, we had fine-grained control of the datasets' different characteristics.

3.1.1 Experimental Design

To generate the synthetic data sets, we used the datasets.make_classification tool found within the scikit-learn package (Pedregosa et al., 2011). This was designed such that only the case where each class would have one cluster was analyzed. These clusters were then centered around the vertices of a hypercube 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. Finally, the features were randomly shifted and scaled.

Throughout our experiments, we observed how the following four factors changed the accuracy, wall clock time, and the number of leaves in the tree. The factors that we controlled were:

1. The number of training examples. Note that since we were able to generate arbitrarily many data points per data set, we were able to have arbitrarily large testing set. 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 data set 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 dimensions, referred to as the Dimension of Data in our results.
4. The number of meaningless dimensions, referred to as the Noise Dimension in our results.

Table 2: Algorithms' nomenclature.

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

For each combination of parameter settings explored, we evaluated the models' results on 30 independently generated datasets. For each of these datasets, the models were presented with the same training/testing examples, which were standardized (zero mean and unit variance) using the scikit-learn StandardScaler (Pedregosa et al., 2011) 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 algorithm's performance should also be invariant.

Relying on multiple independent data sets, we can calculate the significance of our results, for each data generation setting, using a one-tailed Paired Sample T-Test. For each algorithm, j, and data generation setting we consider 14 alternative hypotheses $\{A_i\}_{i=1}^{14}$ and corresponding null hypotheses $\{N_i\}_{1=1}^{14}$. Specifically, let $\{A_i\}_{i=1}^{7}$ represent the alternative hypotheses that algorithm j out performed, in a pairwise comparison, each of the other 7 algorithms. Conversely, let $\{A_i\}_{i=8}^{14}$ represent the alternative hypotheses that algorithm j under performed, in a pairwise comparison, each of the other 7 algorithms. We then used the Holm-Bonferroni Method (Holm, 1979), to correct for multiple hypotheses testing, testing for significance at the 5% level. In the case where all $\{N_i\}_{1=1}^{7}$ were rejected we labeled that data setting on the accuracy significance graph as "Best". In the case where any of the of $\{N_i\}_{1=8}^{14}$ were rejected we 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.

3.1.2 Binary Classification

The first set of experiments served to evaluate how the accuracy and run time of each algorithm varied with the size of the training set as well as the number of informative dimensions. For this experiment, we set the number of noise dimensions to zero. The results of these experiments are summarized in Figure 2 and Figure 3.

Examining Figure 2 and 3, it is apparent that Localized PCA dramatically improves the model accuracy compared to the baseline, as well as the model with PCA applied in preprocessing. Figure 3 makes it clear that with statistical significance the baseline algorithm is not the best performing algorithm on average. On the the other hand Figure 3 indicates that the most promising algorithms 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 Figure 2 and Figure 3, there are no significant trends that 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 data sets 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 Figure 4. It is important to note that this out performance 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 1,000 training samples were used, Node Means PCA, after a small 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 algorithm searches only two basis vectors to select a threshold.

Figure 5 is the histogram all 5,640 ratios of Max Cut Node Means PCA Accuracy divided by Gini Node Features PCA Accuracy. It is evident from the right skew in figure 5 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, we consider the mean and max ratios. Conditioned on the ratio being greater than 1 (which

