XClusters: Explainability-First Clustering

Hyunseung Hwang, Steven Euijong Whang
KAIST
{aguno, swhang}@kaist.ac.kr

Abstract

We study the problem of *explainability-first clustering* where explainability becomes a first-class citizen for clustering. Previous clustering approaches use decision trees for explanation, but only after the clustering is completed. In contrast, our approach is to perform *clustering and decision tree training holistically* where the decision tree's performance and size also influence the clustering results. We assume the attributes for clustering and explaining are distinct, although this is not necessary. We observe that our problem is a monotonic optimization where the objective function is a difference of monotonic functions. We then propose an efficient branch-and-bound algorithm for finding the best parameters that lead to a balance of cluster distortion and decision tree explainability. Our experiments show that our method can improve the explainability of any clustering that fits in our framework.

Introduction

Explainable AI is becoming critical as AI is widely used in our everyday lives. A fundamental issue is that models are usually optimized for accuracy before being explained. As a result, significant effort is needed to make sense out of trained models, especially for complex ones. Even so, the explainability may not be sufficient to fully trust.

Instead, we contend that explainability must be a first-class citizen instead and focus on unsupervised learning. In particular, we propose the new problem *explainable-first clustering* where clustering is performed while balancing accuracy and explainability. As a motivating example, consider a temporal relational database that shows credit card spending trends for customers. In order to identify the key trends, a straightforward approach is to cluster the trends by Euclidean or Dynamic Time Warping (DTW) (Berndt and Clifford 1994) distances. However, when explaining the clusters, each cluster may be a collection of a wide range of customer demographics, which makes it difficult to explain concisely. Instead, we would also like the clusters to be easy to describe as well.

More formally, we assume that the clustering is explained using decision trees, a problem that has attracted significant interest lately under the name of explainable clustering (Moshkovitz et al. 2020; Laber and Murtinho 2021; Makarychev and Shan 2021; Gamlath et al. 2021; Bandyapadhyay et al. 2022). We consider smaller decision trees (i.e., they have fewer nodes) to be more explainable as they are easier to read (Lipton 2018). The features that are used for clustering are not necessarily the same as the ones for training the decision tree as in our motivating example. Our goal is to perform any clustering while ensuring that the decision tree trained on the clusters is as small as possible. There is a natural tradeoff between the cluster distortion and explainability, which we attempt to balance by solving an optimization problem. We call our method XClusters as it explicitly optimizes for explainable clusters.

We note that most explainable clustering approaches assume a fixed reference clustering where the clustering is performed *before* being explained (Moshkovitz et al. 2020; Laber and Murtinho 2021; Makarychev and Shan 2021; Gamlath et al. 2021). Hence, the goal is to minimize the error of the decision tree that classifies examples to clusters. Even for a recent work that removes outliers from clusters for better explainability (Bandyapadhyay et al. 2022), the clustering is still done prior to outlier removal. In addition, if the data does not have many outliers, then removing data for the sake of explainability may result in incorrect clustering. While these approaches have the advantage of requiring little work on the clustering side, we contend that explainability must be incorporated in the clustering itself rather than using a single reference clustering for a truly explainable clustering. XClusters is thus *an orthogonal approach to existing explainable clustering techniques* where it can plug in any decision tree training within the novel holistic optimization of clustering and decision tree training.

Figure 1 illustrates how XClusters can trade off the cluster distortion for better explainability. Suppose that there are time-series trends that need to be clustered, but also explained using separate demographic features A, B, and C. Suppose that clustering by trend and then training a decision tree on top of the clusters results in the top figure. While most explainable clustering works take this approach, XClusters can further explore scenarios where the clustering also reflects demographic similarity and results in a smaller decision tree (bottom left) or the number of clusters is increased to reduce their distortion (bottom right). Notice that these two objectives affect each other and thus cannot be achieved separately. For example, reducing the cluster dis-

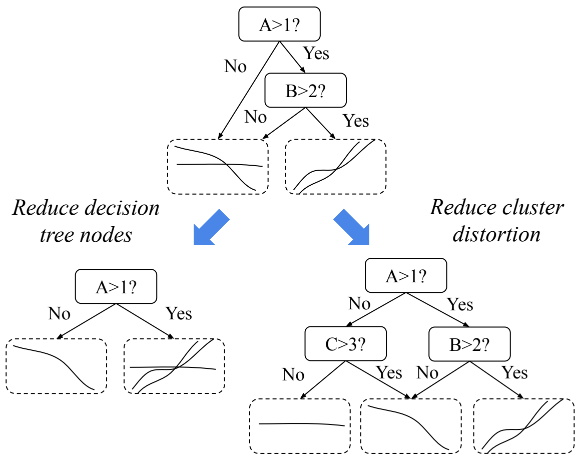


Figure 1: XClusters' strategy for training a decision tree during clustering. There are two objectives: reduce the cluster distortion for better clustering and reduce the decision tree nodes for better explainability. XClusters adjusts the number of clusters and ratio between accuracy and explainability features (explained later) to balance the objectives.

tortion results in a larger decision tree. The end goal is to minimize the decision tree's size and the distortion together.

Our problem is a global optimization that is not convex. However, we make the interesting observation that the objective function is a difference-of-monotonic functions. We exploit this information and utilize existing monotonic optimization techniques (Matthiesen et al. 2020; Hellings et al. 2012), but tailor them to our problem setup. In particular, we propose an efficient branch-and-bound algorithm that tunes the parameters.

In our experiments, we evaluate XClusters on various real time series relational datasets and show how it outperforms baselines that explain only after clustering or perform global optimization without exploiting monotonicity.

