Interpretable Clustering via Multi-Polytope Machines

Connor Lawless[1]*, Jayant Kalagnanam[2], Lam M. Nguyen[2], Dzung Phan[2], Chandra Reddy[2]

[1] Cornell Univeristy, Operations Research and Information Engineering, Ithaca, NY 14850
[2] IBM Research, Thomas J. Watson Research Center, Yorktown Heights, NY 10598, USA
cal379@cornell.edu, jayant@us.ibm.com, LamNguyen.MTLD@ibm.com, phandu@us.ibm.com, creddy@ibm.com

Abstract

Clustering is a popular unsupervised learning tool often used to discover groups within a larger population such as customer segments, or patient subtypes. However, despite its use as a tool for subgroup discovery and description - few state-of-the-art algorithms provide any rationale or description behind the clusters found. We propose a novel approach for interpretable clustering that both clusters data points and constructs polytopes around the discovered clusters to explain them. Our framework allows for additional constraints on the polytopes - including ensuring that the hyperplanes constructing the polytope are axis-parallel or sparse with integer coefficients. We formulate the problem of constructing clusters via polytopes as a Mixed-Integer Non-Linear Program (MINLP). To solve our formulation we propose a two phase approach where we first initialize clusters and polytopes using alternating minimization, and then use coordinate descent to boost clustering performance. We benchmark our approach on a suite of synthetic and real world clustering problems, where our algorithm outperforms state of the art interpretable and non-interpretable clustering algorithms.

1 Introduction

Clustering is an unsupervised machine learning problem that aims to partition unlabelled data into groups. In practice, it is often used as a tool for discovering sub-populations within a dataset such as customer segments, disease heterogeneity, or movie genres. In these applications the group assignment itself is often of secondary importance to the interpretation of the groups found. However, traditional clustering algorithms simply output a set of cluster assignments and provide no explanation or interpretation for the discovered groups. This fundamental misalignment between algorithms and applications force practitioners to work backwards from cluster assignments to fit post-hoc explanations to the output of traditional clustering algorithms. Motivated by these shortcomings, the field of interpretable clustering aims to jointly cluster points and provide an explanation of the groups themselves.

Recent work on interpretable clustering has focused on leveraging decision trees, a popular interpretable supervised learning tool, to construct clusters (Ghattas, Michel, and Boyer 2017; Moshkovitz et al. 2020; Frost, Moshkovitz, and Rashtchian 2020; Bertsimas and Dunn 2017; Fraiman, Ghattas, and Svarc 2013; Basak and Krishnapuram 2005; Jin and Chang 2001; De Raedt and Blockeel 1997; Yasami and Mozaffari 2010). While these approaches have intuitive appeal, as decision trees are a well studied model class that are easy to understand, they focus primarily on clusters that can be defined by axis-parallel partitions of the feature space. In contrast, interpretable supervised learning has a much wider range of tools available to practitioners including sparse linear models (Tibshirani 1996; Ustun, Traca, and Rudin 2013; Bertsimas and Van Parys 2020), rule sets (Dash, Günlük, and Wei 2018; Wang and Rudin 2015; Wang et al. 2017; Lawless and Gunluk 2021), or score cards (Ustun and Rudin 2017) amongst others. In this paper we introduce a novel method for interpretable clustering that provides cluster explanations by constructing polytopes around each cluster. Our framework allows for constraints on the hyperplanes that construct each polytope allowing for a wider range of cluster explanations including axis parallel partitions (similar to decision trees), partitions defined by sparse integer hyperplanes (similar to sparse/integer models), and general linear models (similar to SVMs). Figure 1 shows an example of a polytope cluster explanation under the three different constraints. Overall our approach provides practitioners more flexibility to explain clusters inline with the business requirements of their application.

1.1 Related Work

Existing interpretable clustering methods can be grouped into two general approaches: post-hoc explanations and integrated interpretation and clustering. Post-hoc approaches take the output of any clustering algorithm and attempt to fit an explanation to it. A common heuristic approach is to simply use an interpretable supervised learning algorithm to predict the cluster labels (Jain, Murty, and Flynn 1999; De Koninck, De Weerdt et al. 2017; Kauffmann et al. 2019). Recent work has also looked at designing specialized decision tree algorithms to explain the output of a clustering algorithm using a new splitting criterion (Moshkovitz et al. 2020; Frost, Moshkovitz, and Rashtchian 2020). Another post-hoc approach is to find a representative summary point, or prototype, for each cluster. Common choices in-

*Work done while an intern at IBM Research.

