

# ExKMC: Expanding Explainable $k$ -Means Clustering

Nave Frost  
Tel Aviv University  
navefrost@mail.tau.ac.il

Michal Moshkovitz  
University of California, San Diego  
mmoshkovitz@eng.ucsd.edu

Cyrus Rashtchian  
University of California, San Diego  
crashtchian@eng.ucsd.edu

## Abstract

Despite the popularity of explainable AI, there is limited work on effective methods for unsupervised learning. We study algorithms for  $k$ -means clustering, focusing on a trade-off between explainability and accuracy. Following prior work, we use a small decision tree to partition a dataset into  $k$  clusters. This enables us to explain each cluster assignment by a short sequence of single-feature thresholds. While larger trees produce more accurate clusterings, they also require more complex explanations. To allow flexibility, we develop a new explainable  $k$ -means clustering algorithm, **ExKMC**, that takes an additional parameter  $k' \geq k$  and outputs a decision tree with  $k'$  leaves. We use a new surrogate cost to efficiently expand the tree and to label the leaves with one of  $k$  clusters. We prove that as  $k'$  increases, the surrogate cost is non-increasing, and hence, we trade explainability for accuracy. Empirically, we validate that **ExKMC** produces a low cost clustering, outperforming both standard decision tree methods and other algorithms for explainable clustering. Implementation of **ExKMC** available at <https://github.com/navefr/ExKMC>.

## 1 Introduction

The bulk of research on explainable machine learning studies how to interpret the decisions of supervised learning methods, largely focusing on feature importance in black-box models [3, 25, 42, 46, 50, 51, 56, 57]. To complement these efforts, we study explainable algorithms for clustering, a canonical example of unsupervised learning. Most clustering algorithms operate iteratively, using global properties of the data to converge to a low-cost solution. For center-based clustering, the best explanation for a cluster assignment may simply be that a data point is closer to some center than any others. While this type of explanation provides some insight into the resulting clusters, it obscures the impact of individual features, and the cluster assignments often depend on the dataset in a complicated way.

Recent work on explainable clustering goes one step further by enforcing that the clustering be derived from a binary threshold tree [11, 18, 22, 28, 32, 43]. Each node is associated with a feature-threshold pair that recursively splits the dataset, and labels on the leaves correspond to clusters. Any cluster assignment can be explained by a small number of thresholds, each depending on a single feature. For large, high-dimensional datasets, this provides more information than typical clustering methods.

To make our study concrete, we focus on the  $k$ -means objective. The goal is to find  $k$  centers that approximately minimize the sum of the squared distances between  $n$  data points in  $\mathbb{R}^d$  and their nearest center [1, 2, 4, 21, 37, 52]. In this context, Dasgupta, Frost, Moshkovitz, and Rashtchian have studied the

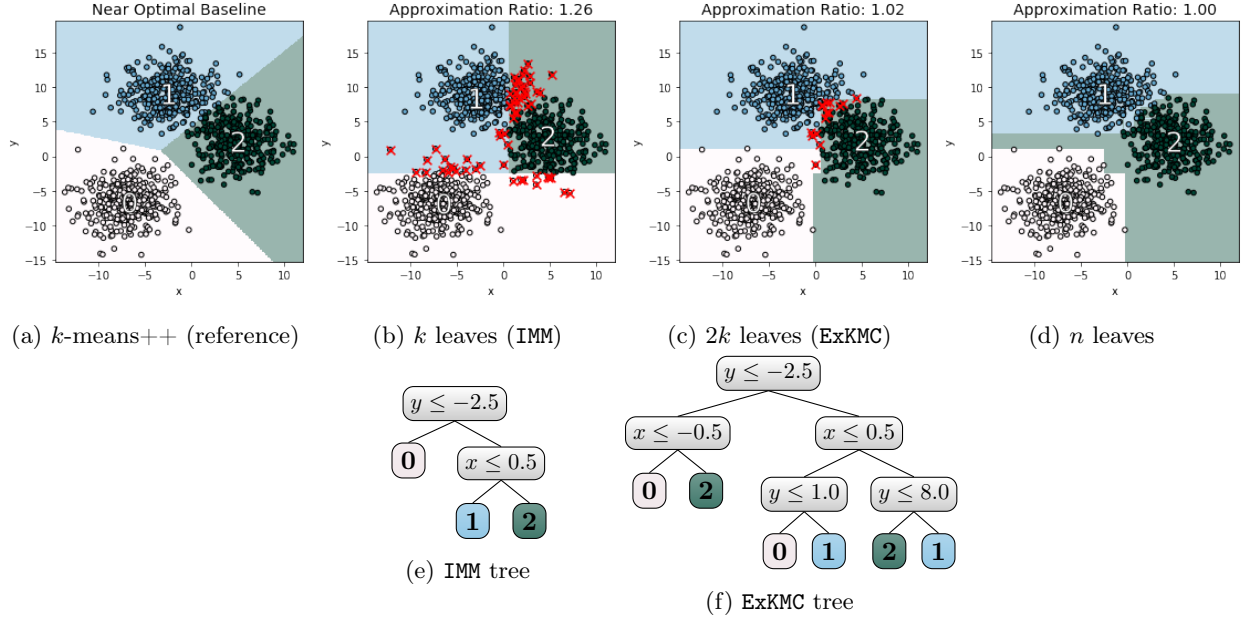


Figure 1: Tree size (explanation complexity) vs.  $k$ -means clustering quality.

use of a small threshold tree to specify the cluster assignments, exhibiting the first explainable  $k$ -means clustering algorithm with provable guarantees [22]. They propose the Iterative Mistake Minimization (IMM) algorithm and prove that it achieves a worst-case  $O(k^2)$  approximation to the optimal  $k$ -means clustering cost. However, the IMM algorithm and analysis are limited to trees with exactly  $k$  leaves (the same as the number of clusters). They also prove a lower bound showing that an  $\Omega(\log k)$  approximation is the best possible when restricted to trees with at most  $k$  leaves.

Our goal is to improve upon two shortcomings of previous work by (i) providing an experimental evaluation of their algorithms and (ii) exploring the impact of using more leaves in the threshold tree. We posit that on real datasets it should be possible to find a nearly-optimal clustering of the dataset. In other words, the existing worst-case analysis may be very pessimistic. Furthermore, we hypothesize that increasing the tree size should lead to monotonically decreasing the clustering cost.

As our main contribution, we propose a novel extension of the previous explainable  $k$ -means algorithm. Our method, ExKMC, takes as input two parameters  $k, k'$  and a set  $\mathcal{X} \subseteq \mathbb{R}^d$  with  $|\mathcal{X}| = n$ . It first builds a threshold tree with  $k$  leaves using the IMM algorithm [22]. Then, given a budget of  $k' > k$  leaves, it greedily expands the tree to reduce the clustering cost. At each step, the clusters form a refinement of the previous clustering. By adding more thresholds, we gain more flexibility in the data partition, and we also allow multiple leaves to correspond to the same cluster (with  $k$  clusters total).

To efficiently determine the assignment of leaves to clusters, we design and analyze a surrogate cost. Recall that the IMM algorithm first runs a standard  $k$ -means algorithm, producing a set of  $k$  centers that are given as an additional input [22]. As ExKMC expands the tree to  $k' > k$  leaves, it minimizes the cost of the current clustering compared to these  $k$  reference centers. We assign each leaf a label based on the best reference center, which determines the cluster assignments. The main idea is that by fixing the centers between steps, we can more efficiently determine the next feature-threshold pair to add. We prove that the surrogate cost is non-increasing throughout the execution as the number of leaves  $k'$  grows. When  $k' = n$ , then the  $k$ -means cost matches that of the reference clustering.

Figure 1 depicts the improvement from using more leaves. The left picture shows a near-optimal 3-means clustering. Next, the IMM algorithm with  $k = 3$  leaves leads to a large deviation from the reference clustering. Extending the tree to use  $2k = 6$  leaves with ExKMC leads to a lower-cost result that better approximates the reference clustering. We form three clusters by subdividing the previous clusters and mapping multiple leaves to the same cluster. Finally, trees with an arbitrary number of leaves can perfectly fit the reference clustering. Appendix B contains the trees for 1(b) and 1(c).

To test our new algorithm, we provide an extensive evaluation on many real and synthetic datasets. We show that ExKMC often has lower  $k$ -means cost than several baselines. We also find that prior IMM analysis is pessimistic because their algorithm achieves clustering cost within 5–30% versus standard  $k$ -means algorithms. Next, we explore the effect of using more than  $k$  leaves. On eleven datasets, we show that threshold trees with between  $2k$  and  $4k$  leaves actually suffice to get within 1–2% of the cost of a standard  $k$ -means implementation. The only outlier is CIFAR-10, where we conjecture that pixels are insufficient features to capture the clustering. Overall, we verify that it is possible to find an explainable clustering with high accuracy, while using only  $O(k)$  leaves for  $k$ -means clustering.

## 1.1 Related Work

We address the challenge of obtaining a low cost  $k$ -means clustering using a small decision tree. Our approach has roots in previous works on clustering with unsupervised decision trees [7, 11, 16, 22, 23, 28, 43, 61] and in prior literature on extending decision trees for tasks beyond classification [29, 30, 35, 45, 60, 54, 55].

Besides the IMM algorithm [22], prior explainable clustering algorithms optimize different objectives than the  $k$ -means cost, such as the Silhouette metric [11], density measures [43], or interpretability scores [58]. Most similar to our approach, a localized version of the 1-means cost has been used for greedily splitting nodes when growing the tree [28, 32]. We compare ExKMC against two existing methods: CUBT [28] and CLTree [43]. We also compare against KDTree [9] and, after generating the cluster labels, the standard decision tree classification method CART [15].

Clustering via trees is explainable by design. We contrast this with the indirect approach of clustering with a neural network and then explaining the network [38]. A generalization of tree-based clustering has been studied using rectangle-based mixture models [18, 17, 53]. Their focus differs from ours as they consider including external information, such as prior knowledge, via a graphical model and performing inference with variational methods. Clustering after feature selection [13, 20] or feature extraction [8, 14, 48] reduces the number of features, but it does not lead to an explainable solution because it requires running a non-explainable  $k$ -means algorithm on the reduced feature space.

## 1.2 Our contributions

We present a new algorithm, ExKMC, for explainable  $k$ -means clustering with the following properties:

**Explainability-accuracy trade-off.** We provide a simple method to expand a threshold tree with  $k$  leaves into a tree with a specified number of leaves. At each step, we aim to better approximate a given reference clustering (such as from a standard  $k$ -means implementation). The key idea is to minimize a surrogate cost that is based on the reference centers instead of using the  $k$ -means cost directly. By doing so, we efficiently expand the tree while obtaining a good  $k$ -means clustering.

**Convergence.** We demonstrate empirically that ExKMC quickly converges to the reference clustering as the number of leaves increases. On many datasets, the cost ratio between our clustering and the reference clustering goes to 1.0 as the number of leaves goes from  $k$  to  $4k$ , where  $k$  is the number of labels for

classification datasets. In theory, we prove that the surrogate cost is non-increasing throughout the execution of ExKMC, verifying that we trade explainability for clustering accuracy.

**Low cost.** On a dozen datasets, our algorithm often achieves a lower  $k$ -means cost for a given number of leaves compared to four baselines: CUBT [28], CLTree [43], KDTree [9], and CART [44].

**Speed.** Using only standard optimizations (e.g., dynamic programming), ExKMC can cluster fairly large datasets (e.g., CIFAR-10 or covtype) in under 15 minutes using a single processor, making it a suitable alternative to standard  $k$ -means in data science pipelines. The main improvement comes from using the surrogate cost, instead of the  $k$ -means cost, to determine the cluster assignments.

## 2 Preliminaries

We let  $[n] = \{1, 2, \dots, n\}$ . For  $k \geq 1$ , a  $k$ -clustering refers to a partition of a dataset into  $k$  clusters. Let  $C^1, \dots, C^k$  be a  $k$ -clustering of  $\mathcal{X} \subseteq \mathbb{R}^d$  with  $|\mathcal{X}| = n$  and  $\mu^j = \text{mean}(C^j)$ . The  $k$ -means cost is  $\sum_{j=1}^k \sum_{\mathbf{x} \in C^j} \|\mathbf{x} - \mu^j\|_2^2$ . It is NP-hard to find the optimal  $k$ -means clustering [2, 21] or a close approximation [5]. Standard algorithms for  $k$ -means are global and iterative, leading to complicated clusters that depend on the data in a hard to explain way [1, 4, 37, 52].