Preliminaries

Explainability has been studied extensively (Ribeiro, Singh, and Guestrin 2016), but is fundamentally a subjective notion that depends on whether it is useful to users. Even for decision trees, the common view is that it is explainable, but others disagree as decision trees can be arbitrarily complex and large. In addition, even if there is some explainability, there are few proposals on how to quantify it.

Explainability Measure We thus propose a concrete explainability measure for understanding clustering results. Suppose that there is a decision tree that is trained to classify examples to clusters. We measure the explainability of the decision tree using the classification accuracy and the size of the decision tree. A decision tree can only be explainable if it is accurate in the first place. In addition, the smaller the decision tree, the easier it is to read (Lipton 2018). If we assume an acceptable minimal accuracy of an explainable decision tree, then explainability can be measured as the size of the tree that has at least that accuracy. We measure the size of a decision tree by counting its nodes. Although other approaches count the number of levels, a tree that is shallow, but very bushy, may be difficult to understand.

Accuracy Features		Explainability Features		
Time	Amount	Age range	Zip code	Online
0:50	50	10–20	456	Yes
2:00	300	30–40	123	No
2:30	200	50–60	999	No

Table 1: A sample credit card transaction table where each row contains a timestamp, transaction amount, and demographic features of the user. If we would like to analyze common spending trends and explain with demographics, then the time and amount columns become the accuracy features, and the others the explainability features.

Accuracy and Explainability Features We assume there are two groups of features: *accuracy features* and *explainability features*. The accuracy features are meant to maximize the clustering accuracy and thus lower cluster distortion. For example, if we are clustering time-series data, then the trends over time become the accuracy features. On the other hand, the explainability features are used to train a decision tree to classify examples to clusters for explanation. For example, if the time-series data is also relational with person attributes, then we can train a decision tree using the person attributes as the explainability features as illustrated in Table 1. This type of data is common in finance (e.g., credit card companies or investment banks) or shopping (e.g., Amazon) industries where credit card transactions, stock values, and item purchases for certain demographics need to be analyzed over time. We assume that the two groups of features are not identical where clustering on the accuracy features leads to clusters with less distortion than when clustering on the explainability features for explaining purposes. It is still possible to use the same set of features for both accuracy and explainability, but then the distance functions applied on the features must be different in order to adjust the explainability during clustering. Our setup is not limited to time-series relational data and can be applied to various types of data including image data with metadata or any relational data.

Clustering We assume any clustering algorithm (Jain, Murty, and Flynn 1999) that uses a distance function for clustering. We are thus not limited to a specific algorithm, although we do assume the clustering uses a distance function. As a default, we use DTW as the distance function as it is widely used and effective in identifying similar trends. We also use k-medoids clustering (Kaufman and Rousseeuw 2008) as a default as it works naturally with DTW distances. When measuring distortion, we take the sum of squares of DTW distances from the examples to their closest center points. The center point of a cluster is its clustroid, which is the example with the lowest mean squared distances to the rest of the points in the cluster. Note that we cannot use centroids because we do not assume a Euclidean space.

For the distance function, there are two distances using the accuracy features (*a-distance*) and explainability features (*e-distance*). For example, one can use DTW for the trend distance and Jaccard distance for the explainability distance. A straightforward combination of the two measures is to normalize them and then take a weighted sum where α is a parameter used for the balancing:

$$\frac{(1-\alpha) \times \text{a-distance}}{\max\{\text{All a-distances}\}} + \frac{\alpha \times \text{e-distance}}{\max\{\text{All e-distances}\}}$$

Decision Tree Training We use decision trees to explain the clustering as they are widely considered more interpretable than other models. An alternative way to explain clusters is to use rule-based approaches. For example, a general boolean formula (GBF) (Singh et al. 2017) is an if-then-else rule that can succinctly describe a cluster. XClusters can possibly be extended to GBFs because they can also be expressed as decision trees.

Our framework is agnostic to the decision tree training algorithm. Although one can use recent techniques that attempt to optimize the decision tree to the clustering, it is also fine to use any other algorithm. What matters is the relative size of the decision trees, which we would like to minimize.

Problem Definition

We formulate our problem as minimizing two values: (1) D, which we define as the cluster distortion where a lower value is better as we would like the clusters to be coherent and (2) N, which we define as the number of decision tree nodes. Given that the decision tree's accuracy is sufficiently high, a smaller number of nodes is better as it means the decision tree is easy to read and thus explainable (Lipton 2018).

$$\min_{k,\alpha} \; D(k,\alpha) + \lambda N(k,\alpha) \tag{1}$$

where we omit the normalization of D and N for brevity.

We use two parameters to adjust D and N: k is the number of clusters, and α is used to balance the accuracy and explainability distances as explained above. k is suitable when using clustering algorithms like k-medoids, k-means, or hierarchical clustering, but one can use a different parameter depending on the clustering algorithm.

Without any assumptions, this global optimization problem is non-convex. In addition, the D and N objectives are not independent. For example, if we increase k in order to decrease D, that also means N is likely to increase due to more clusters. While one can use black-box optimization techniques like Bayesian Optimization (Mockus 1974), we instead utilize monotonicity properties between the parameters and objectives for faster optimization as we explain from the next section.

Monotonic Optimization

We propose two practical monotonicity properties of the D and N objectives, which enable fast optimization.

- *As k increases, D is decreasing while N is increasing.* As k increases, there are more clusters, which means they tend to be smaller on average, causing the distortion to decrease. However, it becomes more difficult for a decision tree to precisely classify examples into the clusters.
- *As α increases, D is increasing while N is decreasing.* As α increases, the distance function is more about the explainability feature distance, so the clusters become less compact, causing the distortion to increase. On the other hand, the decision tree can be trained easier and thus requires fewer nodes to be sufficiently accurate.