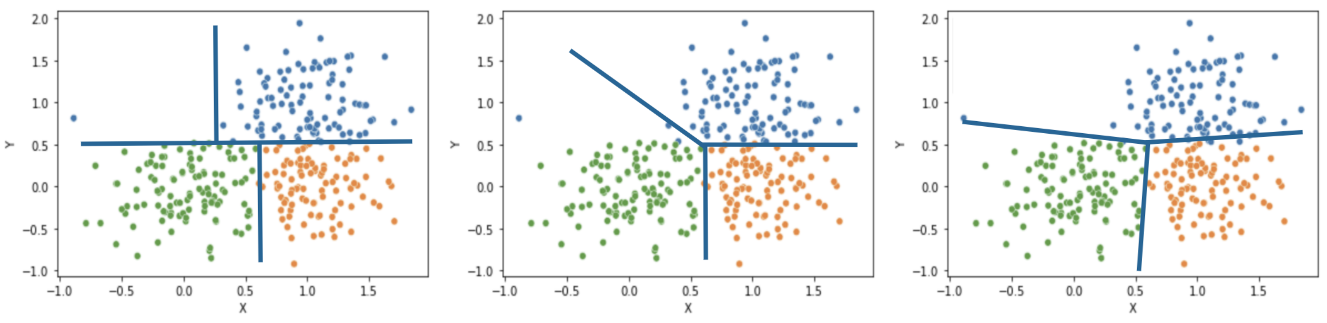

Figure 1: Sample clusters explained by polytopes with different constraints on the hyperplanes. (Left) Polytopes are composed of axis-parallel hyperplanes giving rise to rectangular cluster explanations. (Center) Polytopes are composed with integral hyperplanes allowing only diagonal or axis parallel lines. (Right) Polytopes composed with general hyperplanes.

clude looking at summary statistics for each feature such as the mean, median or mode. Carrisoza et al. present an integer programming (IP) formulation for finding an optimal prototype and radius around the point that captures the cluster, trading off false positives and false negatives (**?**). While these post-hoc approaches work with any clustering algorithm, they fail to incorporate the down-stream interpretation task in the generation of the clusters themselves.

In response to the shortcomings of post-hoc approaches, recent work has focused on integrated approaches that perform clustering with the interpretation task in mind. Liu et al. transform clustering into a supervised learning problem by augmenting the dataset with synthetic data points and use a decision tree algorithm to classify the original points from the synthetic, resulting in a decision tree with leaf nodes representing clusters (Liu, Xia, and Yu 2000). Researchers have also adapted existing heuristic decision tree approaches to perform clustering by using new splitting criterion (Fraiman, Ghattas, and Svarc 2013). Most similar to our work, Bertsimas et al. formulate the problem of finding an optimal decision tree to perform clustering as a mixed integer optimization problem and construct an approximate solution via coordinate descent (Bertsimas, Orfanoudaki, and Wiberg 2021). In a similar vein, recent work has looked at building rule sets to explain clusters (Chen et al. 2016; Chen 2018; Pelleg and Moore 2001). In contrast to the existing state of the art, our approach is the first to look at a more general function class to explain clusters. Our approach has more expressive power than decision-tree or rectangular approaches as polytopes with axis-parallel hyperplanes can be mapped to a decision tree. We also provide more flexibility to practitioners, allowing them to trade off the relative importance of interpretability and cluster quality.

1.2 Main Contributions

We summarize our main contributions as follows:

- We propose a novel mixed integer optimization nonlinear programming (MINLP) formulation for interpretable clustering that jointly clusters points and constructs polytopes surrounding each cluster.
- As a component of our MINLP framework we introduce a representation aware k-means clustering formulation that forms clusters with interpretability integrated into the objective.
- We present a formulation to find separating hyperplanes with sparse integer coefficients for interpretability.
- To approximate the solution of the MINLP formulation we introduce a two-stage optimization procedure that initializes clusters via alternating minimization and then optimizes the Silhouette coefficient via coordinate descent.
- Numerical experiments on both synthetic and real-world datasets show our approach outperforms state of the art interpretable and uninterpretable clustering algorithms.

The remainder of this paper is organized as follows. Section 2 details the MINLP formulation for polytope clustering. Section 3 gives an overview of our two-stage optimization procedure to approximate the optimal solution to our formulation. Finally, we present a suite of numerical experiments on synthetic and real world datasets in Section 4.