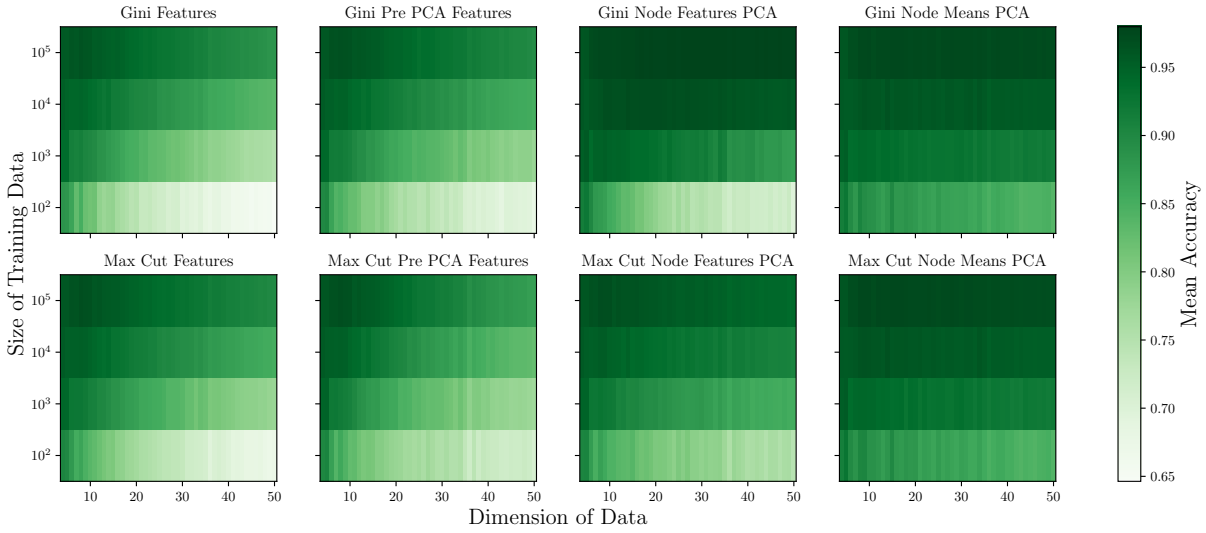

Figure 2: Mean accuracy of binary classification on synthetic data, evaluated on test sets of 100,000.

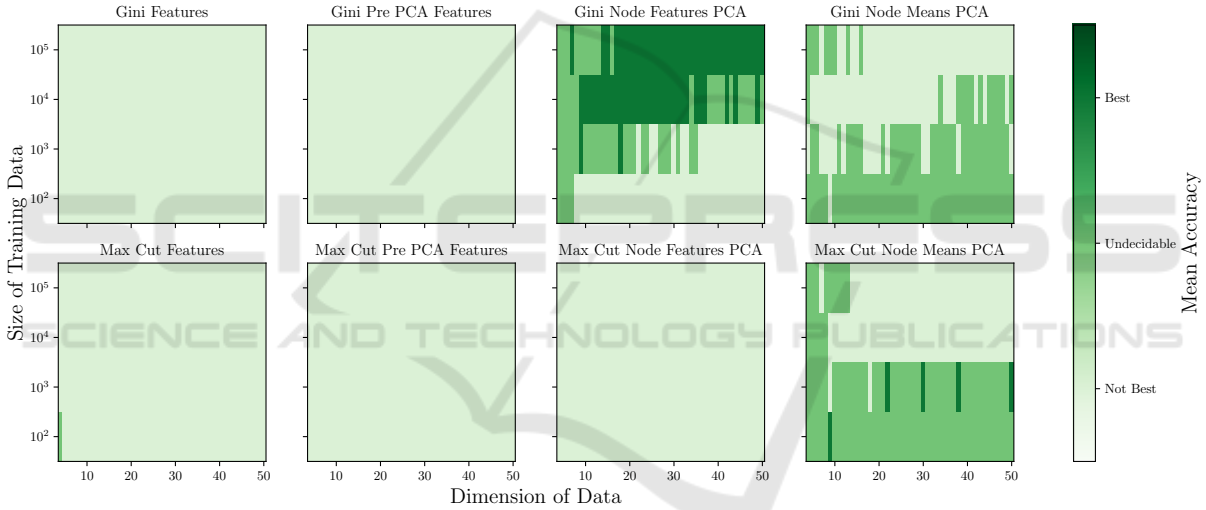

Figure 3: Significance of Mean accuracy of binary classification on synthetic data, evaluated on test sets of 100,000.

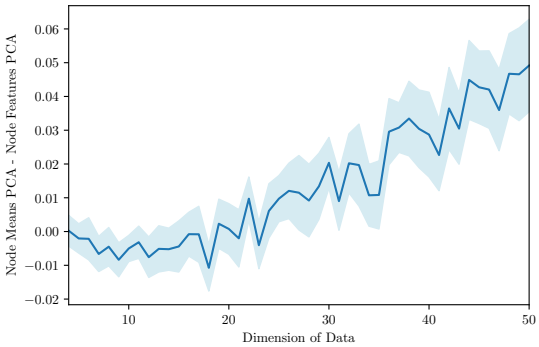

Figure 4: Mean Max Cut Node Means PCA Accuracy minus Mean Gini Node Features PCA Accuracy with 95% confidence interval using the 1,000 Training Sets.

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.