**Explainable clustering.** Let  $T$  be a binary threshold tree with  $k' \geq k$  leaves, where each internal node contains a single feature  $i \in [d]$  and threshold  $\theta \in \mathbb{R}$ . We also consider a labeling function  $\ell : \text{leaves}(T) \rightarrow [k]$  that maps the leaves of  $T$  to clusters. The pair  $(T, \ell)$  induces a  $k$ -clustering of  $\mathcal{X}$  as follows. First,  $\mathcal{X}$  is partitioned via  $T$  using the feature-threshold pairs on the root-to-leaf paths. Then, each point  $\mathbf{x} \in \mathcal{X}$  is assigned to one of  $k$  clusters according to how  $\ell$  labels its leaf. Geometrically, the clusters reside in cells bounded by axis-aligned cuts, where the number of cells equals the number of leaves. This results in a  $k$ -clustering  $\hat{C}^1, \dots, \hat{C}^k$  with means  $\hat{\mu}^j = \text{mean}(\hat{C}^j)$ , and we denote the  $k$ -means cost of the pair  $(T, \ell)$  as  $\text{cost}(T, \ell) = \sum_{j=1}^k \sum_{\mathbf{x} \in \hat{C}^j} \|\mathbf{x} - \hat{\mu}^j\|_2^2$ .

**Iterative Mistake Minimization (IMM).** Previous work has exhibited the IMM algorithm that produces a threshold tree with  $k$  leaves [22]. It first runs a standard  $k$ -means algorithm to find  $k$  centers. Then, it iteratively finds the best feature-threshold pair to partition the data into two parts. At each step, the number of mistakes is minimized, where a mistake occurs if a data point is separated from its center. Each partition also enforces that at least one center ends up in both children, so that the tree terminates with exactly  $k$  leaves. Each leaf contains one center at the end, and the clusters are assigned based this center. The IMM algorithm provides a  $O(k^2)$  approximation to the optimal  $k$ -means cost, assuming that a constant-factor approximation algorithm generates the initial  $k$  centers.

We use the IMM algorithm to build a tree with  $k$  leaves as the initialization for our algorithm. Then, we smoothly trade explainability for accuracy by expanding the tree to have  $k' > k$  leaves, for a user-specified parameter  $k'$ . More formally, we develop an algorithm to solve the following problem:

**Problem Formulation.** Given a dataset  $\mathcal{X} \subseteq \mathbb{R}^d$  and parameters  $k, k'$  with  $k' \geq k$ , the goal is to efficiently construct a binary threshold tree  $T$  with  $k'$  leaves and a function  $\ell : \text{leaves}(T) \rightarrow [k]$  such that  $(T, \ell)$  induces a  $k$ -clustering of  $\mathcal{X}$  with as small  $k$ -means cost as possible.

## 3 Our Algorithm

We describe our explainable clustering algorithm, ExKMC, that efficiently finds a tree-based  $k$ -clustering of a dataset. Starting with a base tree (either empty or from an existing algorithm like IMM), ExKMC expands

the tree by replacing a leaf node with two new children. In this way, it refines the clustering, while allowing the new children to be mapped to different clusters. A key optimization is to use a new surrogate cost to determine both the best threshold cut and the labeling of the leaves. At the beginning, we run a standard  $k$ -means algorithm and generate  $k$  reference centers. Then, the surrogate cost is the  $k$ -means cost if the centers were the reference centers. By fixing the centers, instead of changing them at every step, we determine the cluster label for each leaf independently (via the best reference center). For a parameter  $k'$ , our algorithm terminates when the tree has  $k'$  leaves.

### 3.1 Surrogate cost

Our starting point is a set of  $k$  reference centers  $\mu^1, \dots, \mu^k$ , obtained from a standard  $k$ -means algorithm. This induces a clustering with low  $k$ -means cost. While it is possible to calculate the actual  $k$ -means cost as we expand the tree, it is difficult and time-consuming to recalculate the distances to a dynamic set of centers. Instead, we fix the  $k$  reference centers, and we define the surrogate cost as the sum of squared distances between points and their closest reference center.

**Definition 1** (Surrogate cost). *Given centers  $\mu^1, \dots, \mu^k$  and a threshold tree  $T$  that defines the clustering  $(\widehat{C}^1, \dots, \widehat{C}^{k'})$ , the surrogate cost is defined as*

$$\widetilde{\text{cost}}^{\mu^1, \dots, \mu^k}(T) = \sum_{j=1}^{k'} \min_{i \in [k]} \sum_{\mathbf{x} \in \widehat{C}^j} \|\mathbf{x} - \mu^i\|_2^2.$$

The difference between the new surrogate cost and the  $k$ -means cost is that the centers are *fixed*. In contrast, the optimal  $k$ -means centers are the means of the clusters, and therefore, they would change throughout the execution of the algorithm. Before we present our algorithm in detail, we mention that at each step the goal will be to expand the tree by splitting an existing leaf into two children. To see the benefit of the surrogate cost, consider choosing a split at each step that minimizes the actual  $k$ -means cost of the new clustering. This requires time  $\Omega(dkn)$  for each possible split because we must iterate over the entire dataset to calculate the new cost. In Appendix 3.2, we provide the detailed argument showing that ExKMC only takes time  $O(dkn_v)$  at each step, which is an improvement as the number of surviving points  $n_v$  in a node often decreases rapidly as the tree grows.

The surrogate cost has several further benefits. The best reference center for a cluster is independent of the other cluster assignments. This independence makes the calculation of  $\widetilde{\text{cost}}$  more efficient, as there is no need to recalculate the entire cost if some of the points are added or removed from a cluster. The surrogate cost is also an upper bound on the  $k$ -means cost. Indeed, it is known that using the mean of a cluster as its center can only improve the  $k$ -means cost. In Section 3.3, we provide theoretical guarantees, such as showing that our algorithm always converges to the reference clustering as the number of leaves increases. Finally, in Section 4, we empirically show that minimizing the surrogate cost still leads to a low-cost clustering, nearly matching the reference cost. We now utilize this idea to design a new algorithm for explainable clustering.

Algorithm 1 describes the ExKMC algorithm, which uses subroutines in Algorithm 2. It takes as input a value  $k$ , a dataset  $\mathcal{X}$ , and a number of leaves  $k' \geq k$ . The first step will be to generate  $k$  reference centers  $\mu^1, \dots, \mu^k$  from a standard  $k$ -means implementation and to build a threshold tree  $T$  with  $k$  leaves (for evaluation,  $T$  is the output of the IMM algorithm). For simplicity, we refer to these as inputs as well. ExKMC outputs a tree  $T'$  and a labeling  $\ell : \text{leaves}(T') \rightarrow [k]$  that assigns leaves to clusters. Notably, the clustering induced by  $(T', \ell)$  always refines the one from  $T$ .

---

**ALGORITHM 1: ExKMC: EXPANDING  
EXPLAINABLE  $k$ -MEANS CLUSTERING**


---

**Input** :  $\mathcal{X}$  – Set of vectors in  $\mathbb{R}^d$   
 $\mathcal{M}$  – Set of  $k$  reference centers  
 $T$  – Base tree  
 $k'$  – Number of leaves

**Output** : Labeled tree with  $k'$  leaves

```

1 splits  $\leftarrow$  dict()
2 gains  $\leftarrow$  dict()
3 foreach leaf  $\in T.leaves$  do
4   add_gain(leaf,  $\mathcal{X}$ ,  $\mathcal{M}$ , splits, gains)
5 while  $|T.leaves| < k'$  do
6   leaf  $\leftarrow \arg \max_{\text{leaf}} \text{gains}[\text{leaf}]$ 
7    $i, \theta \leftarrow \text{splits}[\text{leaf}]$ 
8    $\mu^L, \mu^R \leftarrow \text{find\_labels}(\mathcal{X}, \mathcal{M}, i, \theta)$ 
9   leaf.condition  $\leftarrow "x_i \leq \theta"$ 
10  leaf.l  $\leftarrow$  new Leaf(label =  $\mu^L$ )
11  leaf.r  $\leftarrow$  new Leaf(label =  $\mu^R$ )
12  add_gain(leaf.l,  $\mathcal{X}$ ,  $\mathcal{M}$ , splits, gains)
13  add_gain(leaf.r,  $\mathcal{X}$ ,  $\mathcal{M}$ , splits, gains)
14  delete(splits[leaf], gains[leaf])
15 return  $T$ 
```

---



---

**ALGORITHM 2: SUBROUTINES**


---

```

1 add_gain(leaf,  $\mathcal{X}$ ,  $\mathcal{M}$ , splits, gains):
2    $\mathcal{X}_l \leftarrow \{\mathbf{x} \in \mathcal{X} \mid \mathbf{x} \text{ path ends in leaf}\}$ 
3    $i, \theta \leftarrow \arg \min_{i, \theta} \text{split\_cost}(\mathcal{X}_l, \mathcal{M}, i, \theta)$ 
4   best_cost  $\leftarrow \text{split\_cost}(\mathcal{X}_l, \mathcal{M}, i, \theta)$ 
5   splits[leaf]  $\leftarrow (i, \theta)$ 
6   gains[leaf]  $\leftarrow \widetilde{\text{cost}}_{\mathcal{M}}(\mathcal{X}_l) - \text{best\_cost}$ 

1 split_cost( $\mathcal{X}$ ,  $\mathcal{M}$ ,  $i, \theta$ ):
2    $\mathcal{X}_L \leftarrow \{\mathbf{x} \in \mathcal{X} \mid x_i \leq \theta\}$ 
3    $\mathcal{X}_R \leftarrow \{\mathbf{x} \in \mathcal{X} \mid x_i > \theta\}$ 
4   return  $\widetilde{\text{cost}}_{\mathcal{M}}(\mathcal{X}_L) + \widetilde{\text{cost}}_{\mathcal{M}}(\mathcal{X}_R)$ 

1 find_labels( $\mathcal{X}$ ,  $\mathcal{M}$ ,  $i, \theta$ ):
2    $\mu^L \leftarrow \arg \min_{\mu \in \mathcal{M}} \sum_{\mathbf{x} \in \mathcal{X}: x_i \leq \theta} \|\mathbf{x} - \mu\|_2^2$ 
3    $\mu^R \leftarrow \arg \min_{\mu \in \mathcal{M}} \sum_{\mathbf{x} \in \mathcal{X}: x_i > \theta} \|\mathbf{x} - \mu\|_2^2$ 
4   return  $\mu^L, \mu^R$ 
```

---

**Initialization.** In line 3, we first compute and store the gain of the  $k$  leaves in  $T$ . The *gain* of a split is the difference between the cost with split and without. The details are in the subroutine **add\_gain**. It stores the best feature-threshold pair and cost improvement in **splits** and **gains**, respectively.

**Growing the tree.** We expand the tree by splitting the node with largest gain in Line 6. We use best reference center from  $\mathcal{M}$  for each of its two children, using the subroutine **find\_labels**. In Lines 12–14, we update the lists **splits** and **gains**. At the final step, we create a tree  $T'$  with  $k'$  labeled leaves, where the implicit labeling function  $\ell$  maps a leaf to the lowest cost reference center.

### 3.2 Speeding up ExKMC

The running time is dominated by finding the best split at each step. Fortunately, this can be executed in  $O(dkn_v + dn_v \log n_v)$  time where  $n_v$  is the number of points surviving at node  $v$ . The term  $O(n_v \log n_v)$  comes from sorting the points in each coordinate. Then, we go over all splits  $(i, \theta)$  and find the one that minimizes  $\sum_{\mathbf{x} \in C^L} \|\mathbf{x} - \mu^L\|_2^2 + \sum_{\mathbf{x} \in C^R} \|\mathbf{x} - \mu^R\|_2^2$ , where  $C^L$  contains all points in  $v$  where  $x_i \leq \theta$  and  $\mu^L$  is the best center for  $C^L$  among the  $k$  reference centers (cluster  $C^R$  and center  $\mu^R$  are defined similarly for points  $x_i > \theta$ ). At first glance, it seems that the running time is  $O(d^2kn_v^2)$ , where  $dn_v$  is the number of possible splits,  $k$  is the possible value of  $\mu^L$  (and  $\mu^R$ ), and  $O(dn_v)$  is the time to calculate the cost of each split and center value. To improve the running time we can rewrite the cost as

$$\sum_{\mathbf{x} \in C^L \cup C^R} \|\mathbf{x}\|_2^2 - 2 \langle \sum_{\mathbf{x} \in C^L} \mathbf{x}, \mu^L \rangle + |C^L| \|\mu^L\|_2^2 - 2 \langle \sum_{\mathbf{x} \in C^R} \mathbf{x}, \mu^R \rangle + |C^R| \|\mu^R\|_2^2.$$

Since we care about minimizing the last expression, we can ignore the term  $\sum_{\mathbf{x} \in C^L \cup C^R} \|\mathbf{x}\|_2^2$  as it is independent of the identity of the split. Using dynamic programming, we go over all splits and save  $\sum_{\mathbf{x} \in C^L} \mathbf{x}$  and  $\sum_{\mathbf{x} \in C^R} \mathbf{x}$ . An update then only takes  $O(dk)$  time, reducing the total time to  $O(d^2kn_v + dn_v \log n_v)$ . When

the dimensionality  $d$  is large, we can use an additional improvement by saving  $\langle \mathbf{x}, \boldsymbol{\mu} \rangle$  for each  $\mathbf{x}$  and  $\boldsymbol{\mu}$ , reducing the total running time  $O(dkn_v + dn_v \log n_v)$ .