2 Mixed Integer Optimization Framework

In the standard clustering setting we are given a set of unlabelled data points $\mathcal{D} = \{x^t \in \mathbb{R}^D\}_{t=1}^N$ and asked to partition them into a set of K clusters $C_1, \ldots, C_K$, where C_i is the set of points belonging to cluster i. Note that we assume the data to have real-valued features and have an associated metric for defining pairwise distance between points. This is not a restrictive assumption as categorical data can be converted to real valued features via a standard one-hot encoding scheme. In the interpretable clustering setting we also have to provide an explanation of each cluster.

The key idea of our approach is to explain each cluster by constructing a polytope around it. To construct such a polytope we find a separating hyperplane for each pair of clusters. The intersection of the half-spaces generated by the set of hyperplanes involving each cluster then defines the polytope. In this section we present a mixed integer optimization formulation for jointly finding clusters and defining polytopes. We start by considering the interpretation and clustering problems separately before joining them in a unified framework.

2.1 Interpretable Separating Hyperplanes

Consider the setting where we have a fixed set of cluster assignments and need to construct a hyperplane to separate a pair of clusters C_i and C_j. To construct a polytope to describe cluster C_i we would need to construct one such hyperplane between C_i and every other cluster. This problem bares a striking resemblance to the classic support vector machine (SVM) problem (Boser, Guyon, and Vapnik 1992), where the goal is to find a hyperplane that separates two classes of data with a maximum margin. However, the standard SVM approach adds no additional constraints on the resulting hyperplane - potentially leading to dense hyperplanes with decimal values that are difficult for end users to interpret. We instead use a novel IP formulation that constructs a separating hyperplane with small integer coefficients and a limit on the number of non-zero coefficients.

Let w^{ij} and b^{ij} be the slope and intercept of the separating hyperplane between clusters i and j. Furthermore let $w^{ij}_{d,+}$ and $w^{ij}_{d,-}$ represent non-negative components of element d in w - namely $w^{ij}_d = w^{ij}_{d,+} - w^{ij}_{d,-}$. We also introduce a constant M that represents the maximum allowable integer coefficient value. To add constraints on the sparsity of the hyperplane we introduce binary variables $y^{ij}_{d,+}$ and $y^{ij}_{d,-}$ that track whether feature d is included in the hyperplane. We put a hard constraint of β on the number of non-zero coefficients in the final hyperplane. ξ^{ij}_t tracks the misclassification of data point t - specifically its distance from the correct side of the hyperplane to be classified correctly. Finally, ϵ is a fixed constant for the minimum non-zero separation distance with respect to any feasible hyperplane (i.e., the smallest non-zero distance between two points with respect to feasible hyperplane). With this notation in mind, the IP formulation for constructing an interpretable separating hyperplane is the following:

$$\min_{w,b,\xi} \quad \sum_{t \in C_i \cup C_j} (\xi^{ij})_t \tag{1}$$

$$\textbf{s.t.} \quad (w^{ij})^T x^t + b^{ij} \geq -(\xi^{ij})_t, \ \forall x^t \in C_i \tag{2}$$

$$(w^{ij})^T x^t + b^{ij} + \epsilon \leq (\xi^{ij})_t, \ \forall x^t \in C_j \tag{3}$$

$$w^{ij}_{d,+} - w^{ij}_{d,-} = w^{ij}_d \quad \forall d \in [D] \tag{4}$$

$$\sum_{d \in [D]} (w^{ij}_{d,+} + w^{ij}_{d,-}) \geq 1 \tag{5}$$

$$y^{ij}_{d,+} + y^{ij}_{d,-} \leq 1 \quad \forall d \in [D] \tag{6}$$

$$\sum_{d \in [D]} (y^{ij}_{d,+} + y^{ij}_{d,-}) \leq \beta \tag{7}$$

$$0 \leq w^{ij}_{d,+} \leq M y^{ij}_{d,+} \quad \forall d \in [D] \tag{8}$$

$$0 \leq w^{ij}_{d,-} \leq M y^{ij}_{d,-} \quad \forall d \in [D] \tag{9}$$

$$w^{ij} \in \mathcal{Z}^d, w^{ij}_+, w^{ij}_- \in \mathcal{Z}^d_{\geq 0} \tag{10}$$