When examining the experimental results of Node Means PCA vs. Node Features PCA, we see that Node Means PCA has significantly better properties in regard to computational cost. Max Cut Node Means PCA took between 27% and 98% less time compared to Gini Node Features PCA to fit; and between 21% and 98% less time compared to the baseline model. The computational efficiency of Node Means PCA is partially explained by the computa-

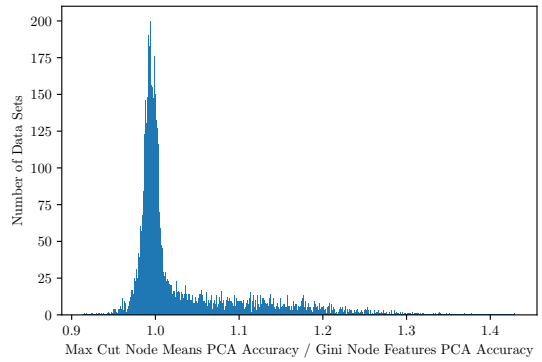

Figure 5: Histogram of Max Cut Node Means PCA Accuracy / Gini Node Features PCA Accuracy.

tional efficiency of the split as provided in table 1. However, another factor is the size of the decision tree for witch the number of leaves is a proxy. The mean number of leaves across all data sets is provided in Table 3.

Table 3: Mean Number of Leaves in Decision Tree.

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

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 when compared to the Baseline model. This reduction can help account for the significant observed computational efficiency of the Node Means PCA vs the Baseline. Therefore, taking into account the costs and benefits related to computational time, accuracy, and robustness to higher dimensional problems, the Node Means PCA algorithm has the best performance, with a slight preference towards the Max Cut metric over Gini Impurity.

The second experiment that was performed considered the addition of noise features in order to see if the Localized PCA algorithms would be negatively affected. For this experiment, the training set sizes were fixed to 10,000 and then for each combination of meaningful dimensions in $\{5, 10, 15, ..., 50\}$ and noise dimensions in $\{0, 5, 10, ..., 50\}$, we generated and analyzed 30 datasets. The major result of this experiment was that localized PCA only under performs in extreme cases. When the real dimension was five and the noise dimension was greater than or equal to 15, our modifications had no improvement upon the baseline, with the baseline actually outperforming our algorithm. However, this represents the cases where noise features accounted for 75% or more of the available features and the number of informative features was small. Therefore, we do not regard this as a fundamental issue with our algorithm.

3.1.3 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 section 3.1.2, one can see how different training set sizes, informative dimensions, and noise dimensions can affect the algorithm's performance. The main results derived from varying the training data size and the underlying dimension of the data can be seen in Figure 6 and Figure 7. 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 Figure 7, the baseline Gini Features algorithm 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 algorithm on average, much like in the binary case (section 3.2). The "Best" markings actually indicate, from an accuracy standpoint, that Max Cut Nodes Means PCA was the best option in most cases. Moreover, 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, but we do not view this as a fundamental issue. In general, lower dimensional problems are computationally faster, so the increased relative time does not translate to significantly more realized computational time.

Upon close inspection of Figure 7, the markings indicate that the Gini Node Features algorithm may be the best option in terms of accuracy for larger training sets and fewer dimensions. This is the same pattern that emerged in the binary case, and Figure 8 allows for a closer comparison of these two algorithms, 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 dimensions increases.

When the comparison is made between the Max Cut Node Means PCA algorithm and the Gini Node Means PCA algorithm, Figure 9 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 al-