### 3.3 Theoretical guarantees

Now that we have defined our ExKMC algorithm, we provide some guarantees on its performance (we defer all proofs for this section to Appendix A). We first prove two theorems about the surrogate cost, showing that it satisfies the two desirable properties of being non-increasing and also eventually converging to the reference clustering. Next, we verify that ExKMC has a worst-case approximation ratio of  $O(k^2)$  compared to the optimal  $k$ -means clustering when using IMM to build the base tree. Finally, we provide a separation between IMM and ExKMC for a difficult dataset.

**Theorem 1.** *The surrogate cost,  $\widetilde{\text{cost}}$ , is non-increasing throughout the execution of ExKMC.*

To prove the theorem notice that when the algorithm performs a split at node  $v$ , the cost of all points not in  $v$  will remain intact. Additionally, the algorithm can assign the two children of  $v$  the same label as in  $v$ . This choice will not change the cost of the points in  $v$ . The full proof appears in the appendix. Eventually ExKMC converges to a low-cost tree, as the next theorem proves. Specifically, after  $n$  steps in the worst-case, we will always get a refinement, see Corollary 2 in Appendix A.

**Theorem 2.** *Let  $C$  be a reference clustering. If while running ExKMC the threshold tree  $T$  refines  $C$ , then the clustering induced by  $(T, \ell)$  equals  $C$ , where  $\ell$  is the surrogate cost labeling.*

We also provide a worst-case guarantee compared to the optimal  $k$ -means solution. We show that the ExKMC algorithm using the IMM base tree provably achieves an  $O(k^2)$  approximation to the  $k$ -means cost. The proof uses Theorem 1 combined with the previous analysis of the IMM algorithm [22], see Appendix A.

**Theorem 3.** *The cost of the Algorithm 1 with IMM base tree is an  $O(k^2)$  approximation.*

### 3.4 Example where ExKMC provably improves upon IMM

Prior work designed a dataset and proved that a tree with  $k$  leaves incurs an  $\Omega(\log k)$  approximation on this dataset [22], which we call *Synthetic II*. It contains  $k$  clusters that globally are very far from each other, but locally, clusters look the same.

**Synthetic II dataset.** This dataset,  $\mathcal{D}$ , consists of  $k$  clusters where any two clusters are very far from each other while inside any cluster the points differ by at most two features. The dataset is created by first taking  $k$  random binary *codewords*  $\mathbf{v}^1, \dots, \mathbf{v}^k \in \{-1, 1\}^d$ . The distance between any two codewords is at least  $\alpha d$ , where  $\alpha$  is some constant. Each cluster,  $C^{v^i}$ , is created by first taking a codeword and then changing one feature at a time to 0. The size of each cluster is  $|C^{v^i}| = d$ . There are in total  $dk$  points in the dataset. The optimal centers for this dataset are the codewords. We take  $d > ck^2$  for some universal constant  $c > 1$ , and  $k$  is sufficiently large constant.

Our approach of explaining this dataset contains a few steps (i) using  $k$ -means++ to generate the reference centers, (ii) building the base tree with IMM, and (iii) expanding the tree with ExKMC. We prove that this leads to an optimal tree-based clustering with  $O(k \log k)$  leaves. In fact, the cost ratio decreases linearly as the number of leaves increases, as the next theorem proves. We experimentally confirm this linear decrease in the next section.

**Theorem 4.** *There is a constant  $c' > 0$  such that with probability at least  $1 - O\left(\frac{k \log k}{d}\right)$  over the randomness in  $k$ -means++, the output of **ExKMC** with centers from  $k$ -means++, base tree of **IMM** and  $k'$  desired centers, the approximation ratio is at most*

$$O\left(\max\left\{\log k - c' \cdot \frac{k'}{k}, 1\right\}\right)$$

We show that initially the **IMM** algorithm has approximation ratio of  $O(\log k)$ . Interestingly, this is also the lower bound for trees with exactly  $k$  leaves, and hence, it constructs the best possible tree up to a constant factor. The theorem also states that after  $O(k \log k)$  steps we reach the optimal clustering. We state this observation as the following corollary.

**Corollary 1.** *With probability at least  $1 - O\left(\frac{k \log k}{d}\right)$  over the randomness in  $k$ -means++, the output of **ExKMC**, with centers from  $k$ -means++ and base tree of **IMM**, after  $O(k \log k)$  steps of the algorithm, it reaches the optimal clustering of  $\mathcal{D}$ .*

The full proof of Theorem 4 is in the appendix. The main steps of the proof are the following. We first prove that with high probability the output of the  $k$ -means++ algorithm is actually the optimal set of  $k$  centers, that is, the codewords in the dataset construction. Intuitively, this holds because globally the optimal centers and clusters are very far from each other. The second step is to define points that are *mistakes*, which are points in a leaf  $u$  with the property that their optimal center is different than the label of  $u$ . To guarantee that the algorithm always makes progress, we prove that there is always a split that reduces the number of mistakes. Lastly, we prove that each such split reduces the cost by  $\Omega(d)$ . This completes the proof.

## 4 Empirical Evaluation

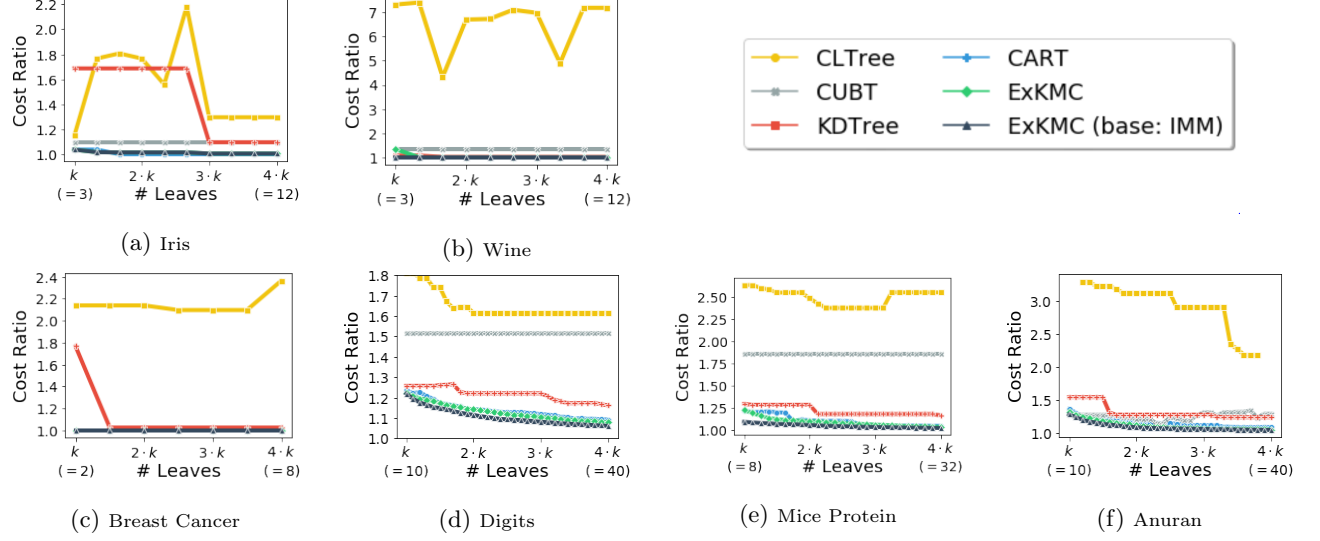
**Algorithms.** We compare the following clustering methods (see Appendix B for details):

- **Reference Clustering.** `sklearn KMeans`, 10 random initializations, 300 iterations.
- **CART.** Standard decision tree from `sklearn`, minimizing `gini` impurity. Points in the dataset are assigned labels using the reference clustering.
- **KDTree.** Split highest variance feature at median threshold. Size determined by `leaf_size` parameter. Label leaves to minimize  $\widetilde{\text{cost}}$  w.r.t. centers of the reference clustering.
- **CLTree.** Explainable clustering method. Public implementation [19, 43].
- **CUBT.** Explainable clustering method. Public implementation [31, 28].
- **ExKMC.** Our method (Algorithm 1) starting with an empty tree; minimizes  $\widetilde{\text{cost}}$  at each split w.r.t. centers of the reference clustering.
- **ExKMC (base: IMM).** Our method (Algorithm 1) starting with an **IMM** tree with  $k$  leaves; then, minimizes  $\widetilde{\text{cost}}$  at each split w.r.t. centers of the reference clustering.

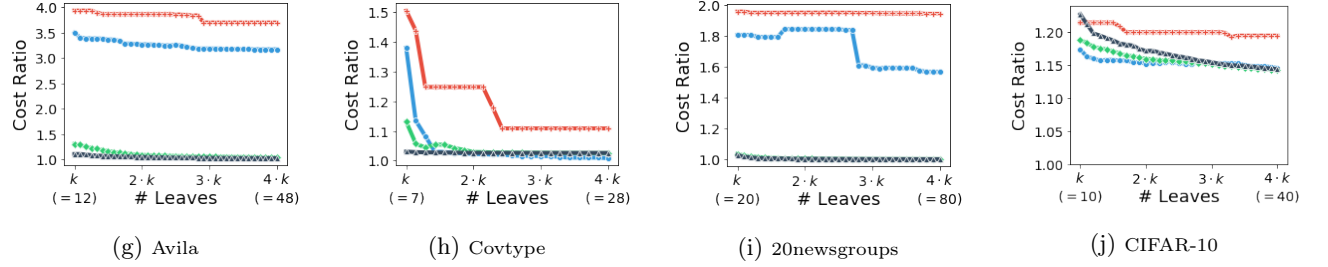
**Set-up.** We use 10 real and 2 synthetic datasets; details in Appendix B.2. The number of clusters  $k$  and the number of leaves  $k'$  are inputs. We start with  $k$  equal to number of labels for classification datasets. We plot the cost ratio compared to the reference clustering (best = 1.0). For the baselines, we do hyperparameter tuning and choose the lowest cost clustering at each  $k'$ . **CUBT** and **CLTree** could only be feasibly executed on six small real datasets (we restrict computation time to one hour).



## Small Datasets



## Larger Datasets



## Synthetic Datasets

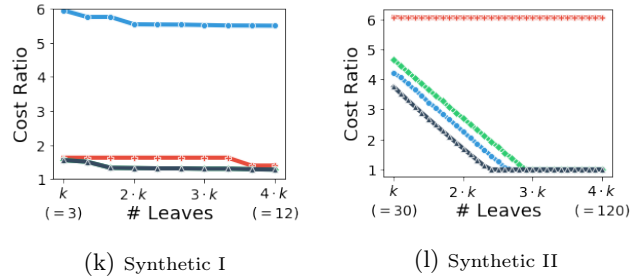


Figure 2: Each graph plots the ratio ( $y$ -axis) of the tree-based clustering cost to the near-optimal  $k$ -means clustering as a function of the number of leaves ( $x$ -axis). Lower is better with best = 1.0. Our algorithm (black line) consistently performs well. See Figure 9 in Appendix B.4 for error bars.