$$(\xi^{ij})_t \geq 0 \tag{11}$$

$$y^{ij}_d \in \{0,1\}. \tag{12}$$

The objective (1) is simply to minimize the classification error of the separating hyperplane. Constraints (2) and (3) track whether we are classifying data point x^t correctly. Note that we add ϵ to constraint (3) to ensure that the separating hyperplane is only inclusive of cluster C_i. In other words, a data point in cluster C_j is only classified correctly if it lies strictly below the hyperplane. Constraints (5) and (6) ensure that the trivial hyperplane (all zero) is excluded by ensuring that the ℓ_1 norm of the hyperplane is above 1 (the smallest possible ℓ_1 norm of an integer hyperplane) and that only at most one of $w^{ij}_{d,+}$ and $w^{ij}_{d,-}$ are non-zero. Constraint (7) bounds the number of non-zero coefficients. Finally constraints (8) and (9) constrain the maximum integer values of the coefficients. One interpretation of the constant M is that it controls the search space of possible hyperplanes. Start by noting that any general hyperplane can be normalized to have coefficients between -1 and 1 by dividing by the largest coefficient. For integer hyperplanes with maximum value M normalizing by M gives possible coefficient values $\frac{n}{M}$ where n is an integer between $-M$ and M. Thus increasing M grows the number of feasible hyperplanes.

Note that when solving this problem we only consider data points in C_i and C_j, which in settings with a large number of clusters can be substantially less than the full data set. This allows our IP formulation to scale to larger datasets while limiting the computational burden of solving each individual IP.

2.2 Silhouette Clustering with Cardinality Constraints

Now consider the problem of finding cluster assignments. High quality clusters are generally defined by having low *intra*-cluster distance (i.e. the distance between points in the same cluster), and high *inter*-cluster distance (i.e. distance between points in different clusters). There are a number of cluster quality metrics that incorporate this high-level concept, one of the most popular being silhouette coefficient. The silhouette coefficient uses the average distance between a data point t and all other data in the same cluster as a measure of intra-cluster distance, and the average distance between t and every point in the second closest cluster as a measure of inter-cluster distance.

Definition 1 (Silhouette Coefficient) *Consider data point t with cluster label k. Let $r(t)$ be the average distance between data point t and every other point in the same cluster. Let $q(t)$ be the average distance between data point t and every point in the second closest cluster. The silhouette score for data point t is defined as:*

$$r(t) = \frac{1}{|C_k| - 1} \sum_{j \in C_k} d_{tj}$$

$$q(t) = \min_{l=1,\ldots,K: l \neq k} \frac{1}{|C_l|} \sum_{j \in C_l} d_{tj}$$

$$s(t) = \frac{q(t) - r(t)}{\max(q(t), r(t))}$$

The silhouette score for a set of cluster assignments is the *average of the silhouette scores for all the data points. The possible values range from -1 (worst) to +1 (best).*

Similar to (Bertsimas, Orfanoudaki, and Wiberg 2021) we now formulate the silhouette clustering problem as a MINLP. Let z_{tk} be the binary variable indicating whether data point t is assigned to cluster k, and the variables u_k be the binary variable indicating whether cluster k is used. To track the silhouette coefficient, let s_t be the silhouette score for data point t, q_t represent the intercluster distance measure $q(t)$, and r_t represent the intracluster distance measure $r(t)$. Let c_{tk} track the distance from data point t to cluster k, and γ_{tk} be the binary variable indicating the second closest cluster for data point t. Using this notation, the formulation for finding the cluster assignments is:

$$\min_{z,c_k,u_k} \quad -\frac{1}{n}\sum_{t\in\mathcal{D}} s_t \tag{13}$$

$$\textbf{s.t.} \quad \sum_{k=1}^{K} z_{tk} = 1 \ \forall t \in \mathcal{D} \tag{14}$$

$$N_{min}u_k \leq \sum_{t\in\mathcal{D}} z_{tk} \leq N_{max}u_k, \ \forall k = 1,\dots,K \tag{15}$$

$$s_t = \frac{q_t - r_t}{m_t} \quad \forall t \in \mathcal{D} \tag{16}$$

$$c_{tk} = \frac{1}{N_k}\sum_{j\in\mathcal{D}} d_{tj}z_{jk}, \ \forall t, k \tag{17}$$

$$N_k = \sum_{t\in\mathcal{D}} z_{tk} \quad \forall k \in [K] \tag{18}$$

$$r_t = \sum_k c_{tk}z_{tk} \quad \forall t \in \mathcal{D} \tag{19}$$

$$q_t \geq \sum_k \gamma_{tk}c_{tk} \quad \forall t \in \mathcal{D} \tag{20}$$

$$\sum_k \gamma_{tk} = 1 \quad \forall t \in \mathcal{D} \tag{21}$$

$$\gamma_{tk} \leq 1 - z_{tk} \quad \forall t \in \mathcal{D} \tag{22}$$

$$m_t \geq r_t, q_t \tag{23}$$

$$u_k, z_k \in \{0,1\}. \tag{24}$$

The objective of the formulation is to maximize the silhouette coefficient. Constraint (14) ensures that we assign each data point to at most one cluster. Constraint (15) sets cardinality constraints (i.e., minimum and maximum values: N_{min} and N_{max}) on the size of the clusters. Finally, constraints (16)-(23) track the silhouette coefficient for each data point t. Note that this formulation is a mixed integer non-linear program with non-linearity introduced by the silhouette coefficient.