In general, the monotonocity properties are not guaranteed to hold due to incomplete and imperfect data. For example, an increase in k could suddenly cause one of the clusters to increase in size and thus increase the overall distortion. Or an increase in α may actually make the clusters more compact because the explainability features happen to be more effective than the accuracy features in reducing distortion.

However, our approach is inspired by learning lattice networks using partial monotonic functions (Gupta et al. 2016; You et al. 2017) where models are trained with monotonicity assumptions on model predictions against certain input features using domain knowledge. For example, if users are searching for restaurants on a website, it is reasonable to assume that they will more likely click on restaurants with higher ratings although there are exceptions where a user may want to explore new restaurants with lower ratings. Using this property is said to result in more accurate and flexible model training. Another analogy can be found in k-means clustering where an elbow method uses binary searching to find a k value assuming that a larger k results in a lower cluster distortion, even though there is no absolute guarantee of this trend. Likewise, our monotonicity properties can be thought as domain knowledge where given complete data, they should hold. In the worst case, XClusters will return a sub-optimal result while still being efficient. In our experiments on real datasets, we empirically show that the monotonicity properties mostly hold.

Using the monotonicity properties, we can re-formulate our problem into a monotonic optimization. Notice that increasing or decreasing k and α have opposite affects on D and N. We can thus rewrite the objective function of minimizing $D(k,\alpha) + \lambda N(k,\alpha)$ as minimizing $F(k,\alpha) - G(k,\alpha)$ where F captures the decreasing D and λN values when α and k increase, respectively, and G captures the decreasing D and λN values when α and k decrease, respectively. We cannot directly compute F and G as they are embedded in D and N, but we only require their existence. The new formulation is then a difference of two monotonically increasing functions, which is a well-known optimization problem (Alizamir 2009). Hence, our optimization problem can also be written as:

$$\min_{k,\alpha} \; F(k,\alpha) - G(k,\alpha) \tag{2}$$

where F and G are decreasing functions of k and α.

Methodology: XClusters

We now solve the monotonic optimization problem efficiently. There are largely two known methods for solving a difference-of-monotonic (DM) function problem: the

Algorithm 1: XClusters algorithm

Input: training data S, maximum k value $k_{\max}$
Parameters: k, α
Output: clusters and decision tree

```
1:  $B \leftarrow [(1,0),(k_{\max},1)]$
2:  Compute upper and lower bounds of $B$
3:  $B^* \leftarrow B$
4:  $Q.push(B)$
5:  while $\neg Q.empty()$ do
6:      $B \leftarrow Q.pop()$ // Block with lowest lower bound
7:      if $B$'s normalized $k$ width is longer than the normalized $\alpha$ width then
8:          $\{B_1, B_2\} \leftarrow$ Split $B$ by $k$ into two blocks
9:      else
10:         $\{B_1, B_2\} \leftarrow$ Split $B$ by $\alpha$ into two blocks
11:     Compute upper and lower bounds of $B_1$ and $B_2$
12:     $Q.push(\{B_1, B_2\})$
13:     if $\min_{B \in Q} B.upper() < B^*.upper()$ then
14:         $B^* \leftarrow \arg\min_{B \in Q} B.upper()$
15:     $Q \leftarrow Q \setminus \{B' \in Q | B'.lower() + \epsilon_b \geq B^*.upper()\}$
16: return Clusters and decision tree of $B^*.upper()$
```

Polyblock algorithm (Alizamir 2009; Tuy 2000; Tuy, Al-Khayyal, and Ahmed 2001) and branch-and-bound algorithm (Tuy, Minoux, and Hoai-Phuong 2006; Tuy 2005). The Polyblock algorithm is originally used to solve problems with monotonic objective functions and normal set constraints, which satisfies the condition that for any $x, y \in \mathbb{R}^n$, $x \in S$ and $y \leq x$ implies that $y \in S$ where S is a normal set (Cheon, Al-Khayyal, and Ahmed 2005). A DM problem can be converted to a monotonic optimization problem by introducing an auxiliary dimension. However, the Polyblock algorithm is known to be slow in practice (Hellings et al. 2012). Another popular method is the branch-reduce-and-bound (BRB) algorithm, which uses a branch-and-bound strategy to search parameters, but also has a reduce stage to improve convergence. The BRB algorithm is known to be faster than Polyblock and is thus more widely used.

We propose an efficient algorithm based on the branch-and-bound algorithm that is suitable for our problem (see Algorithm 1). Our algorithm extends a branch-and-bound algorithm (Hellings et al. 2012) that does not assume normal set constraints. The goal is to minimize the objective score $D + \lambda N$ by iteratively searching blocks of k and α ranges.

For each block, we compute a lower and upper bound of the objective score. The conventional way to find a lower bound of a block is to subtract the F value at the bottom-left point by the G value at the top-right point exploiting the monotonicity properties. However, in our setting, we do not know the exact F and G functions and need to compute the lower bound using D and N only. Suppose that the bottom-left point is (k_1, α_1), the top-right point (k_2, α_2), and $k \in [k_1, k_2]$ and $\alpha \in [\alpha_1, \alpha_2]$. We know that

$$\begin{aligned} F(k,\alpha) - G(k,\alpha) &= D(k,\alpha) + \lambda N(k,\alpha) \\ &\geq D(k_2,\alpha) + \lambda N(k_1,\alpha) \\ &\geq D(k_2,\alpha_1) + \lambda N(k_1,\alpha_2). \end{aligned}$$

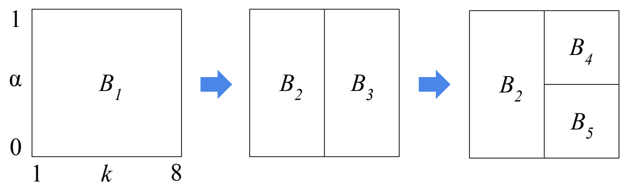


Figure 2: The XClusters algorithm iteratively splits blocks while pruning blocks that are not worth exploring based on their lower and upper bounds.