## 4.1 Experimental Results

**Real datasets.** CLTree performs the worst in most cases, and the cost is often much larger than the other methods. Turning to CUBT, we see that on most datasets it is competitive (but often not the best). However, on Digits and Mice Protein, CUBT fails to converge to a good clustering. We see that CART performs well on many of the datasets, as expected. On Avila and 20newsgroups, CART has a fairly high cost. For the small datasets, KDTree performs competitively, but on large datasets, the cost remains high. On all of the datasets, our method ExKMC performs well. It often has the lowest cost throughout, except for CIFAR-10, where it requires around  $3.5 \cdot k$  leaves to be competitive. When the number of leaves is exactly  $k$ , we can also see the performance of the IMM algorithm, where we see that its cost is quite low, in contrast to the previous theoretical analysis [22]. As usual, the outlier is CIFAR-10, where the cost is very high at  $k$  leaves, and then quickly decreases as ExKMC expands the tree. We also evaluate ExKMC by starting with an empty tree (without using IMM at all). In general, the cost is worse or the same as ExKMC with IMM. We observe that the CLTree cost varies as a function of the number of leaves. The reason is that we perform a hyperparameter search for each instance separately, and perhaps surprisingly, the cost can sometimes increase. While we could have used the best tree with fewer leaves, this may be less representative of real usage. Indeed, the user would specify a desired number of leaves, and they may not realize that fewer leaves would lead to lower  $k$ -means cost.

**Synthetic datasets.** We highlight two other aspects of explainable clustering. The Synthetic I dataset in Figure 2(k) is bad for CART. We adapt a dataset from prior work that forces CART to incur a large cost due to well-placed outliers [22]. CART has cost ratio above five, while other methods converge to a near-optimal cost. The Synthetic II dataset in Figure 2(l) is sampled from a hard distribution that demonstrates an  $\Omega(\log k)$  lower bound for any explainable clustering with  $k$  leaves [22]. The  $k$  centers are random  $\pm 1$  vectors in  $\mathbb{R}^d$ , and the clusters contain the  $d$  vectors where one of the center’s coordinates is set to zero. As expected, the IMM algorithm starts with a high  $k$ -means cost. When ExKMC expands the tree to have more leaves, the cost quickly goes down (proof in Appendix A.1).

**Distance to the reference clustering.** In Appendix B.4, we report accuracy when points are labeled by cluster. Overall, we see the same trends as with cost. However, on some datasets, CART more closely matches the reference clustering than ExKMC, but the CART clustering has a higher cost. This highlights the importance of optimizing the  $k$ -means cost, not the classification accuracy.

**Qualitative analysis.** Figure 3 depicts two example trees with four and eight leaves, respectively, on a subset of four clusters from the 20newsgroups dataset. The IMM base tree uses three features (words) to define four clusters. Then, ExKMC expands one of the leaves into a larger subtree, using seven total words to construct more nuanced clusters that better correlate with the newsgroup topics.

**Running time.** Figure 4 shows the runtime of three methods on seven real datasets (commodity laptop, single process, i7 CPU @ 2.80GHz, 16GB RAM). Both IMM and ExKMC first run KMeans (from sklearn, 10 initializations, 300 iterations), and we report cumulative times. The explainable algorithms construct trees in under 15 minutes. On six datasets, they incur  $0.25\times$  to  $1.5\times$  overhead compared to standard KMeans. The 20newsgroups dataset has the largest overhead because sklearn optimizes for sparse vectors while IMM and ExKMC currently do not. In Appendix B.4, we also measure an improvement to ExKMC with feature-wise parallelization of the expansion step.

**Surrogate vs. actual cost.** In Figure 5, we compare the surrogate cost (using the  $k$  reference centers) with the actual  $k$ -means cost (using the means of the clusters as centers). In general, the two costs differ by at most 5-10%. On three datasets, they converge to match each other as the number of leaves grows. Covtype is an exception, where the two costs remain apart.

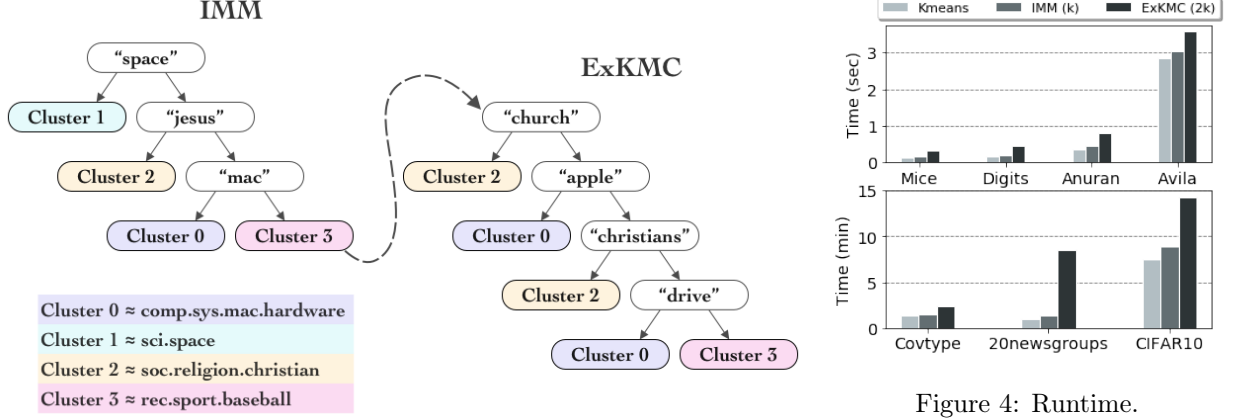


Figure 4: Runtime.

Figure 3: Explainability-accuracy trade-off using more leaves on a subset of four clusters from 20newsgroups. As ExKMC expands the IMM tree, it refines the clusters and reduces the 4-means cost.

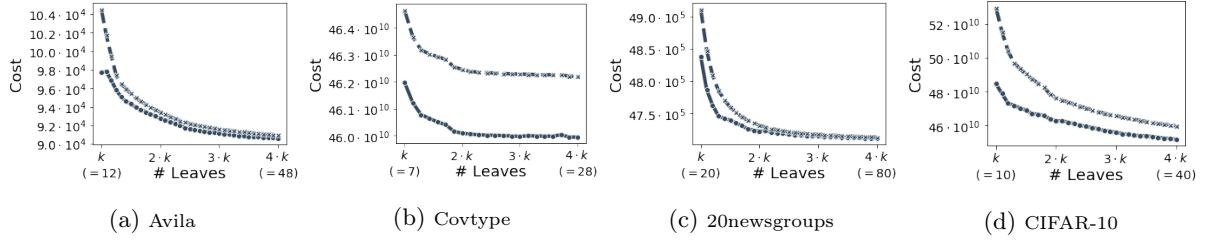


Figure 5: Comparison of  $\widetilde{\text{cost}}$  (dashed line) vs. cost (full line) of ExKMC with IMM base tree.

## 4.2 Discussion

**Trade-off.** The main objective of our new algorithm is to provide a flexible trade-off between explainability and accuracy. Compared to the previous IMM algorithm, we see that using ExKMC to expand the threshold tree consistently leads to a lower cost clustering. We also see that using our surrogate cost improves the running time without sacrificing effectiveness.

**Convergence.** On some datasets, IMM produces a fairly low  $k$ -means cost with  $k$  leaves. Expanding the tree with ExKMC to between  $2k$  and  $4k$  leaves often results in nearly the same cost as the reference clustering. CIFAR-10 is an outlier, where none of the methods converge when using pixels as features (see also Appendix B.4). In practice, a tree with  $4k$  leaves only slightly increases the explanation complexity compared to a tree with  $k$  leaves (and  $k$  leaves are necessary). ExKMC successfully achieves a good tree-based clustering with better interpretability than standard  $k$ -means methods.

**Low cost.** The most striking part of the experiments is that tree-based clusterings can match the cost of standard clusterings. This is possible with a small tree, even on large, high-dimensional datasets. Prior to our work, this was not known to be the case. Therefore, ExKMC demonstrates that explainability can be obtained in conjunction with low cost on many datasets.

## 5 Conclusion

We exhibited a new algorithm, **ExKMC**, for generating an explainable  $k$ -means clustering using a threshold tree with a specified number of leaves. Theoretically, our algorithm has the property that as the number of leaves increases, the cost is non-increasing. This enables a deliberate trade-off between explainability and clustering cost. Extensive experiments showed that our algorithm often achieves lower  $k$ -means cost compared to several baselines. Moreover, we saw that **ExKMC** usually matches the reference clustering cost with only  $4k$  leaves. Overall, **ExKMC** efficiently produces explainable clusters, and it could potentially replace standard  $k$ -means implementations in data science pipelines. For future work, it would be interesting to prove convergence guarantees for **ExKMC** either based on separation properties of real data or a Gaussian mixture model assumption. Another direction is to reduce the running time through better parallelism and sparse feature vector optimizations. Finally, our algorithm is feature-based, and therefore, if the features or data have biases, then these biases may also propagate and lead to biased clusters. It would be interesting future work to develop an explainable clustering algorithm that enforces fairness either in the feature selection process or in the composition of the clusters (for examples of such methods, see [6, 10, 34, 39, 47, 59]).

## Acknowledgements

We thank Sanjoy Dasgupta for helpful discussions. We also thank Alyshia Olsen for help designing the figures.

## References

- [1] A. Aggarwal, A. Deshpande, and R. Kannan. Adaptive sampling for  $k$ -means clustering. In *Approximation, Randomization, and Combinatorial Optimization. Algorithms and Techniques*, pages 15–28. Springer, 2009.
- [2] D. Aloise, A. Deshpande, P. Hansen, and P. Popat. NP-hardness of Euclidean sum-of-squares clustering. *Machine learning*, 75(2):245–248, 2009.
- [3] A. B. Arrieta, N. Díaz-Rodríguez, J. Del Ser, A. Bennetot, S. Tabik, A. Barbado, S. García, S. Gil-López, D. Molina, R. Benjamins, et al. Explainable artificial intelligence (xai): Concepts, taxonomies, opportunities and challenges toward responsible ai. *Information Fusion*, 58:82–115, 2020.
- [4] D. Arthur and S. Vassilvitskii.  $k$ -means++: The advantages of careful seeding. In *Proceedings of the Eighteenth Annual ACM-SIAM Symposium on Discrete algorithms*, pages 1027–1035. Society for Industrial and Applied Mathematics, 2007.
- [5] P. Awasthi, M. Charikar, R. Krishnaswamy, and A. K. Sinop. The hardness of approximation of Euclidean  $k$ -means. In *31st International Symposium on Computational Geometry, SoCG 2015*, pages 754–767. Schloss Dagstuhl-Leibniz-Zentrum für Informatik GmbH, Dagstuhl Publishing, 2015.
- [6] A. Backurs, P. Indyk, K. Onak, B. Schieber, A. Vakilian, and T. Wagner. Scalable fair clustering. In *International Conference on Machine Learning*, pages 405–413, 2019.
- [7] J. Basak and R. Krishnapuram. Interpretable hierarchical clustering by constructing an unsupervised decision tree. *IEEE transactions on knowledge and data engineering*, 17(1):121–132, 2005.
- [8] L. Becchetti, M. Bury, V. Cohen-Addad, F. Grandoni, and C. Schwiegelshohn. Oblivious dimension reduction for  $k$ -means: beyond subspaces and the Johnson-Lindenstrauss lemma. In *STOC*, 2019.
- [9] J. L. Bentley. Multidimensional binary search trees used for associative searching. *Communications of the ACM*, 18(9):509–517, 1975.