2.3 Joint Optimization Framework

The goal of the joint framework for clustering and polytope construction is to both maximize cluster quality, defined by the silhouette coefficient, and minimize the representation error. To control the relative importance given to representation error and cluster quality we also introduce a regularization parameter λ. We define the representation error as the sum of the mis-classification costs for all the hyperplanes defining each cluster's polytope. Given a hyperplane between two clusters C_i, C_j, we can determine the hypothetical mis-classification error for every data point (including those not in C_i and C_j) if they were assigned to C_i or C_j by computing the distance from the point to the hyperplane. Let $(\xi_+^{ij})_t$ and $(\xi_-^{ij})_t$ be the error for data point t if it is assigned to cluster i and j respectively. Let $\mathcal{K} \times \mathcal{K}$ be the set of all pairs of clusters. Combining the hyperplane and clustering formulations, the joint framework for constructing polytope clusters is:

$$\min_{z,c_k,u_k} -\frac{1}{n}\sum_{t\in\mathcal{D}} s_t \ + \tag{25}$$

$$\lambda \sum_{x^t\in\mathcal{D}} \sum_{i=1}^{K-1} \sum_{j=i+1}^{K} (z_{tk}(\xi_+^{ij})_t + z_{tj}(\xi_-^{ij})_t) \tag{26}$$

$$\textbf{s.t. (14)-(24)} \tag{27}$$

$$\textbf{(4)-(12) for } i,j \in \mathcal{K}\times\mathcal{K} \tag{28}$$

$$(w^{ij})^T x^t + b^{ij} \geq -(\xi_+^{ij})_t \ \ \forall i,j \in \mathcal{K}\times\mathcal{K}, t \in \mathcal{D} \tag{29}$$

$$(w^{ij})^T x^t + b^{ij} + \epsilon \leq (\xi_-^{ij})_t \ \ \forall i,j \in \mathcal{K}\times\mathcal{K}, t \in \mathcal{D} \tag{30}$$

Note that the objective now trades off cluster quality, i.e., Eq. (25), and representation error, i.e., Eq. (26). We also need one separating hyperplane problem per pair of clusters.

Remark 1 *There is no requirement that the feature space for clustering and hyperplane separation be the same. For instance real valued features can be used to compute the cluster quality, but the separation is done in a binarized feature space to ensure more interpretable explanations.*

3 Algorithm Overview

The joint formulation for polytope clustering is a MILNP, and thus is difficult to optimize globally. Instead, we use a two-stage procedure to find a high quality approximation of the solution. In the first stage we use alternating minimization to find an initial set of clusters and separating hyperplanes. We then use coordinate descent to improve the clustering performance of our assignments and explanation.

3.1 Initialization via Alternating Minimization

We decompose the joint framework (25)-(30) into two components: cluster assignments, and constructing hyperplanes to separate clusters. The key intuition behind our initialization procedure is we alternate between clustering the points into K clusters $C_1, C_2, \dots C_K$ and then constructing interpretable separating hyperplanes between each pair of clusters. The silhouette clustering problem as presented in Section 2.2 remains a difficult problem to solve due to the non-linearity presented by the silhouette metric. For computational tractability, we instead use the k-means clustering objective (Jain, Murty, and Flynn 1999) as a proxy for silhouette coefficient during the initialization phase. This leads to the following much simpler formulation for determining cluster assignments:

$$\min_{z,c_k,u_k} \sum_{k=1}^{K}\sum_{x^t\in\mathcal{D}} z_{tk}\|x^t - c_k\|^2 + \lambda \sum_{x^t\in\mathcal{D}}\sum_{i=1}^{K-1}\sum_{j=i+1}^{K}\left(z_{tk}(\xi_+^{ij})_t + z_{tj}(\xi_-^{ij})_t\right) \quad (31)$$

$$\textbf{s.t. (14)-(15)} \quad (32)$$

Note that this problem is now similar to a traditional k-means clustering problem, with an additional objective term to capture representation error. We denote this new clustering formulation the representation aware k-means clustering problem. To solve this formulation for a fixed set of representation errors ξ we alternate between fixing the cluster centers c_k and generating assignments by solving the IP, and then fix the assignments and update the cluster centers and repeat the process until convergence. To begin the initialization procedure we solve the cluster assignment with no representation errors (i.e., $\xi_t = 0\ \forall t$), and then generate polytopes by solving the separating hyperplane problem detailed in Section 2.1 for every pair of clusters. The representation errors generated by the separating hyperplane problems are then used in the representation aware k-means problem and the process is repeated until convergence. The entire initialization procedure is outlined in Algorithm 1.