We thus use $D(k_2, \alpha_1) + \lambda N(k_1, \alpha_2)$ as the lower bound. Notice that we need to perform clustering and decision tree training to compute each of the D and N values, so minimizing the number of these trainings is important for efficiency. The upper bound can be computed by computing $D(k, \alpha) + \lambda N(k, \alpha)$ on an arbitrary point in the block. To save computation, we use the minimum of $D(k_2, \alpha_1) + \lambda N(k_2, \alpha_1)$ and $D(k_1, \alpha_2) + \lambda N(k_1, \alpha_2)$, whose components are already computed at this point to derive the lower bound. We avoid redundant clustering and decision tree training by keeping track of all (k, α) results.

We iteratively split the block with the lowest lower bound score by its longer normalized width until there is no block to split. When splitting by α, we simply divide the α range in half. When splitting by k, we split $[k_1, \ldots, k_n]$ into $[k_1, \ldots, k_{\lfloor \frac{n}{2} \rfloor}]$ and $[k_{\lfloor \frac{n}{2} \rfloor}, \ldots, k_n]$. Starting the second range with $k_{\lfloor \frac{n}{2} \rfloor}$ instead of $k_{\lfloor \frac{n}{2} \rfloor + 1}$ is intentional to save computation when computing lower and upper bounds of the split blocks. After generating two blocks B_1 and B_2, we check if they can be pruned. Notice that a block's upper bound is an actual objective score for some parameter values while the lower bound is potentially lower than the actual lowest objective score. We thus discard any block whose lower bound plus some tolerance ϵ_b is larger than or equal to any upper bound of another block. Using a higher ϵ_b results in more pruning at the cost of a sub-optimal result.

As a running example, suppose the algorithm starts with $k \in [1, 2, \ldots, 8]$ where $k_{max} = 8$, $\alpha \in [0, 0.1, 0.2, \ldots, 1]$, and $\epsilon_b = 0.1$ (see Figure 2). The entire block is denoted as $B_1 = [(k = 1, \alpha = 0), (8, 1)]$. Suppose we split B_1 into $B_2 = [(1, 0), (4, 1)]$ and $B_3 = [(4, 0), (8, 1)]$, and they have lower and upper bound ranges of $[0.3, 0.4]$ and $[0.2, 0.5]$, respectively. Then, $B^* = B_2$ because B_2 has a smaller upper bound. We next split B_3 as it has a smaller lower bound. If the resulting blocks B_4 and B_5 have lower bounds at least $B^*.lower() + \epsilon_b = 0.4 + 0.1 = 0.5$, we can prune them as we know B^* is a better block. We continue until Q is empty and then return the clustering and decision tree of B^*'s upper bound result.

The complexity of Algorithm 1 to obtain an ϵ_b-optimal solution can be derived using Theorem 4 in (Vavasis 1995) and is $O\left(\left(\frac{p}{\epsilon_b}\right)^{\frac{2}{q}}\right)$ assuming that $D + \lambda N$ is q-times differentiable, and the q^{th} derivative is bounded by p. Here q and p can be viewed as constants that depend on the properties of $D + \lambda N$ (Hellings et al. 2012). This result shows how fast XClusters optimizes when monotonicity holds perfectly. The less monotonicity holds, the worse XClusters performs.

Experiments

Datasets We use three real time series relational datasets.

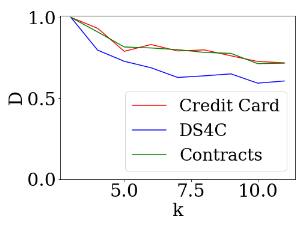
- *Credit Card*: a proprietary dataset used in a major payment processing company. This dataset contains transactions of the credit card users before and after the COVID-19 outbreak (Dec. 2019 – Jan. 2021) containing 5.7 billion transactions amounting to $120B USD. The transactions include online or offline purchases made by each age group and gender for different business categories. We consider 2,551 demographics. We take a 90-day moving average of the time-series data.
- *DS4C* (Kim 2020): a public COVID-19 dataset containing patient data, policy data, and provincial data released by the Korea Centers for Disease Control & Prevention (KCDC). We use floating population data of the city of Seoul for each age and gender group (Jan. 2020 – May 2020). We take a 7-day moving average as the floating population data shows a cyclic pattern on a weekly basis.
- *Contracts* (Linville 2022): a public contract dataset maintained by the State of Washington. There are 170K contracts where each one contains customer, contract, vendor, and sales information.

For the three datasets, the accuracy features are the time-series trends – transaction amounts, population, and sales, respectively – while the explainability features are the demographics information.

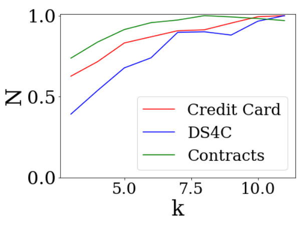
Measures To evaluate a decision tree, we compute a weighted F_1 score on how it classifies examples to clusters. We weight each example in the Credit Card, DS4C, and Contracts dataset by its transaction amount, population, and sales amount, respectively. In our technical report (2022), we also use the Accuracy measure and obtain similar results.

Methods Compared We compare XClusters with baselines that represent the state-of-the-art approaches.