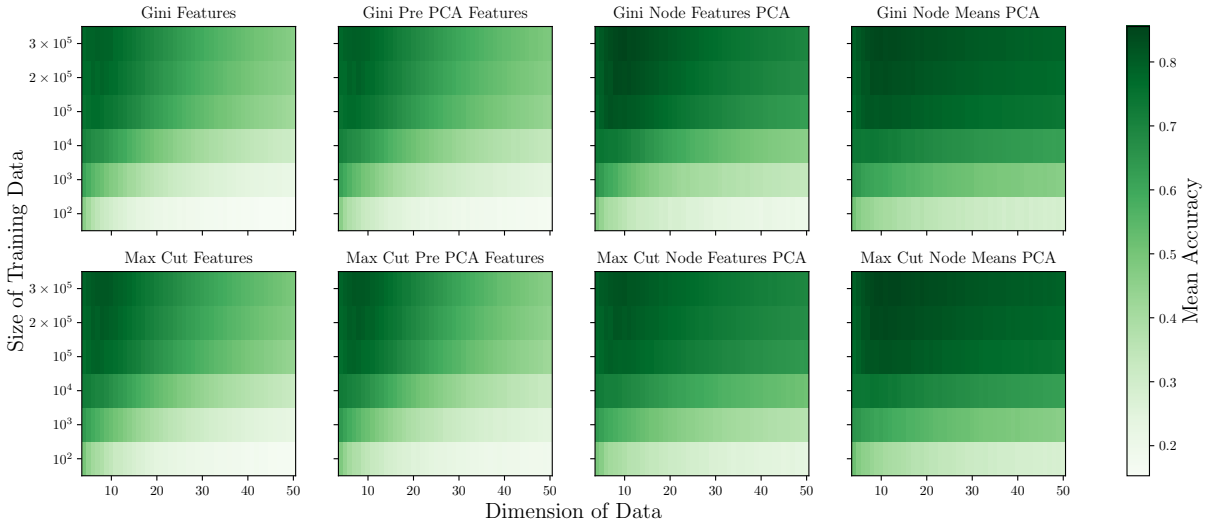

Figure 6: Mean accuracy of multiclass classification on synthetic data, evaluated on test sets of 100,000.

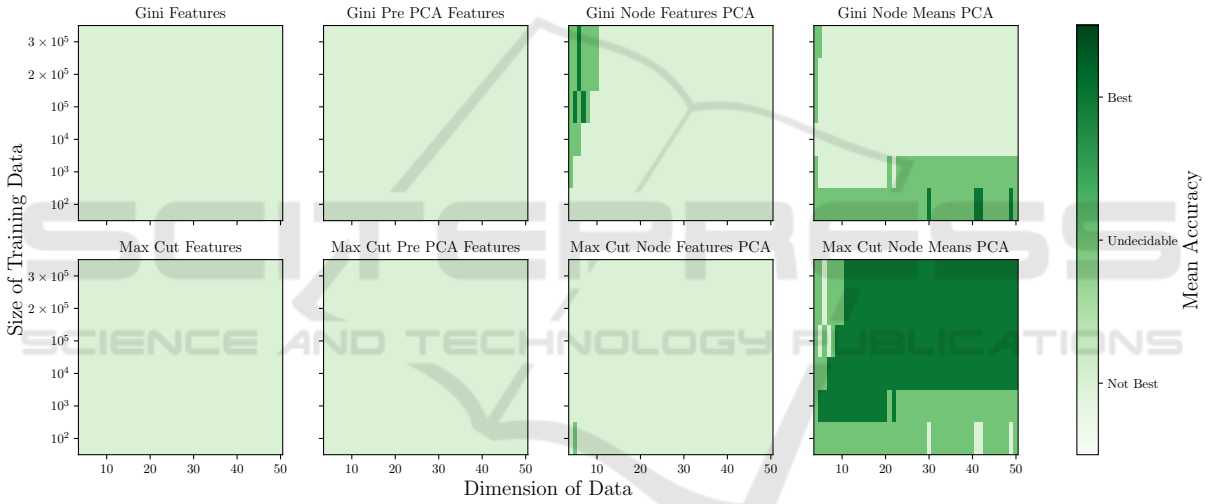

Figure 7: Significance of Mean accuracy of multiclass classification on synthetic data, evaluated on test sets of 100,000.

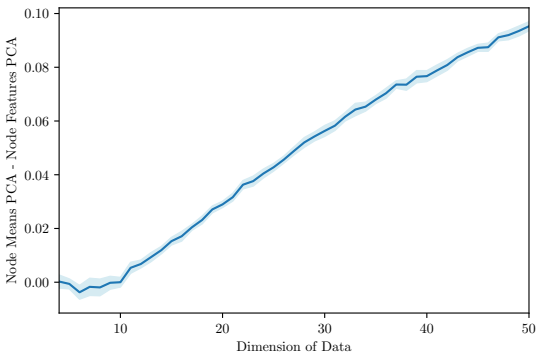

Figure 8: Mean Max Cut Node Means PCA Accuracy minus Mean Gini Node Features PCA Accuracy with 95% confidence interval using the 300,000 Training Sets.