The choice of k-means as a proxy for silhouette coefficient is motivated both by its relative ease of optimization and the fact that k-means generally performs well as a proxy for silhouette coefficient (see experimental results in (Bertsimas, Orfanoudaki, and Wiberg 2021)). In the absence of interpretability constraints on the separating hyperplanes, all local solutions of the k-means clustering problem also have the appealing property that they can be perfectly explained by a polytope.

Theorem 1 (k-means Polytope Interpretability) *Local solutions to the k-means clustering problem with Euclidean distance can be perfectly separated from the other clusters by a polytope.*

Unfortunately this result does not hold with the addition of interpretability constraints. However the following theorem shows that our initialization procedure is still guaranteed to converge to a solution in a finite number of iterations.

Theorem 2 (Alternating Minimization Improvement) *Algorithm 1 generates objective values for the representation aware k-means clustering problem that are monotonically decreasing for $l \geq 2$, and terminates in a finite number of iterations.*

3.2 Coordinate Descent

Once an initial clustering and polytope explanation are in place we use a local search procedure to optimize the original clustering objective. The key idea behind the local search is we consider each polytope and try to adjust it to boost clustering performance. We consider the following local search operations:

Algorithm 1: Cluster Initialization via Alternating Minimization

Inputs: Data $\mathcal{D}$, initial number of clusters K, $\lambda \geq 0$, $M \in \mathbb{Z} > 1, \beta \in \mathbb{Z} > 1, \epsilon > 0$
Output: Cluster assignments $z \in \{0,1\}^{n\times K}$, separating hyperplanes w, b

```
1: Set $(\xi_+^{ij})_t = (\xi_-^{ij})_t = 0\ \forall i,j,t$
2: Initialize $z_{tk}, c_k$ using conventional $k$-means algorithm on $\mathcal{D}$
3: for $l = 1, 2, \ldots$ do
4:     /* Compute Cluster Assignments */
5:     for $m = 1, 2, \ldots$ do
6:         Fix $c_k$. Solve (31)-(32) for updated $z_{tk}$
7:         Set $c_k = \frac{1}{N_k}\sum_{t\in\mathcal{D}} z_{tk}x^t$, $N_k = \sum_{t\in\mathcal{D}} z_{tk}$.
8:     end for
9:     Set $C_i = \{x^t \in \mathcal{D} : z_{ti} = 1\}\ \forall i = 1,\ldots,K$
10:    /* Compute Separating Hyperplanes */
11:    for i=1,...,K-1 do
12:        for j=i+1,...,K do
13:            Solve (1)-(12) with $M, \beta, C_i, C_j$ for $w^{ij}, b^{ij}$
14:            Set $(\xi_+^{ij})_t = \max(-w^{ij}x^t + b, 0)\ \forall t \in \mathcal{D}$
15:            Set $(\xi_-^{ij})_t = \max(w^{ij}x^t + b + \epsilon, 0)\ \forall t \in \mathcal{D}$
16:        end for
17:    end for
18: end for
19: return z, w, b
```

- **Boundary Shift**: For a given hyperplane we alter the slope and the intercept of the hyperplane and change cluster assignments based on the new boundary. Intuitively this can be thought of as shifting one boundary of the defining polytope for a cluster. Here M and α restrict the search space of potential hyperplanes to consider, making an exhaustive consideration of potential hyperplanes feasible for small M and α.
- **Cluster Splitting**: For a given cluster we attempt to add a new hyperplane to split the cluster into two smaller clusters. Here we consider any feasible separating hyperplane (i.e., in accordance with M and α).
- **Cluster Merging**: For two adjacent clusters (i.e., two clusters separated only by one hyperplane), we attempt to remove that hyperplane and merge the clusters.

We consider each local search operation and retain the cluster assignment with the best objective value. One of the properties of this coordinate descent approach is that it can increase or decrease the number of clusters present in the assignment through merging or splitting clusters. This allows our approach to be less sensitive to the initial number of clusters specified during the initialization procedure. To provide a fair comparison to algorithms that have a fixed number of clusters and to support applications where there is a constraint on the number of clusters desired, our coordinate descent procedure also puts an upper bound on the total number of possible clusters that can be generated. The entire clustering algorithm including coordinate descent is outlined in Algorithm 2.