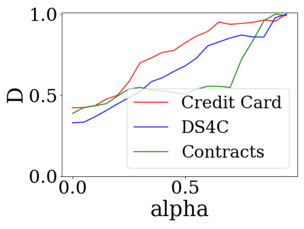
- *2-Step*: Performs clustering and then trains a decision tree on the clusters. This approach represents previous works that assume fixed reference clusters where a decision tree can only be trained afterwards. We use the same clustering and decision tree training algorithms used for XClusters and set $\alpha = 0$. We also fix k using an elbow method.
- *Grid Search (GS)*: Performs clustering and decision tree training using all the possible k values considered by XClusters and a fixed set of α values in the range $[0, 0.05, 0.1, \ldots, 1]$.
- *Bayesian Optimization (BO)* (Mockus 1974): Performs clustering and decision tree training using Bayesian Optimization for tuning k and α. BO is widely used for hyperparameter tuning and has an O(n^3) complexity where n is the number of observations. We set the initial number of examples to explore and the number of iterations to be within the range [10, 30] and tune it to ensure a fair comparison with XClusters.



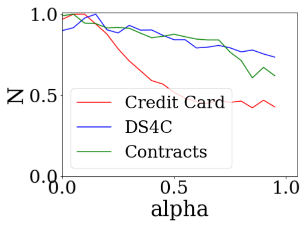
(a) k versus D



(b) k versus N



(c) α versus D



(d) α versus N

Figure 3: Monotonic relationships between k and α, and the normalized objectives D and N for the three datasets.

Other settings We use Scikit-learn (Pedregosa et al. 2011) for the decision tree training (we also evaluate with (Moshkovitz et al. 2020) in our technical report (2022)). For simplicity, we always make the decision tree overfit on the clusters and thus obtain perfect accuracy in classifying examples into clusters. We use the k-medoids algorithm (Kaufman and Rousseeuw 2008) for clustering. Before performing any clustering, we compute the pairwise DTW distances between all example pairs. We search k within the range $[3, 4, \ldots, 11]$ for all datasets. For XClusters, we set $\lambda = 1$, and $\epsilon_b = 0.05$ as default values. We repeat each experiment 10 times. All experiments are performed on a server with Intel Xeon Gold 5115 CPUs.

Monotonicity Properties

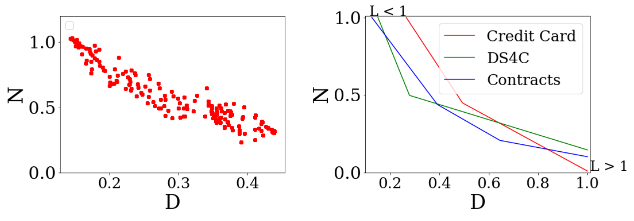
We empirically verify the monotonocity assumptions, which form the basis of our algorithm. Figure 3 shows the trends between either k or α and D or N for all the three datasets. For each k (α), we average the D or N values for all α (k) values considered by the GS baseline. As a result, all the trends are monotonic overall. However, some plots do have occasional violations due to incompleteness in the data. In our technical report (2022), we show similar monotonicity results using other distance measures.

Explainability and Distortion Results

Table 2 compares XClusters with the other baselines in terms of $D + \lambda N$ and runtime. Recall that when running 2-Step, we need to fix k using an elbow method by finding the smallest k where the cluster distortion starts to converge. For the three datasets, the elbow point turns out to be $k = 5$ or $k = 6$. Since 2-Step cannot adjust the clustering for better explainability, it returns the worst results although it runs the fastest. GS usually finds the lowest $D + \lambda N$ with its brute-force searching, but does not outperform XClusters on

Dataset	Method	$D + \lambda N$	D	N	Runtime (sec)
Credit Card	2-Step	$1.005_{\pm 0.000}$	$0.166_{\pm 0.000}$	$0.840_{\pm 0.000}$	$0.426_{\pm 0.034}$
	GS	$0.625_{\pm 0.000}$	$0.391_{\pm 0.000}$	$0.234_{\pm 0.000}$	$64.455_{\pm 0.506}$
	BO	$0.657_{\pm 0.018}$	$0.364_{\pm 0.039}$	$0.293_{\pm 0.049}$	$16.596_{\pm 0.377}$
	XClusters	$0.656_{\pm 0.000}$	$0.385_{\pm 0.000}$	$0.271_{\pm 0.000}$	$6.336_{\pm 0.104}$
DS4C	2-Step	$0.803_{\pm 0.000}$	$0.101_{\pm 0.000}$	$0.703_{\pm 0.000}$	$0.009_{\pm 0.004}$
	GS	$0.494_{\pm 0.000}$	$0.193_{\pm 0.000}$	$0.301_{\pm 0.000}$	$1.359_{\pm 0.026}$
	BO	$0.537_{\pm 0.078}$	$0.126_{\pm 0.005}$	$0.152_{\pm 0.034}$	$1.554_{\pm 0.140}$
	XClusters	$0.547_{\pm 0.000}$	$0.216_{\pm 0.000}$	$0.331_{\pm 0.000}$	$0.122_{\pm 0.001}$
Contracts	2-Step	$0.916_{\pm 0.000}$	$0.103_{\pm 0.000}$	$0.814_{\pm 0.000}$	$0.012_{\pm 0.001}$
	GS	$0.644_{\pm 0.000}$	$0.268_{\pm 0.000}$	$0.375_{\pm 0.000}$	$1.906_{\pm 0.044}$
	BO	$0.650_{\pm 0.002}$	$0.270_{\pm 0.004}$	$0.381_{\pm 0.016}$	$7.265_{\pm 0.562}$
	XClusters	$0.644_{\pm 0.000}$	$0.268_{\pm 0.000}$	$0.375_{\pm 0.000}$	$0.197_{\pm 0.015}$

Table 2: Detailed comparison of XClusters and the three baselines – 2-Step, GS, and BO – using the default parameters. For each result, we show $D + \lambda N$, D, N, and runtime in seconds. We repeat all experiments 10 times and take average values.