- [10] S. Bera, D. Chakrabarty, N. Flores, and M. Negahbani. Fair algorithms for clustering. In *Advances in Neural Information Processing Systems*, pages 4955–4966, 2019.
- [11] D. Bertsimas, A. Orfanoudaki, and H. Wiberg. Interpretable clustering via optimal trees. *arXiv preprint arXiv:1812.00539*, 2018.
- [12] J. A. Blackard and D. J. Dean. Comparative accuracies of artificial neural networks and discriminant analysis in predicting forest cover types from cartographic variables. *Computers and electronics in agriculture*, 24(3):131–151, 1999.
- [13] C. Boutsidis, P. Drineas, and M. W. Mahoney. Unsupervised feature selection for the k-means clustering problem. In *NIPS*, pages 153–161, 2009.
- [14] C. Boutsidis, A. Zouzias, M. W. Mahoney, and P. Drineas. Randomized dimensionality reduction for  $k$ -means clustering. *IEEE Transactions on Information Theory*, 61(2):1045–1062, 2014.
- [15] L. Breiman, J. Friedman, C. J. Stone, and R. A. Olshen. *Classification and regression trees*. CRC press, 1984.
- [16] J.-W. Chang and D.-S. Jin. A new cell-based clustering method for large, high-dimensional data in data mining applications. In *Proceedings of the 2002 ACM symposium on Applied computing*, pages 503–507, 2002.
- [17] J. Chen. *Interpretable Clustering Methods*. PhD thesis, Northeastern University, 2018.
- [18] J. Chen, Y. Chang, B. Hobbs, P. Castaldi, M. Cho, E. Silverman, and J. Dy. Interpretable clustering via discriminative rectangle mixture model. In *2016 IEEE 16th International Conference on Data Mining (ICDM)*, pages 823–828. IEEE, 2016.
- [19] D. Christodoulou. Python-package for clustering via decision tree construction. <https://github.com/dimitrs/CLTree>.
- [20] M. B. Cohen, S. Elder, C. Musco, C. Musco, and M. Persu. Dimensionality reduction for k-means clustering and low rank approximation. In *STOC*, 2015.
- [21] S. Dasgupta. The hardness of k-means clustering. In *Technical Report*. University of California, San Diego (Technical Report), 2008.
- [22] S. Dasgupta, N. Frost, M. Moshkovitz, and C. Rashtchian. Explainable  $k$ -means and  $k$ -medians clustering. *arXiv preprint arXiv:2002.12538*, 2020.
- [23] L. De Raedt and H. Blockeel. Using logical decision trees for clustering. In *International Conference on Inductive Logic Programming*, pages 133–140. Springer, 1997.
- [24] C. De Stefano, M. Maniaci, F. Fontanella, and A. S. di Freca. Reliable writer identification in medieval manuscripts through page layout features: The “avila” bible case. *Engineering Applications of Artificial Intelligence*, 72:99–110, 2018.
- [25] D. Deutch and N. Frost. Constraints-based explanations of classifications. In *2019 IEEE 35th International Conference on Data Engineering (ICDE)*, pages 530–541. IEEE, 2019.
- [26] D. Dua and C. Graff. UCI machine learning repository, 2017.
- [27] R. A. Fisher. The use of multiple measurements in taxonomic problems. *Annals of eugenics*, 7(2):179–188, 1936.
- [28] R. Fraiman, B. Ghattas, and M. Svarc. Interpretable clustering using unsupervised binary trees. *Advances in Data Analysis and Classification*, 7(2):125–145, 2013.
- [29] P. Geurts and G. Louppe. Learning to rank with extremely randomized trees. In *JMLR: workshop and conference proceedings*, volume 14, pages 49–61, 2011.
- [30] P. Geurts, N. Touleimat, M. Dutreix, and F. d’Alché Buc. Inferring biological networks with output kernel trees. *BMC Bioinformatics*, 8(2):S4, 2007.
- [31] B. Ghattas. R-package for interpretable clustering using unsupervised binary trees. <http://www.i2m.univ-amu.fr/perso/badih.ghattas/CUBT.html>.

- [32] B. Ghattas, P. Michel, and L. Boyer. Clustering nominal data using unsupervised binary decision trees: Comparisons with the state of the art methods. *Pattern Recognition*, 67:177–185, 2017.
- [33] C. Higuera, K. J. Gardiner, and K. J. Cios. Self-organizing feature maps identify proteins critical to learning in a mouse model of down syndrome. *PloS one*, 10(6), 2015.
- [34] L. Huang, S. Jiang, and N. Vishnoi. Coresets for clustering with fairness constraints. In *Advances in Neural Information Processing Systems*, pages 7587–7598, 2019.
- [35] Y. Jernite, A. Choromanska, and D. Sontag. Simultaneous learning of trees and representations for extreme classification and density estimation. In *Proceedings of the 34th International Conference on Machine Learning-Volume 70*, pages 1665–1674. JMLR. org, 2017.
- [36] T. Joachims. A probabilistic analysis of the rocchio algorithm with tfidf for text categorization. Technical report, Carnegie-mellon univ pittsburgh pa dept of computer science, 1996.
- [37] T. Kanungo, D. M. Mount, N. S. Netanyahu, C. D. Piatko, R. Silverman, and A. Y. Wu. A local search approximation algorithm for k-means clustering. In *Proceedings of the Eighteenth Annual Symposium on Computational Geometry*, pages 10–18, 2002.
- [38] J. Kauffmann, M. Esders, G. Montavon, W. Samek, and K.-R. Müller. From clustering to cluster explanations via neural networks. *arXiv preprint arXiv:1906.07633*, 2019.
- [39] M. Kleindessner, P. Awasthi, and J. Morgenstern. Fair k-center clustering for data summarization. In *International Conference on Machine Learning*, pages 3448–3457, 2019.
- [40] A. Krizhevsky, G. Hinton, et al. Learning multiple layers of features from tiny images. *Technical report*, 2009.
- [41] Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, 1998.
- [42] Z. C. Lipton. The mythos of model interpretability. *Queue*, 16(3):31–57, 2018.
- [43] B. Liu, Y. Xia, and P. S. Yu. Clustering via decision tree construction. In *Foundations and Advances in Data Mining*, pages 97–124. Springer, 2005.
- [44] W.-Y. Loh. Classification and regression trees. *Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery*, 1(1):14–23, 2011.
- [45] G. Louppe, L. Wehenkel, A. Sutera, and P. Geurts. Understanding variable importances in forests of randomized trees. In *Advances in neural information processing systems*, pages 431–439, 2013.
- [46] S. M. Lundberg and S.-I. Lee. A unified approach to interpreting model predictions. In *Advances in Neural Information Processing Systems*, pages 4765–4774, 2017.
- [47] S. Mahabadi and A. Vakilian. (individual) fairness for  $k$ -clustering. *arXiv preprint arXiv:2002.06742*, 2020.
- [48] K. Makarychev, Y. Makarychev, and I. Razenshteyn. Performance of Johnson-Lindenstrauss transform for k-means and k-medians clustering. In *STOC*, 2019.
- [49] A. M. Mendoza-Henao, Á. M. Cortes-Gomez, M. A. Gonzalez, O. D. Hernandez-Córdoba, A. R. Acosta-Galvis, F. Castro-Herrera, J. M. Daza, J. M. Hoyos, M. P. Ramirez-Pinilla, N. Urbina-Cardona, et al. A morphological database for colombian anuran species from conservation-priority ecosystems. *Ecology*, 100(5):e02685, 2019.
- [50] C. Molnar. *Interpretable Machine Learning*. Lulu. com, 2019. <https://christophm.github.io/interpretable-ml-book/>.
- [51] W. J. Murdoch, C. Singh, K. Kumbier, R. Abbasi-Asl, and B. Yu. Interpretable machine learning: definitions, methods, and applications. *arXiv preprint arXiv:1901.04592*, 2019.
- [52] R. Ostrovsky, Y. Rabani, L. J. Schulman, and C. Swamy. The effectiveness of Lloyd-type methods for the k-means problem. *Journal of the ACM*, 59(6):1–22, 2013.

- [53] D. Pelleg and A. W. Moore. Mixtures of rectangles: Interpretable soft clustering. In *Proceedings of the Eighteenth International Conference on Machine Learning (ICML)*, pages 401–408, 2001.
- [54] K. Pliakos, P. Geurts, and C. Vens. Global multi-output decision trees for interaction prediction. *Machine Learning*, 107(8-10):1257–1281, 2018.
- [55] P. Ram and A. G. Gray. Density estimation trees. In *Proceedings of the 17th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 627–635, 2011.
- [56] M. T. Ribeiro, S. Singh, and C. Guestrin. Why should I trust you?: Explaining the predictions of any classifier. In *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 1135–1144. ACM, 2016.
- [57] C. Rudin. Stop explaining black box machine learning models for high stakes decisions and use interpretable models instead. *Nature Machine Intelligence*, 1(5):206–215, 2019.
- [58] S. Saisubramanian, S. Galhotra, and S. Zilberstein. Balancing the tradeoff between clustering value and interpretability. In *Proceedings of the AAAI/ACM Conference on AI, Ethics, and Society*, pages 351–357, 2020.
- [59] M. Schmidt, C. Schwiegelshohn, and C. Sohler. Fair coresets and streaming algorithms for fair k-means. In *International Workshop on Approximation and Online Algorithms*, pages 232–251. Springer, 2019.
- [60] M. Schrynemackers, L. Wehenkel, M. M. Babu, and P. Geurts. Classifying pairs with trees for supervised biological network inference. *Molecular Biosystems*, 11(8):2116–2125, 2015.
- [61] Y. Yasami and S. P. Mozaffari. A novel unsupervised classification approach for network anomaly detection by k-means clustering and id3 decision tree learning methods. *The Journal of Supercomputing*, 53(1):231–245, 2010.

## A Omitted proofs and extra theoretical results

*Proof of Theorem 1.* Denote by  $T^{k'}$  the tree at iteration  $k'$  of the algorithm. We want to show that

$$\widetilde{\text{cost}}^{\mu^1, \dots, \mu^k}(T^{k'+1}) \leq \widetilde{\text{cost}}^{\mu^1, \dots, \mu^k}(T^{k'}).$$

We prove a stronger claim: for any possible split (not just for the one by ExKMC) the surrogate cost will not increase. Fix a cluster  $\widehat{C}^r$  in  $T^{k'}$  and suppose that we split it into two clusters  $\widehat{C}^{r,1}, \widehat{C}^{r,2}$  and that this results in the tree  $T$ . For the split the algorithm chooses, we have  $T = T^{k'+1}$ .

We separate the surrogate cost into two terms: points that are in  $\widehat{C}^r$  and those that are not

$$\begin{aligned} \widetilde{\text{cost}}^{\mu^1, \dots, \mu^k}(T) &= \sum_{\substack{\widehat{C}^j \in T^{k'} \\ j \neq r}} \sum_{\mathbf{x} \in \widehat{C}^j} \|\mathbf{x} - c^{\mu^1, \dots, \mu^k}(\widehat{C}^j)\|_2^2 \\ &\quad + \sum_{\mathbf{x} \in \widehat{C}^{r,1}} \|\mathbf{x} - c^{\mu^1, \dots, \mu^k}(\widehat{C}^{r,1})\|_2^2 + \sum_{\mathbf{x} \in \widehat{C}^{r,2}} \|\mathbf{x} - c^{\mu^1, \dots, \mu^k}(\widehat{C}^{r,2})\|_2^2, \end{aligned}$$

Importantly, the first term appears also in  $\widetilde{\text{cost}}^{\mu^1, \dots, \mu^k}(T^{k'})$ . Thus,

$$\widetilde{\text{cost}}^{\mu^1, \dots, \mu^k}(T) - \widetilde{\text{cost}}^{\mu^1, \dots, \mu^k}(T^{k'})$$

is equal to

$$\sum_{\mathbf{x} \in \widehat{C}^{r,1}} \|\mathbf{x} - c^{\mu^1, \dots, \mu^k}(\widehat{C}^{r,1})\|_2^2 + \sum_{\mathbf{x} \in \widehat{C}^{r,2}} \|\mathbf{x} - c^{\mu^1, \dots, \mu^k}(\widehat{C}^{r,2})\|_2^2 - \sum_{\mathbf{x} \in \widehat{C}^r} \|\mathbf{x} - c^{\mu^1, \dots, \mu^k}(\widehat{C}^r)\|_2^2.$$

This is at most 0 because we can give  $\widehat{C}^{r,1}$  and  $\widehat{C}^{r,2}$  the same center in  $\mu^1, \dots, \mu^k$  that was given to  $\widehat{C}^r$ .  $\square$

*Proof of Theorem 2.* Once  $T$  defines a clustering  $(\widehat{C}^1, \dots, \widehat{C}^{k'})$  that is a refinement, we know that for any cluster  $i \in [k']$  each points will be assigned the same center, i.e., for all  $\mathbf{x} \in \widehat{C}^i$ , the value  $c^{\mu^1, \dots, \mu^k}(\mathbf{x})$  is the same. Thus,

$$\widetilde{\text{cost}}^{\mu^1, \dots, \mu^k}(T) = \sum_{j=1}^{k'} \sum_{\mathbf{x} \in \widehat{C}^j} \|\mathbf{x} - c^{\mu^1, \dots, \mu^k}(\mathbf{x})\|_2^2 = \sum_{\mathbf{x}} \|\mathbf{x} - c^{\mu^1, \dots, \mu^k}(\mathbf{x})\|_2^2,$$

which is exactly equal to the  $k$ -means reference clustering.  $\square$

**Corollary 2.** *If ExKMC builds a tree with  $n$  leaves, then it exactly matches the reference clustering, and the surrogate cost is equal to the reference cost.*