Algorithm 2: Multi-Polytope Clustering (MPC) Algorithm

Input: Data $\mathcal{D}$, initial number of clusters K, maximum cluster number $K_{\max}$, $\lambda \geq 0$, $M \in \mathbb{Z} > 1$, $\beta \in \mathbb{Z} > 1$
Output: Cluster assignments $z \in \{0,1\}^{n \times K}$, separating hyperplanes w, b

```
1: Initialize $z, w, b$ using Algorithm 1
2: Initialize processing queue $\mathcal{Q}$ with all hyperplane indices ($(i,j)$ $\forall i = 1,\dots,K-1, j = i+1,\dots,K$) and cluster indices ($i = 1,\dots,K$)
3: Compute current loss $\ell$ using (31) with cluster assignment $z$
4: while $\mathcal{Q}$ is not empty do
5:     for $q \in \mathcal{Q}$ do
6:         if $q$ corresponds to a hyperplane $(i,j)$ then
7:             Find best new hyperplane between $i, j$
8:         else if $q$ corresponds to cluster $C_i$ then
9:             Find best split for cluster $C_i$
10:        end if
11:        Compute updated loss $\ell'$ using (31)
12:        if $\ell' < \ell$ then
13:            Update $z, \ell = \ell', w, b$
14:            Reset $\mathcal{Q}$ with all hyperplanes and clusters
15:        end if
16:    end for
17: end while
18: return $z, w, b$
```

Remark 2 *Our optimization procedure only uses the choice of clustering metric (i.e., silhouette coefficient) during coordinate descent. This means that our framework can be easily extended to other clustering metrics such as Dunn Index.*

4 Numerical Results

To showcase the performance of the MPC algorithm we present two results: (1) clustering performance comparison to state of the art clustering algorithms on synthetic and real world datasets, (2) a view of sample cluster explanations under different settings of our hyperparameters.

4.1 Clustering Performance

We ran the MPC algorithm under two sets of hyperparameters to represent different levels of interpretability. The first, MPC-1, sets $M = \beta = 1$ and represents cluster explanations with only axis-parallel hyperplanes, providing a fair comparison to univariate decision tree based methods. The second, MPC-2, sets $M = 3, \beta = 2$ which allows for more general hyperplanes with up to two non-zero integer coefficients and coefficients within $[-3, 3]$. To benchmark the performance of MPC we compared it to the following suite of traditional and interpretable clustering algorithms: k-means++ (Arthur and Vassilvitskii 2006), Gaussian Mixture Models (GMM) (Hastie, Tibshirani, and Friedman 2009), Hierarchical Clustering (HClust) (Hastie, Tibshirani, and Friedman 2009), Density-based Spatial Clustering of Applications with Noise (DBSCAN) (Ester et al. 1996), Interpretable Clustering via Optimal Decision Trees (ICOT) (Bertsimas, Orfanoudaki, and Wiberg 2021), and Explainable k-means Clustering (ExKMC) (Frost, Moshkovitz, and Rashtchian 2020). For the ExKMC algorithm we present two sets of results: one where we only allow 1 leaf node per cluster (ExKMC-1) and one with up to five leaf nodes per cluster (ExKMC-5). See the supplementary materials for a detailed description of all the benchmark algorithms and the implementation we used.

For all algorithms that require a specification of the number of clusters we tune k between 2 and 10. We ran each algorithm 100 times with different random seeds and report the best result. We tested our algorithm on two sets of clustering datasets. The first is a set of 9 synthetic clustering instances called the fundamental clustering and projection suite (Ultsch and Lötsch 2020). The suite contains a range of problem sizes (212-4096 data points) with two or three real valued features. For every dataset we normalize all feature values to be between 0 and 1. A summary of the silhouette score for each algorithm on the synthetic instances is included in Table 1. The MPC-2 algorithm is able to dominate existing clustering methods with respect to the silhouette coefficient, obtaining the best silhouette coefficient on all 9 datasets. Increasing interpretability, namely using MPC-1 instead of MPC-2, comes at a cost to performance with lower scores in a third of the datasets. However, MPC-1 is still able to match or outperform other interpretable algorithms (ICOT, ExKMC-1, ExKMC-5) on all datasets.