(a) Varying k and α (b) Varying λ

Figure 4: D and N tradeoffs on the datasets by varying the k, α, and λ parameters.

the Contracts dataset because it only considers a fixed number of α values whereas XClusters explores an infinite space of values and can thus find better α values. When evaluating BO, we adjust its initial number of examples so that BO's $D+\lambda N$ becomes similar to that of XClusters. As a result, BO obtains a slightly lower $D+\lambda N$ than XClusters for the DS4C dataset, but is 12x slower. In comparison, XClusters exploits the monotonicity properties for fast searching. Moreover, BO is strictly worse than XClusters for the other datasets in terms of $D + \lambda N$ and runtime. While BO's runtime varies significantly depending on its number of initial examples, XClusters does not need such tuning to find near-optimal solutions.

Varying Parameters We study the effect of varying k, α, and λ on the datasets. Figure 4a shows how D and N change for different k and α values on the Credit Card dataset where the two objectives tradeoff as expected according to the monotonicity properties. Figure 4b shows how D and N change against λ when taking the results of GS with the lowest $D + \lambda N$ on the three datasets. A higher λ means that there is more emphasis on explainability, and the decision trees tend to be smaller (lower N) with the risk of increasing the cluster distortions (higher D).

Visualization We visualize clusterings and their decision trees generated from the Credit Card dataset using Graphviz (Ellson et al. 2001) in Figure 5 (see technical report (2022) for larger images). We show two results with low $D + \lambda N$ scores: (a) $k = 5, \alpha = 0.45$ and (b) $k = 6, \alpha = 0.75$. The trees and clusters show how D and N can trade off. We note that some clusters of trends look quite noisy, but that is because the trends are clustered using DTW where they do not have to perfectly align visually to be clustered. Figure 5b's tree is smaller than Figure 5a's and is thus easier to understand. However, the clusters are relatively noisier where similar trends end up in different clusters, for the sake of a simpler decision tree. Which result is more desirable depends on the application and can be configured with λ.

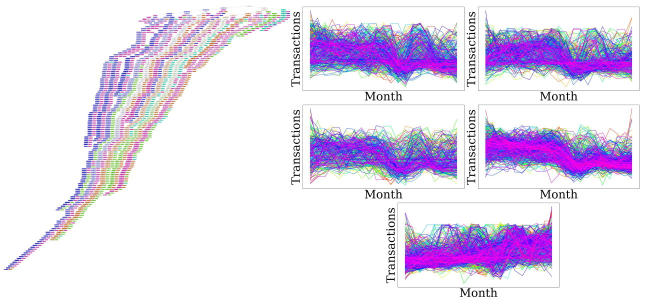
Runtime Varying ϵ_b

We compare XClusters' runtime with the BO baseline while varying ϵ_b, which determines how aggressively XClusters prunes blocks. Table 3 shows that XClusters can be significantly faster than BO on the three datasets while still having a lower or comparable $D + \lambda N$.

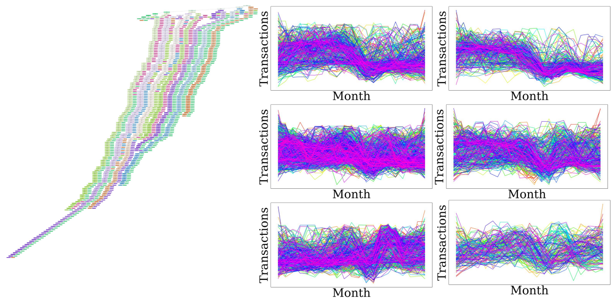
Related Work

Explainable AI is a broad field that attempts to explain trained models (see many surveys (Doshi-Velez and Kim 2017; Lipton 2018; Tjoa and Guan 2020; Arrieta et al. 2020; Molnar 2019)). There are many explaining techniques for classifiers in general (Ribeiro, Singh, and Guestrin 2016, 2018; Wang et al. 2017; Lou, Caruana, and Gehrke 2012; Adebayo et al. 2018). Most of these techniques attempt to explain a model *after* the training is finished. In comparison, our focus is to make a model explainable during its training.

A recent line of work focuses on training decision trees on top of clustering for explanation. Explainable clustering (Moshkovitz et al. 2020; Frost, Moshkovitz, and Rashtchian 2020; Laber and Murtinho 2021; Makarychev and Shan 2021; Gamlath et al. 2021) makes the k-means and k-medians algorithm results interpretable by showing a small decision tree that partitions the input data into clusters. ExCut (Gad-Elrab et al. 2020) explains embedding-based clustering results over knowledge graphs by also performing



(a) $k = 5, \alpha = 0.45$



(b) $k = 6, \alpha = 0.75$

Figure 5: XClusters results on the Credit Card dataset for two (k, α) configurations that have similar low $D + \lambda N$ values. (a) and (b) have 1,703 and 1,477 nodes, respectively, and the visualizations demonstrate how D and N trade off.

rule mining. More recently, there is an emphasis in making the explaining decision tree shallow (Laber, Murtinho, and Oliveira 2021). However, these methods assume a fixed reference clustering (e.g., a k-means clustering result) and fit a decision tree to that clustering. In comparison, XClusters varies the reference clustering to possibly find more explainable decision trees. Another key difference is that XClusters uses accuracy features for clustering and explainability features for decision tree training although the two types of features may overlap.