*Proof.* If there is a step such that  $T$  is a refinement of  $C^{\mu^1, \dots, \mu^k}$ , then, using Theorem 2, the corollary holds. Using Theorem 1, the tree will continue to expand until either it is a refinement of  $C^{\mu^1, \dots, \mu^k}$  or there are no more points and the tree contains  $n$  leaves. A clustering defined by  $n$  leaves is one where each point is in a cluster of its own, which is a refinement of  $C^{\mu^1, \dots, \mu^k}$ . More specifically, for the tree  $T$  that contains  $n$  leaves, we see that the surrogate cost is equal to

$$\widetilde{\text{cost}}^{\mu^1, \dots, \mu^k}(T) = \sum_{\mathbf{x}} \|\mathbf{x} - c^{\mu^1, \dots, \mu^k}(\mathbf{x})\|_2^2.$$

This is exactly equal to the  $k$ -means reference clustering.  $\square$



*Proof of Theorem 3.* Denote by  $T$  the output threshold tree of the IMM algorithm and by  $T^{k'}$  the output threshold tree of Algorithm 1. We want to show that

$$\text{cost}(T^{k'}) \leq O(k^2) \cdot \text{cost}(\text{opt}_k).$$

The proof of Theorem 3 in [22] shows that  $\widetilde{\text{cost}}(T) \leq O(k^2) \cdot \text{cost}(\text{opt}_k)$ . Together with Theorem 1, we know that the surrogate cost is non-increasing, thus

$$\widetilde{\text{cost}}(T^{k'}) \leq O(k^2) \cdot \text{cost}(\text{opt}_k).$$

As the surrogate cost upper bounds the  $k$ -means cost, we know that  $\text{cost}(T^{k'}) \leq \widetilde{\text{cost}}(T^{k'})$ .  $\square$

## A.1 Hard-to-explain dataset

In [22] a dataset was shown that cannot have an  $O(1)$ -approximation with any threshold tree that has  $k$  leaves. In this section we will show that there is a tree with only  $O(k \log k)$  leaves that achieves the optimal clustering, even on this difficult dataset. Moreover, the ExKMC algorithm outputs such a threshold tree.

It will be useful later on to analyze the pairwise distances between the codewords. Using Hoeffding's inequality we prove that the distances are very close to  $d/2$ .

**Proposition 1** (Hoeffding's inequality). *Let  $X_1, \dots, X_n$  be independent random variables, where for each  $i$ ,  $X_i \in [0, 1]$ . Define the random variable  $X = \sum_{i=1}^n X_i$ . Then, for any  $t \geq 0$ ,*

$$\Pr(|X - \mathbb{E}[X]| \geq t) \leq 2e^{-\frac{2t^2}{n}}.$$

**Claim 1.** *Let  $\delta \in (0, 1)$ . With probability at least  $1 - \delta$  over the choice of  $k$  random vectors in  $\{-1, 1\}^d$ , all the pairwise distances between the vectors are  $\frac{d}{2} \pm 2\sqrt{d} \ln \frac{k}{2\delta}$ .*

*Proof.* The average distance between any two random vectors is  $d/2$ . From Hoeffding's inequality and union bound, the probability that there is a pair with distance that deviates by more than  $t = 2\sqrt{d} \ln \frac{k}{2\delta}$  is bounded by

$$k^2 \cdot 2e^{-\frac{2t^2}{d}} \leq \delta.$$

$\square$

For the analysis of ExKMC on dataset  $\mathcal{D}$  we will use  $k$ -means++ to find the reference centers. For completeness, we write its pseudo-code and introduce the notion of distance of point  $\mathbf{p}$  from a set of points  $S$  as

$$\text{dist}(\mathbf{p}, S) = \min_{\mathbf{x} \in S} \|\mathbf{p} - \mathbf{x}\|_2^2.$$

In Claim 2 we will show that one iteration of Lloyd's algorithm is enough for convergence. Summarizing, we consider the following form of our algorithm, that uses  $k$ -means++ to find the reference centers and IMM algorithm for the base tree:

1. Use the  $k$ -means++ sampling procedure to choose  $k$  centers from the dataset. Letting  $S^i$  denote the centers at step  $i$ , we add a new center by sampling a point  $\mathbf{p}$  with probability

$$\frac{\text{dist}(\mathbf{p}, S^i)}{\sum_{\mathbf{x} \in \mathcal{X}} \text{dist}(\mathbf{x}, S^i)}.$$

---

**ALGORITHM 3:**  $k$ -means++

---

**Input** :  $\mathcal{X}$  – Set of vectors in  $\mathbb{R}^d$   
           $k$  – Number of centers  
**Output** :  $\mathcal{M}$  – Set of  $k$  centers

```
1  $S \leftarrow \text{set}()$ 
2 while  $|S| < k$  do
3   sample  $\mathbf{p} \in \mathcal{X}$  with probability  $\frac{\text{dist}(\mathbf{p}, S)}{\sum_{\mathbf{x} \in \mathcal{X}} \text{dist}(\mathbf{x}, S)}$ 
4    $S.\text{add}(\mathbf{p})$ 
5 run Lloyd's algorithm with centers  $S$ 
6 return  $S$ 
```

---

2. Use one iteration of Lloyd's algorithm, where we first recluster based on the centers in  $S^k$ , then take the means of the new clusters  $\mathcal{M}$  as the  $k$  reference centers.
3. Run the IMM algorithm with reference centers  $\mathcal{M}$  to construct a base tree with  $k$  leaves.
4. Expand the tree to  $k' > k$  leaves using ExKMC with reference centers  $\mathcal{M}$  and the IMM tree.

We first prove that after using the  $k$ -means++ initialization and one iteration of Lloyd's algorithm, the resulting centers are the optimal centers, i.e., the codewords.

**Claim 2.** *When running  $k$ -means++ on  $\mathcal{D}$ , with probability at least  $1 - O(\frac{k \log k}{d})$  the resulting centers are the optimal ones.*

*Proof.* We will show that with probability at least  $1 - O(\frac{k \log k}{d})$ , after the initialization step, the  $k$ -means++ algorithm took one point from each cluster. This will imply that the optimal clusters have been found, and the optimal centers will be returned after the next step in the  $k$ -means++ algorithm, one iteration of Lloyd's algorithm.

We prove by induction on  $|S| = i$  that with probability at most  $\sum_{j=1}^i \frac{2j}{\alpha d(k-j)}$ , the variable  $S$  does not contain  $i$  points from  $i$  different codewords, where  $\alpha d \geq d/4$  is a lower bound to the codewords pairwise distances. The base of the induction is trivial. Suppose that  $|S| = i$ , where  $1 < i < k$ , and  $S$  contains points from  $C^{v^1}, \dots, C^{v^i}$ . The distance from  $S$  of each point  $\mathbf{p}$  that its cluster was taken already,  $\mathbf{p} \in \cup_{j=1}^i C^{v^j}$ , is at most  $\text{dist}(\mathbf{p}, S) \leq 2$ . There are at most  $di$  such points. The distance of points not taken is at least  $\alpha d$ . There are at least  $d(k-i)$  such points. Thus, using union bound, the probability to take a point from  $\cup_{j=1}^i C^{v^j}$  is at most

$$\frac{2di}{\alpha d^2(k-i)} = \frac{2}{\alpha d} \cdot \frac{i}{k-i}.$$

We use the induction hypothesis and show that in total the probability to return  $i+1$  points not all from different clusters is at most

$$\frac{2}{\alpha d} \sum_{j=1}^i \frac{j}{k-j}.$$

Specifically, at the end, the error probability is bounded by

$$\frac{2}{\alpha d} \sum_{i=1}^{k-1} \frac{i}{k-i} \leq \frac{2}{\alpha d} \sum_{i=1}^{k-1} \frac{k}{k-i} = \frac{2k}{\alpha d} \sum_{i=1}^{k-1} \frac{1}{i} = O\left(\frac{k \log k}{d}\right)$$

□

In the next claim we show that if we run IMM with the codewords as the reference centers, then the cost is bounded by  $O(dk \log k)$ . The IMM algorithm can get the codewords as centers by running the  $k$ -means++ algorithm and use the previous claim.

**Claim 3.** *The output threshold tree of the IMM algorithm on  $\mathcal{D}$  with the  $k$  codewords as the set of reference centers, has  $k$ -means cost  $O(dk \log k)$ .*

*Proof.* The work [22] defines the mistakes of an inner node  $u_{\theta,i}$  as follows: If  $u_{\theta,i}$  defines a split using feature  $i$  and threshold  $\theta$ , a mistake is a point  $\mathbf{p}$  and its center  $v^j$  that both reached  $u_{\theta,i}$  but now going to different direction, i.e.,

$$(p_i \leq \theta \text{ and } v_i^j > \theta) \quad \text{or} \quad (p_i > \theta \text{ and } v_i^j \leq \theta).$$

The cost of the IMM tree is composed of points that are mistakes and those that are not. All the points that are not mistakes contribute in total  $O(dk)$  to the cost. Each point that is a mistake contributes at most  $d$  to the cost. We will show that there are only  $O(k \log k)$  mistakes and this completes the proof.

We consider each level of a tree, where a *level* is a set of nodes with the same number of edges on the path from the root. Due to the definition of the IMM algorithm, each center survives in at most one node in any level. Each center  $v^j$  at a node  $u_{\theta,i}$  can cause at most one mistake, the point in  $C^{v^j}$  with a zero in the  $i$ -th coordinate. Thus at each level there are only  $O(k)$  mistakes. From [22], Section B.5, we know that the depth is  $O(\log k)$ . Thus, the total number of mistakes in the IMM tree is  $O(k \log k)$  and the total cost is  $O(dk \log k)$ .  $\square$

Putting these results together, we are now ready to prove that our algorithm produces the optimal clustering on the Synthetic II dataset  $\mathcal{D}$  using a tree with  $O(k \log k)$  leaves.

*Proof of Theorem 4.* We generalize the notion of a mistake introduced in [22]. A mistake in a leaf  $u$  is a point in the leaf that its center is not  $\ell(u)$ , the labeling of  $u$ . Note that the number of mistakes in IMM upper bounds the number of mistakes in the leaves at the beginning of ExKMC run, which implies that at the beginning there are only  $O(k \log k)$  mistakes in the leaves.

We will show that at each iteration of ExKMC, either there are no mistakes in the leaves or there is a split that will not incur more mistakes and actually reduce the number of mistakes. We will show that this will imply that the surrogate cost decreases by  $\Omega(d) = c_1 d$ , for some constant  $c_1 > 0$ . A split that does not change the number of mistakes, or even increase it will have a smaller gain. Thus, number of mistakes will guarantee to decrease. This will immediately prove that after  $O(k \log k)$  iterations of ExKMC the clustering is the one defined by  $\mathcal{M}$ . We want to prove something stronger, we want to analyze the trade-off between complexity (number of leaves added) and accuracy (the surrogate cost).