gorithm and the Gini Node Means PCA algorithm 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 algorithms were affected by the inclusion of noise features, the same results as in the binary analysis are seen. That is, for low dimensional data (informative dimensions of 5 and 10) and high amounts of noise (noise dimensions greater than or equal to 5 and 25 respectively), the baseline model has better performance. However, these cases have very high percentages of noise features (greater than 50%) and few dimensions. Since the improved accuracy of Max Cut Node Means PCA still held for higher dimensions, even when the percentage of noise features exceeded 50%, we believe that our Max Cut Node Means PCA

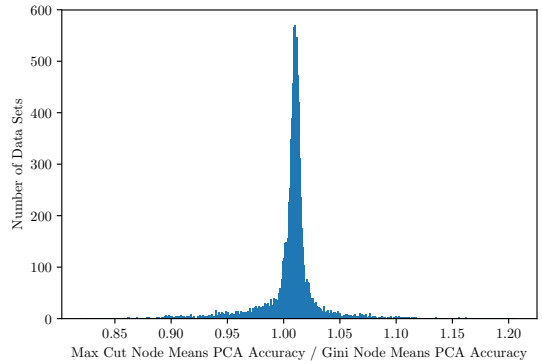

Figure 9: Histogram of Max Cut Node Means PCA Accuracy / Gini Node Means PCA Accuracy.

algorithm is broadly applicable.

3.1.4 Synthetic Dataset Results Summary

After extensive testing, it is apparent that our localized PCA algorithms provide dramatic improvements over the baseline model in all but the most extreme noise conditions. A thorough analysis of these localized PCA algorithms found that from an accuracy standpoint, Node Means PCA should be preferred for increasingly difficult classification tasks, particularly in cases where there are more classes, higher dimensions, 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 end up dramatically improving results. Empirically, Node Means PCA results in significantly faster computation times compared to the baseline method and the Node Features method. 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 algorithm, on average, it can increase a decision tree's performance. 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 model. We have thus shown that for synthetic datasets Max Cut Node Means PCA yields a significant and dramatic improvement over the baseline model in both accuracy and computational efficiency. We present next the performance of our algorithm for real-world datasets.

3.2 Real-world Data Sets

We proceeded to explore the effects of the eight different algorithms on real-world datasets, considering a total of 5 different datasets. We used the Iris Data Set and Wine Quality Data Set (Cortez et al., 2009), both taken from the UCI Machine Learning Repository (Dua and Graff, 2017). The Wine Quality Data Set is separated into 2 sets, red and white wines, and in our experimentation, we consider each of these sets as a single combined dataset and as individual sets. In the combined dataset, the wine's classification was included as another feature, one if red, zero otherwise. We also used the MNIST dataset (LeCun et al., 2010), as well as the CIFAR-10 and CIFAR-100 datasets (Krizhevsky, 2009). Table 4 provides a brief overview of each of these datasets.

Table 4: Data Set Characteristics.

Data Set	Samples	Features	Classes
Iris	150	4	3
Wine-Red	1,599	11	6
Wine-White	4,898	11	7
Wine-Both	6,497	12	7
MNIST	70,000	784	10
CIFAR-10	60,000	3,072	10
CIFAR-100	60,000	3,072	100

All of 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. Rather than recording wall clock time as in the previous synthetic data set experiments, the total CPU time was used for these experiments.

We considered each algorithm with modified features as well as with standardized features in the initial input and reported the best result. To evaluate the performance of each algorithm we used one of two methods: either 10×10 cross-validation (as in the case of Iris, 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 to not use 10×10 cross-validation for MNIST, CIFAR-10, and CIFAR-100 was a result of 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, Wine-Red, Wine-White, and Wine-Both datasets.

Table 5: Results of Real-World Data Set Experiments.

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

The results of the real-world analysis verify that the improvements in accuracy are similar to those in the synthetic datasets. In the case of the Node Means PCA modification, we found that this was the best option to use for all the problems analyzed. Defining performance as the percentage improvement in accuracy relative to the baseline model (utilizing the better-performing metric between Max Cut and Gini), this modification had improvements of 1.6%, 1.9%, 1.9%, 1.9%, 6.3% 33.3%, and 49.4% for each of the data sets. Moreover, Max Cut was the best choice in 5 out of 7 problems. Defining performance as the percentage improvement of the best Max Cut algorithm compared to the best performing Gini algorithm, Max Cut had improvements of 1.1% -0.3%, 0.0%, 0.5%, 0.2% 2.0%, and 6.9%. Finally, Max Cut Node Means PCA was the best choice in 5 out of 7 problems and was tied for the best in 1 problem. Defining performance as the percentage improvement of accuracy compared to the baseline model, Max Cut Node Means PCA represented an improvement of 1.6%, 1.0%, 1.9%, 1.9%, 6.3%, 33.3%, and 49.4%.