The most relevant work to XClusters is an outlier removal method that removes outliers from clusters to make a decision tree train accurately on them (Bandyapadhyay et al. 2022). Given a set of clusters, the objective is to find outlier examples to remove and a decision tree that exactly explains the clusters without the outliers. Here the explainability is measured as how many examples need to be removed for a perfectly-accurate decision tree. In contrast, we do not assume that some data is incorrect and can be removed. In addition, we do not assume the clusters are given as an input, but optimize the clustering itself for better explainability. An interesting future work is to remove outliers and optimize clustering for explainability together.

Monotonic optimization has been studied extensively (Tuy 2000; Alizamir 2009; Cheon, Al-Khayyal, and Ahmed 2005; Tuy, Al-Khayyal, and Ahmed 2001; Tuy, Minoux, and Hoai-Phuong 2006; Tuy 2005) where its techniques have impacted signal processing applications (Hellings et al. 2012; Matthiesen et al. 2020). The popular algorithm are the Polyblock and branch-reduce-and-bound (BRB) algorithm where the latter is known to be faster in practice. Our problem also falls into monotonic optimization, and we specialize the BRB algorithm to our setting. We note that our key contribution is identifying the monotonicity properties in our problem setup, and utilizing them for faster tuning.

Dataset	Method	ϵ_b	$D + \lambda N$	Runtime (sec)
Credit Card	XClusters	0.01	$0.625_{\pm 0.000}$	$9.330_{\pm 0.019}$
		0.05	$0.656_{\pm 0.000}$	$6.336_{\pm 0.104}$
		0.10	$0.656_{\pm 0.000}$	$3.278_{\pm 0.017}$
		0.2	$0.656_{\pm 0.000}$	$2.119_{\pm 0.016}$
	BO	n/a	$0.657_{\pm 0.018}$	$16.596_{\pm 0.377}$
DS4C	XClusters	0.01	$0.547_{\pm 0.000}$	$0.166_{\pm 0.023}$
		0.05	$0.547_{\pm 0.000}$	$0.122_{\pm 0.001}$
		0.10	$0.547_{\pm 0.000}$	$0.081_{\pm 0.003}$
		0.2	$0.547_{\pm 0.000}$	$0.045_{\pm 0.002}$
	BO	n/a	$0.537_{\pm 0.078}$	$1.554_{\pm 0.140}$
Contracts	XClusters	0.01	$0.644_{\pm 0.000}$	$0.207_{\pm 0.033}$
		0.05	$0.644_{\pm 0.000}$	$0.197_{\pm 0.015}$
		0.10	$0.644_{\pm 0.000}$	$0.150_{\pm 0.013}$
		0.2	$0.710_{\pm 0.000}$	$0.065_{\pm 0.002}$
	BO	n/a	$0.650_{\pm 0.002}$	$7.265_{\pm 0.562}$

Table 3: Runtime comparison between XClusters and the BO baseline while varying ϵ_b on the three datasets.

Conclusion

To our knowledge, we are the first to propose an *explainable-first clustering* technique where explainability is also a primary objective of clustering. Given features for clustering and explaining, our XClusters framework minimizes the cluster distortion and the size of the decision tree trained to explain the clusters. We observed that our optimization is a difference of monotonic functions and proposed a branch-and-bound algorithm that efficiently finds the optimal number of clusters and distance function balancing ratio. We empirically showed on real datasets how XClusters outperforms baselines that explain after clustering or use black-box optimization without exploiting the monotonicity properties. A future work is to apply our techniques on various other types of data including images.

Acknowledgments

This work was supported by the National Research Foundation of Korea(NRF) grant funded by the Korea government(MSIT) (No. NRF-2022R1A2C2004382). We appreciate BC Card for providing the Credit Card dataset. Steven E. Whang is the corresponding author.

References

Adebayo, J.; Gilmer, J.; Muelly, M.; Goodfellow, I. J.; Hardt, M.; and Kim, B. 2018. Sanity Checks for Saliency Maps. In *NeurIPS*, 9525–9536.

Alizamir, S. 2009. Monotonic Optimization. In *Encyclopedia of Optimization, Second Edition*, 2316–2323.

Arrieta, A. B.; Díaz-Rodríguez, N.; Del Ser, J.; Bennetot, A.; Tabik, S.; Barbado, A.; García, S.; Gil-López, S.; Molina, D.; Benjamins, R.; et al. 2020. Explainable Artificial Intelligence (XAI): Concepts, taxonomies, opportunities and challenges toward responsible AI. *Information Fusion*, 58: 82–115.

Bandyapadhyay, S.; Fomin, F. V.; Golovach, P. A.; Lochet, W.; Purohit, N.; and Simonov, K. 2022. How to Find a Good Explanation for Clustering? In *AAAI*, 3904–3912.

Berndt, D. J.; and Clifford, J. 1994. Using Dynamic Time Warping to Find Patterns in Time Series. In *Knowledge Discovery in Databases: Papers from the 1994 AAAI Workshop, Seattle, Washington, USA, July 1994. Technical Report WS-94-03*, 359–370.

Cheon, M.-S.; Al-Khayyal, F.; and Ahmed, S. 2005. *Global Optimization of Monotonic Programs: Applications in Polynomial and Stochastic Programming*. Ph.D. thesis.

Doshi-Velez, F.; and Kim, B. 2017. Towards A Rigorous Science of Interpretable Machine Learning. arXiv:1702.08608.

Ellson, J.; Gansner, E. R.; Koutsofios, E.; North, S. C.; and Woodhull, G. 2001. Graphviz - Open Source Graph Drawing Tools. In *Graph Drawing*, 483–484.

Frost, N.; Moshkovitz, M.; and Rashtchian, C. 2020. ExKMC: Expanding Explainable k-Means Clustering. *CoRR*, abs/2006.02399.