Let us analyze the approximation ratio as a function of the number of leaves added to the tree. If there are no mistakes, then we have found the optimal clustering. Otherwise, the total surrogate cost after IMM tree is  $O(dk \log k) = c_2 \cdot dk \log k$ , for some constant  $c_2$ , and each new leaf will decrease the cost by  $c_1 \cdot d$ . Together with the optimal cost being  $\Theta(kd) = c_3 kd$ , for some constant  $c_3 > 0$ , we deduce that if there are  $k'$  leaves (equivalently,  $k' - k$  iterations of ExKMC) and there are still mistakes, the approximation ratio is bounded by

$$\frac{c_2 dk \log k - c_1 d(k' - k)}{c_3 dk} = \frac{c_2}{c_3} \cdot \log k - \frac{c_1}{c_3} \cdot \frac{k'}{k} + \frac{c_1}{c_3}$$

In different words, there is a constant  $c > 0$  such that the approximation ratio is bounded by

$$O\left(\max\left\{\log k - c \cdot \frac{k'}{k}, 1\right\}\right).$$

Let us prove that if there is a mistake in a leaf, then some split decreases the surrogate cost by  $\Omega(d)$ . If there is leaf  $u$  with a point  $\mathbf{p}$  that is a mistake, then  $\mathbf{p}$  does not belong to the codeword  $v^{\ell(u)}$ . There is a feature  $i$  where  $p_i \neq 0$  and  $p_i \neq v_i^{\ell(u)}$  (actually there are  $\alpha d - 1$  such coordinates, and any one of them will be good). Without loss of generality, assume that  $p_i = -1$ . Focus on the split  $(i, \theta)$  with  $\theta = -0.5$ . Denote by  $C^{\mathbf{p}}, C^{-\mathbf{p}}$  the partition of points in  $u$  defined by this split. Suppose that  $\mathbf{p} \in C^{\mathbf{p}}$ . Denote by  $C^{\ell(u)}$  all the points in  $u$  that belongs to the cluster  $\ell(u)$ . This choice of threshold ensures that all points in  $C^{\ell(u)}$  are in  $C^{-\mathbf{p}}$  and not in  $C^{\mathbf{p}}$ . This means that all points in  $C^{\ell(u)}$  are now in a different cluster than  $\mathbf{p}$  after this split. In different words, all points that were not mistakes as they were in  $C^{\ell(u)}$  will remain as non-mistakes. Furthermore, this split will make  $\mathbf{p}$  a non-mistake. In  $C^{\mathbf{p}}$  there are at most  $O(k \log k)$  points (because only mistakes can be in  $C^{\mathbf{p}}$ ). Using Claim 1, their cost from changing the center can increase by at most  $O(\sqrt{d} \log k)$ , and they decrease the surrogate cost by  $\Omega(d)$ . Thus the decrease in this split is at least  $\Omega(d - \sqrt{d} k \log^2 k) = \Omega(d)$ . As the ExKMC algorithm chooses the split minimizing the surrogate cost, the cost decreases by  $\Omega(d)$  in this step.

The last thing we need to show is that for every split with  $\Omega(d)$  gain, the number of mistakes must go down. Suppose the algorithm made a split at node  $u$  where previously there were  $n_u$  points and  $m_u$  mistakes. After the split there are  $n_r$  nodes on the right successor and  $n_\ell$  on the left,  $n_r + n_\ell = n_u$  and  $m_r$  mistakes on the right and  $m_\ell$  mistakes on the left.

Letting  $\Delta = 2\sqrt{d} \ln \frac{k}{10} = \Theta(\sqrt{d})$ , the cost of each node is, by Claim 1,

$$\# \text{mistakes} \cdot \left( \frac{d}{2} \pm \Delta \right) + \# \text{non-mistakes}.$$

Thus, the maximal gain of a split is achieved in case  $u$  has the largest cost and its successors the smallest. Specifically, the maximal gain is

$$\left[ m_u \left( \frac{d}{2} + \Delta \right) + (n_u - m_u) \right] - \left[ m_r \left( \frac{d}{2} - \Delta \right) + (n_r - m_r) \right] - \left[ m_\ell \left( \frac{d}{2} - \Delta \right) + (n_\ell - m_\ell) \right].$$

The last term is equal to

$$(\Delta - 1)(m_u - m_r - m_\ell).$$

For the gain to be  $\Omega(d)$ , or even positive, the number of mistakes must decrease.  $\square$

## B More Experimental Details

### B.1 Algorithms and Baselines

Our empirical evaluation compared the following clustering methods.

- CART [15]: Each data point was assigned with a class label, based on the clustering result of the near-optimal baseline. We have used `sklearn` implementation of decision tree minimizing the gini impurity. Number of leaves was controlled by `max_leaf_nodes` parameter.
- KDTree [9]: For each tree node the best cut was chosen by taking the coordinate with highest variance, and split according to the median threshold. The size of the constructed tree is controlled through `leaf_size` parameter, since the splits are always balanced we obtained a tree with up to  $k'$  leaves by constructing a KDTree with `leaf_size` =  $\lfloor \frac{n}{k'} \rfloor$ . Each of the  $k'$  leaves was labeled with a cluster id from 1 to  $k$ , such the  $\widetilde{\text{cost}}$  will be minimized with the centers of the near-optimal baseline.

- CUBT [28]: Constructed clustering tree using CUBT R package [31]. CUBT algorithm is composed out of three steps: (1) build a large tree, (2) prune branches, and (3) join tree leaves labels. Each step is controlled by multiple parameters. For each step we applied grid-search over multiple parameters, and selected the parameters that minimize the final tree cost. The hyper-parameters were chosen based on [28] recommendation, and available in B.3. To construct a tree with  $k'$  leaves the we have set `nleaves` =  $k'$  in the prune step, and to verify that only  $k$  clusters will be constructed we have set `nclass` =  $k$  in the join step.
- CLTree [43]: Constructed clustering tree using CLTree Python package [19]. CLTree algorithm first construct a large tree with up to `min_split` samples in each leaf, and afterwards prune its branches. Pruning step is controlled by `min_y` and `min_rd` parameters. We applied grid-search over those parameters (values specified in B.3), for each combination we counted the number of leaves, and for each number of leaves we have taken the tree with minimal cost.
- ExKMC: Applied our proposed method for tree construction (Algorithm 1), that minimize  $\widetilde{\text{cost}}$  at each split with the centers of the near-optimal baseline.
- ExKMC (base: IMM): Constructed a base tree with  $k$  leaves according to the IMM algorithm [22]. The tree was expanded with our proposed method (Algorithm 1) that minimize  $\widetilde{\text{cost}}$  at each split. Both IMM and the ExKMC method used the centers of the near-optimal baseline.

For each combination of parameters, if execution time was more than 1 hour, the execution was terminated and ignored. Execution termination occurred only with CUBT and CLTree over the larger datasets.

## B.2 Datasets

Datasets in the empirical evaluation are depicted in Table 1.

Categorical values and class labels were removed from the datasets. The number of clusters,  $k$ , was set to be equal to the number of class labels. For the 20newsgroups dataset, documents were converted to vectors by removal English stop-words and construction of word count vectors, ignoring rare terms (with document frequency smaller than 1%).

**Synthetic I dataset.** The dataset of synthetic I is a slight adaptation of the one described in [22], which was designed the highlight the weakness of CART algorithm. It contains 5,000 points:

- Two of them are  $(\nu, 1, 1, \dots, 1)$  and  $(\nu, 0, 0, \dots, 0)$ , where  $\nu = 1000$  is a large number.
- Half of the remaining points are 0 in the first feature and another random 100 features are also 0, all the remaining features are 1.
- The remaining points also have zero in the first feature, and the other random 100 features are set to 1 and the rest of the features are 0.

Properties of synthetic II are described in Section A.1.

## B.3 Hyper-parameters

Grid search was executed on the following hyper-parameters values:

Table 1: Datasets properties

Dataset	$k$	$n$	$d$
<b>Small Datasets</b>			
Iris [27]	3	150	4
Wine [26]	3	178	13
Breast Cancer [26]	2	569	30
Digits [41]	10	1,797	64
Mice Protein [33]	8	1,080	69
Anuran Calls [49]	10	7,195	22
<b>Larger Datasets</b>			
Avila [24]	12	20,867	10
Covtype [12]	7	581,012	54
20 Newsgroups [36]	20	18,846	1,893
CIFAR-10 [40]	10	50,000	3,072
<b>Synthetic Datasets</b>			
Synthetic I	3	5,000	1,000
Synthetic II	30	30,000	1,000

- CUBT:

`minsize`  $\in [5, 10, 15]$

`mindev`  $\in [0.001, 0.7, 0.9]$

`mindist`  $\in [0.3, 0.5]$

`alpha`  $\in [0.2, 0.4, 0.6]$

To construct a tree with  $k'$  leaves the we have set `nleaves` =  $k'$  in the prune step, and to verify that only  $k$  clusters will be constructed we have set `nclass` =  $k$  in the join step.

- CLTree:

`min_split`  $\in [10, 20, 30, 50, 100, 200, 500, 1000, 1500, 2000]$

`min_y`  $\in [0.1, 0.2, 0.3, 0.4, 0.5, 1, 1.5, 2, 2.5, 3, 3.5, 4, 4.5, 5]$

`min_rd`  $\in [1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 15, 20, 25, 30, 50]$

## B.4 Extra experiments

Figure 4 depicted the run time of **ExKMC** constructing a tree with  $2k$  leaves using a single process. The operations of **IMM** and **ExKMC** can be feature-wise paralleled, where **KMeans** iterations can also be executed in parallel. Figure 6 compare the running times of single processor and four processor. Using four process **ExKMC** constructs tree with 20 leaves over CIFAR-10 dataset in less than 7 minutes.

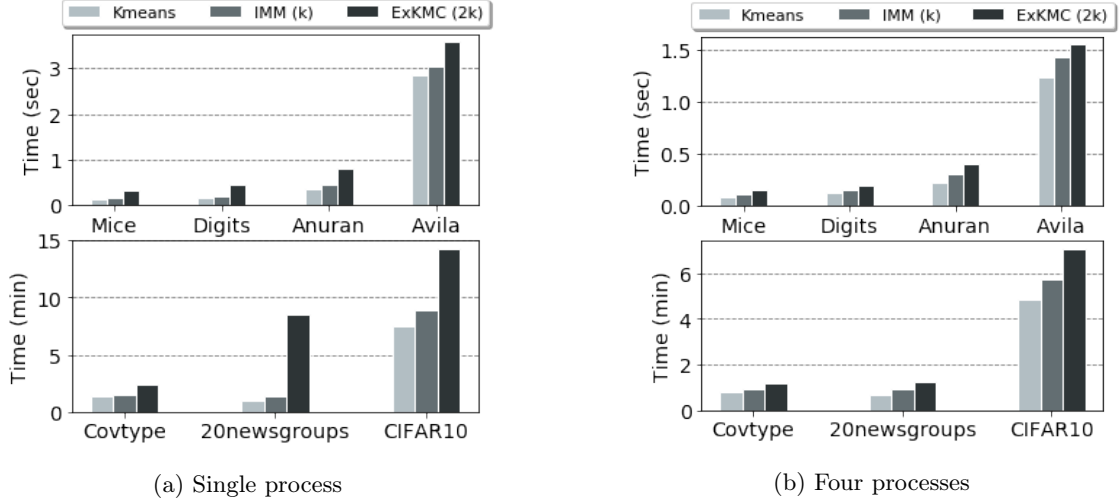


Figure 6: Running times of constructing a tree-based clustering with  $2k$  leaves. We consider both a single processor (left) and a parallel version using four processes (right). The y-axis labels differ between left and right graphs. Overall, parallelism improves the running time of ExKMC by  $2\text{--}3\times$ .

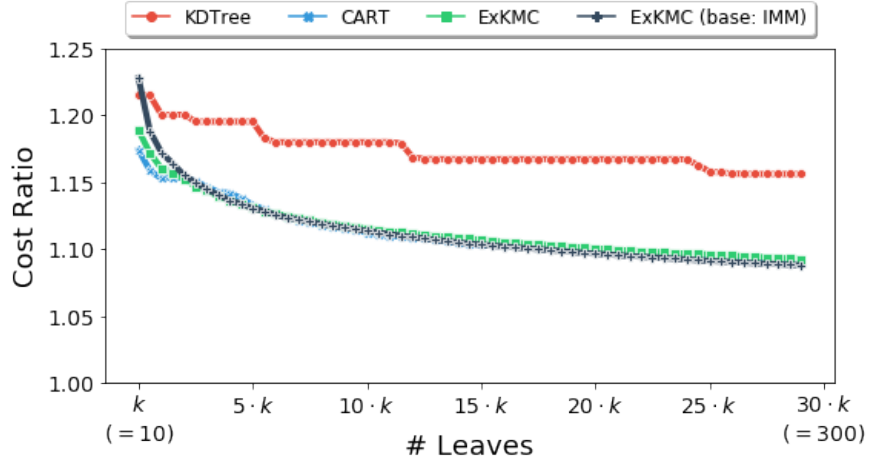


Figure 7: **CIFAR-10 Convergence Rate.** We separately showcase CIFAR-10 because it is an exception to many trends. The algorithms fail to converge to a cost ratio of 1.0 compared to the reference clustering. This is likely because pixels are not good features in this context, since the trees only use a subset of them, one at a time. We also notice that IMM starts with the worst performance, but when expanded using ExKMC, eventually outperforms the competitors (for  $k' > 10k$ ).