From Table 5, it can be seen that Max Cut Node Means PCA is invariably the fastest algorithm for larger problems. Although Max Cut Node Means PCA took more time for small datasets, we do not be-

lieve this discounts its overall performance. Regardless of the algorithm 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 algorithm reduced the required CPU time by 94% compared to the baseline algorithm. Thus, Max Cut Node Means PCA provides significant computational and accuracy advantages.

4 CONCLUSIONS

In this paper, we propose 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 the use of localized PCA to determine locally viable directions for considering splits and further modified the traditional PCA algorithm, Means PCA, to find suitable directions for discriminating between classes. We follow this study with a theoretical commentary on how these modifications are reflected in the asymptotic bounds of node splitting and demonstrate that Means PCA improves upon the traditional method.

Our extensive experimental analysis included more than 20,000 synthetically generated data sets with training sizes ranging from 100 and 300,000 and informative dimensions ranging between 4 and 50, We further considered both binary and multiclass classification tasks. These experiments demonstrate the significant improvements in increased accuracy and decreased computational time provided by utilizing localized Means PCA and Max Cut. 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 algorithm remains advantageous even when using real-world data. For example, we show that in the case of CIFAR-100 (100 classes, 3,072 dimensions, and 48,000 training points out of 60,000 total) our algorithm results in a 49.4% increase in accuracy performance compared to the baseline CART model, while simultaneously reducing the CPU time required by 94%. This novel algorithm helps bring decision trees into the world of big, high-dimensional data. Our experiments demonstrate the significant improvements that Max Cut Node Means PCA 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.

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). This research was supported in part by NSF award No. CMMI-1760102.

REFERENCES

Breiman, L., Friedman, J., Stone, C., and Olshen, R. (1984). *Classification and Regression Trees*. The Wadsworth and Brooks-Cole statistics-probability series. Taylor & Francis.

Cortez, P., Cerdeira, A., Almeida, F., Matos, T., and Reis, J. (2009). Modeling wine preferences by data mining from physicochemical properties. *Decis. Support Syst.*, 47:547–553.

Dua, D. and Graff, C. (2017). UCI machine learning repository.

F.R.S., K. P. (1901). Liii. on lines and planes of closest fit to systems of points in space. *The London, Edinburgh, and Dublin Philosophical Magazine and Journal of Science*, 2(11):559–572.

Holm, S. (1979). A simple sequentially rejective multiple test procedure. *Scandinavian Journal of Statistics*, 6(2):65–70.

Karp, R. M. (1972). Reducibility among combinatorial problems. In Miller, R. E., Thatcher, J. W., and Bohlinger, J. D., editors, *Complexity of Computer Computations*, pages 85–103. Plenum Press, New York.

Krizhevsky, A. (2009). Learning multiple layers of features from tiny images. Technical report.

LeCun, Y., Cortes, C., and Burges, C. (2010). Mnist handwritten digit database. *ATT Labs [Online]. Available: http://yann.lecun.com/exdb/mnist*, 2.

Oliphant, T. E. (2006). *A guide to NumPy*, volume 1. Trelgol Publishing USA.

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., and Duchesnay, E. (2011). Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12:2825–2830.

Van Rossum, G. and Drake, F. L. (2009). *Python 3 Reference Manual*. CreateSpace, Scotts Valley, CA.

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:

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

Since $\mathbb{1}_{y_i \neq y_i} = 0\ \forall i$:

$$\begin{aligned}
&= \theta_{i-1} + \sum_{t} \mathbb{1}_{y_i \neq y_t} x_{t,j} - \mathbb{1}_{y_i \neq y_t} x_{i,j} \\
&= \theta_{i-1} + \left(\sum_{t} \mathbb{1}_{y_i \neq y_t} x_{t,j} \right) - x_{i,j} \left(\sum_{t} \mathbb{1}_{y_i \neq y_t} \right)
\end{aligned}$$

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 $O(n\log n)$.