Gad-Elrab, M. H.; Stepanova, D.; Tran, T.; Adel, H.; and Weikum, G. 2020. ExCut: Explainable Embedding-Based Clustering over Knowledge Graphs. In *ISWC*, volume 12506, 218–237.

Gamlath, B.; Jia, X.; Polak, A.; and Svensson, O. 2021. Nearly-Tight and Oblivious Algorithms for Explainable Clustering. In *NeurIPS*, 28929–28939.

Gupta, M. R.; Cotter, A.; Pfeifer, J.; Voevodski, K.; Canini, K. R.; Mangylov, A.; Moczydlowski, W.; and Esbroeck, A. V. 2016. Monotonic Calibrated Interpolated Look-Up Tables. *J. Mach. Learn. Res.*, 17: 109:1–109:47.

Hellings, C.; Joham, M.; Riemensberger, M.; and Utschick, W. 2012. Minimal Transmit Power in Parallel Vector Broadcast Channels With Linear Precoding. *IEEE Trans. Signal Process.*, 60(4): 1890–1898.

Hwang, H.; and Whang, S. E. 2022. XClusters: Explainability-first Clustering. Technical report. https://arxiv.org/abs/2209.10956.

Jain, A. K.; Murty, M. N.; and Flynn, P. J. 1999. Data Clustering: A Review. *ACM Comput. Surv.*, 31(3): 264–323.

Kaufman, L.; and Rousseeuw, P. J. 2008. *Partitioning Around Medoids (Program PAM)*, 68–125. ISBN 9780470316801.

Kim, J. 2020. DS4C: Data Science for COVID-19 in South Korea. https://www.kaggle.com/kimjihoo/coronavirusdataset. Accessed Aug. 15th, 2022.

Laber, E. S.; and Murtinho, L. 2021. On the price of explainability for some clustering problems. In *ICML*, 5915–5925.

Laber, E. S.; Murtinho, L.; and Oliveira, F. 2021. Shallow decision trees for explainable k-means clustering. *CoRR*, abs/2112.14718.

Linville, R. 2022. Master Contract Sales Data by Customer, Contract, Vendor. https://data.wa.gov/Procurements-and-Contracts/Master-Contract-Sales-Data-by-Customer-Contract-Ve/n8q6-4twj. Accessed Aug. 15th, 2022.

Lipton, Z. C. 2018. The Mythos of Model Interpretability: In machine learning, the concept of interpretability is both important and slippery. *Queue*, 16(3): 31–57.

Lou, Y.; Caruana, R.; and Gehrke, J. 2012. Intelligible models for classification and regression. In *KDD*, 150–158.

Makarychev, K.; and Shan, L. 2021. Near-Optimal Algorithms for Explainable k-Medians and k-Means. In *ICML*, 7358–7367.

Matthiesen, B.; Hellings, C.; Jorswieck, E. A.; and Utschick, W. 2020. Mixed Monotonic Programming for Fast Global Optimization. *IEEE Trans. Signal Process.*, 68: 2529–2544.

Mockus, J. 1974. On Bayesian Methods for Seeking the Extremum. In *Optimization Techniques, IFIP Technical Conference, Novosibirsk, USSR, July 1-7, 1974*, volume 27 of *Lecture Notes in Computer Science*, 400–404.

Molnar, C. 2019. Interpretable Machine Learning. https://christophm.github.io/interpretable-ml-book/. Accessed: 2022-10-25.

Moshkovitz, M.; Dasgupta, S.; Rashtchian, C.; and Frost, N. 2020. Explainable k-Means and k-Medians Clustering. In *ICML*, volume 119, 7055–7065.

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.

Ribeiro, M. T.; Singh, S.; and Guestrin, C. 2016. "Why Should I Trust You?": Explaining the Predictions of Any Classifier. In *SIGKDD*, 1135–1144.

Ribeiro, M. T.; Singh, S.; and Guestrin, C. 2018. Anchors: High-Precision Model-Agnostic Explanations. In *AAAI*, 1527–1535.

Singh, R.; Meduri, V. V.; Elmagarmid, A. K.; Madden, S.; Papotti, P.; Quiané-Ruiz, J.; Solar-Lezama, A.; and Tang, N. 2017. Synthesizing Entity Matching Rules by Examples. *Proc. VLDB Endow.*, 11(2): 189–202.

Tjoa, E.; and Guan, C. 2020. A Survey on Explainable Artificial Intelligence (XAI): Toward Medical XAI. *IEEE Transactions on Neural Networks and Learning Systems*, 1–21.

Tuy, H. 2000. Monotonic Optimization: Problems and Solution Approaches. *SIAM J. Optim.*, 11(2): 464–494.

Tuy, H. 2005. Polynomial optimization: a robust approach. *Pacific Journal of Optimization*, 1(357-374): 29.

Tuy, H.; Al-Khayyal, F.; and Ahmed, S. 2001. Polyblock algorithms revisited. *Preprint, Institute of Mathematics, Hanoi*.

Tuy, H.; Minoux, M.; and Hoai-Phuong, N. T. 2006. Discrete Monotonic Optimization with Application to a Discrete Location Problem. *SIAM J. Optim.*, 17(1): 78–97.

Vavasis, S. A. 1995. *Complexity Issues in Global Optimization: A Survey*, 27–41.

Wang, T.; Rudin, C.; Doshi-Velez, F.; Liu, Y.; Klampfl, E.; and MacNeille, P. 2017. A Bayesian Framework for Learning Rule Sets for Interpretable Classification. *J. Mach. Learn. Res.*, 18: 70:1–70:37.

You, S.; Ding, D.; Canini, K. R.; Pfeifer, J.; and Gupta, M. R. 2017. Deep Lattice Networks and Partial Monotonic Functions. In *NeurIPS*, 2981–2989.