### Zooming in on the Cost for Small Datasets

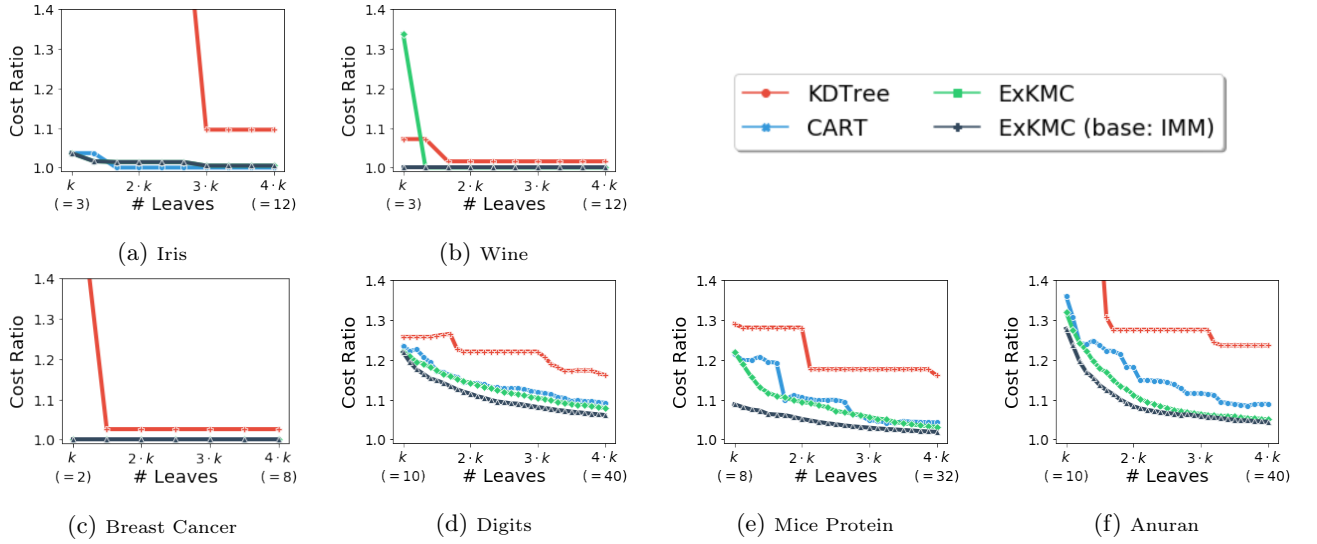
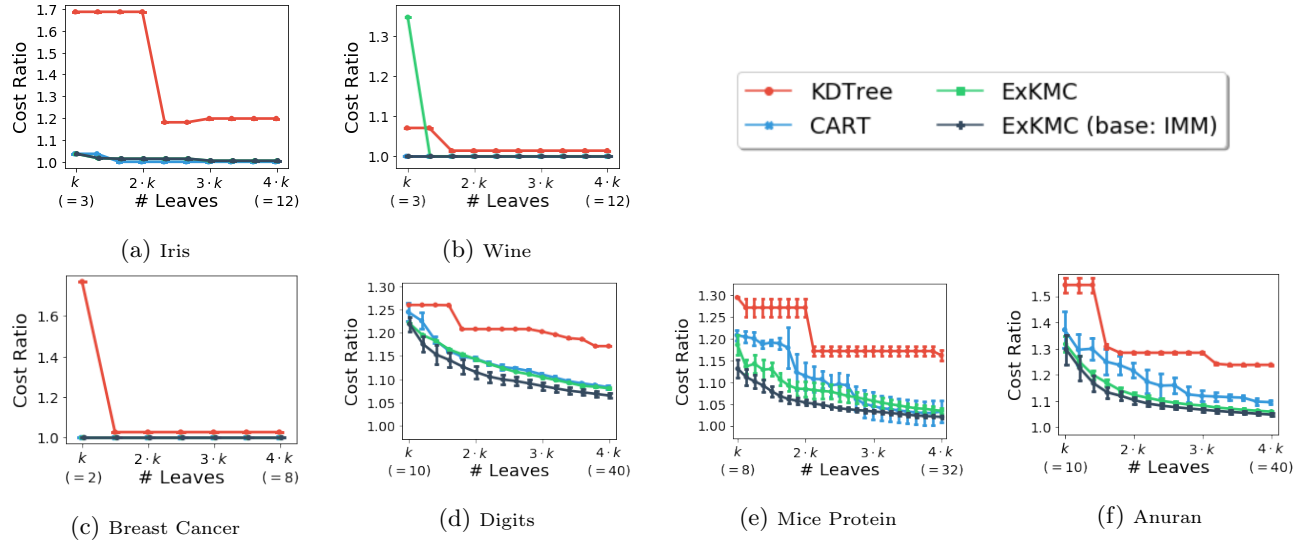


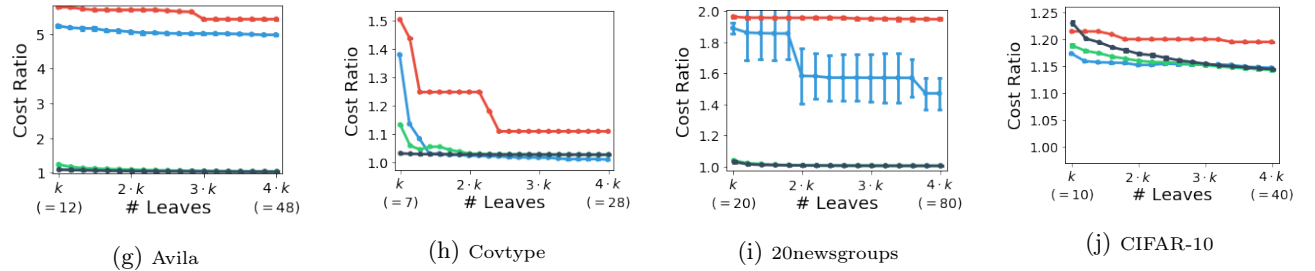
Figure 8: Results of small datasets presented in Figure 2 where cost is in range 1.0 – 1.4.



### Small Datasets ( $k$ -means cost ratio, comparing five initializations)



### Larger Datasets ( $k$ -means cost ratio, comparing five initializations)



### Synthetic Datasets ( $k$ -means cost ratio, comparing five initializations)

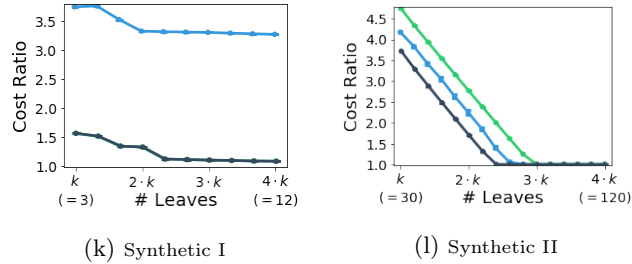


Figure 9: Aggregation of results presented in Figure 2 over five different executions of  $KMeans$ . All tree construction algorithms that were examined are deterministic, but different initialization of centers to  $KMeans$  may result in different reference clustering. Points are the median of the five executions, and error bars are the standard deviation. In many datasets, the deviations are negligible or even equal to 0. When the number of leaves surpasses  $2k$  the results of all algorithms remain separated in the presence of deviations, verifying the significance and reproducibility of the results.

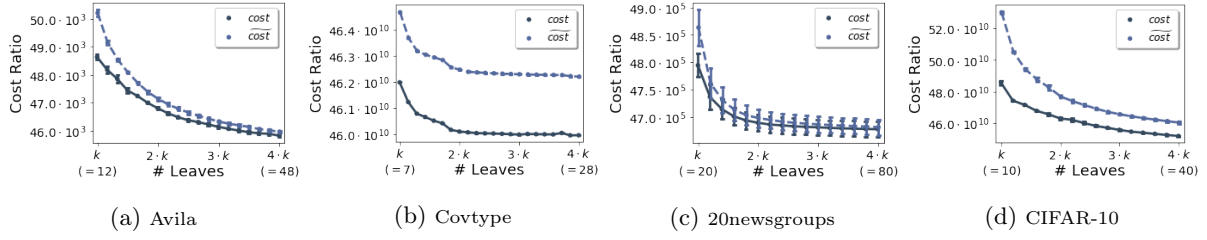
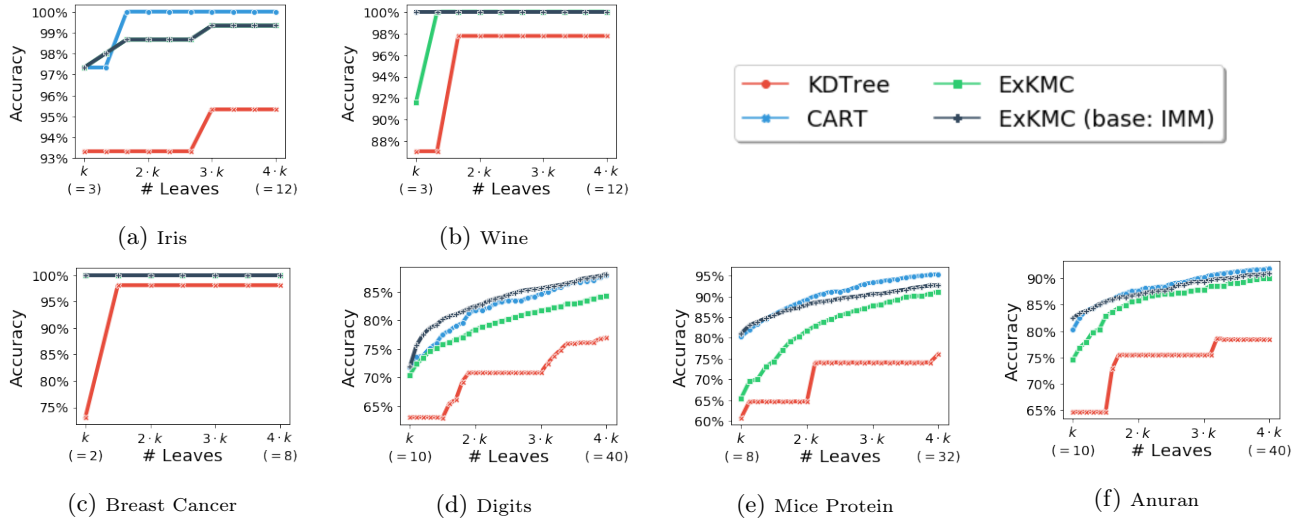
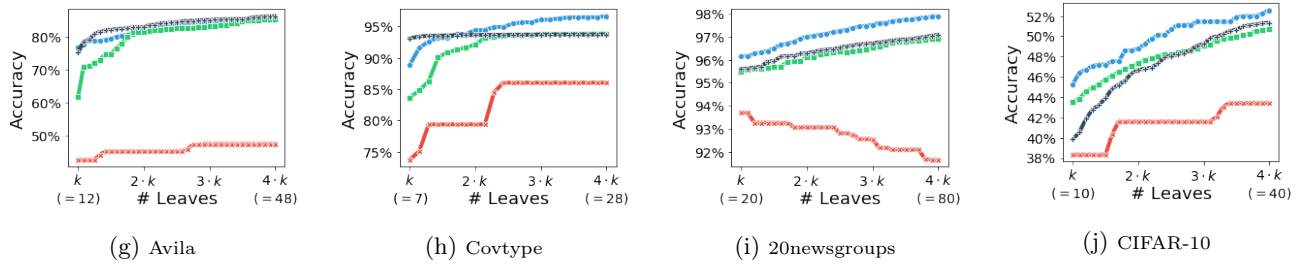


Figure 10: Aggregation of results presented in Figure 5 over five different executions of **KMeans**. For three of the datasets, the deviations are negligible. For 20newsgroups, we see that both the  $k$ -means and the surrogate cost have relatively higher variance. Overall, taking the best clustering of the five initialization leads to consistent and reproducible results.

### Small Datasets (Clustering accuracy)



### Larger Datasets (Clustering accuracy)



### Synthetic Datasets (Clustering accuracy)

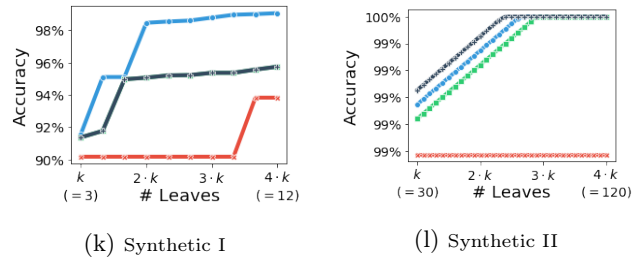


Figure 11: Results of clustering accuracy experiments over 12 different datasets. Graphs depict the accuracy of the tree-based clustering with respect to the reference  $k$ -means ( $y$ -axis, best result is 100%) as function of the number of tree leaves ( $x$ -axis).