To test the performance of MPC on a more realistic suite of clustering problems we present results for a set of 8 datasets from the UCI machine learning repository. While the number of data points remains similar to the synthetic instances (150-4601 data points), these datasets have larger feature spaces (4 to 89 dimensions). These datasets also have a mix of data types including numeric and categorical features. For numeric features we normalized all values to be between 0 and 1, and for categorical features we apply a one hot encoding. A summary of the results on the UCI datasets is included in Table 2. Unlike the synthetic datasets, increasing the cardinality of the hyperplanes (i.e. using MPC-2 instead of MPC-1) has no impact on the performance of the algorithm. Both algorithms match or outperform the benchmark algorithms on all but one dataset (Framingham), where ICOT achieves the highest score. Additional experimental results including computation time and the optimal number of clusters for each approach are included in the supplementary materials.

4.2 Interpretability

The output of Algorithm 2 is a set of cluster assignments and hyperplanes between clusters. To construct an explanation for a given cluster we include all hyperplanes related to the cluster to construct a polytope. We remove redundant hyperplanes (i.e., ones weaker than other constraints already included in the polytope). The flexibility of the MPC framework allows the resulting polytope explanation to resemble a number of different model classes. If we restrict the polytopes to only include axis-parallel hyperplanes (i.e., $M = \beta = 1$), then each cluster example is a rule. A sample cluster explanation for the zoo dataset is:

$$\text{(HAS HAIR) AND (HAS MILK) AND (LEGS} > 0)$$

Dataset	(n,d)	k-means++	GMM	HClust	DBSCAN	ICOT	ExKMC-1	ExKMC-5	**MPC-1**	**MPC-2**
Atom	(800,2)	0.615	0.613	0.601	0.525	0.508	0.584	0.609	0.601	0.617*
Chainlink	(1000,2)	0.517	0.524	0.504	0.244	0.396	0.508	0.516	0.518	0.519*
Engytime	(4096,2)	0.438	0.422	0.410	0.439	0.573*	0.408	0.429	0.573*	0.573*
Hepta	(212, 3)	0.702*	0.702*	0.702*	0.702*	0.455	0.702*	0.702*	0.702*	0.702*
Lsun	(400,2)	0.558	0.549	0.535	0.530	0.562	0.557	0.558	0.568*	0.568*
Target	(770,2)	0.592	0.575	0.580	0.416	0.629*	0.584	0.592	0.629*	0.629*
Tetra	(400,3)	0.506*	0.506*	0.495	0.506*	0.504	0.506*	0.506*	0.506*	0.506*
Two Diamonds	(800,2)	0.631*	0.631*	0.631*	0.631*	0.486	0.631*	0.631*	0.631*	0.631*
Wingnut	(1070,2)	0.460*	0.460*	0.435	0.117	0.422	0.448	0.452	0.450	0.460*

Table 1: Silhouette score on fundamental clustering and projection suite. Asterisk indicates best performing algorithm on each data set.

Dataset	(n,d)	k-means++	GMM	HClust	DBSCAN	ICOT	ExKMC-1	ExKMC-5	**MPC-1**	**MPC-2**
Iris	(150,4)	0.629*	0.629*	0.629*	0.629*	0.629*	0.629*	0.629*	0.629*	0.629*
Wine	(178,12)	0.381	0.386*	0.386	0.357	0.386*	0.357	0.381	0.386*	0.386*
Zoo	(101, 16)	0.409	0.395	0.416*	0.405	0.416*	0.396	0.409	0.416*	0.416*
Seeds	(210, 6)	0.505	0.478	0.493	0.165	0.243	0.500	0.505	0.506*	0.506*
Libras	(360, 89)	0.245*	0.227	0.230	0.164	0.154	0.182	0.228	0.245*	0.245*
Framingham	(3658, 14)	0.405	0.371	0.380	0.189	0.454*	0.403	0.405	0.406	0.406
Bank	(4521, 50)	0.124	0.079	0.120	0.046	0.113	0.122	0.124	0.125*	0.125*
Spam	(4601, 56)	0.677	0.524	0.602	0.546	0.760*	0.129	0.677	0.760*	0.760*

Table 2: Silhouette score on UCI Machine Learning Clustering problem sets. Asterisk indicates best performing algorithm on each data set.

Each hyperplane corresponds to a single clause in the resulting rule. Note that this rule could be equivalent to one leaf node in a decision tree, however the unordered conditions in a rule have been shown in a user study to be easier to interpret than a decision tree (Lakkaraju, Bach, and Leskovec 2016).

If we increase β, each explanation resembles a rule set where each hyperplane constitutes a rule. The zoo datasets contains primarily boolean features meaning that each hyperplane corresponds to a scorecard (Ustun and Rudin 2017) where each condition has an associated weight which is compared against a threshold. One cluster explanation for the zoo dataset with $\beta = 2, M = 3$ is:

[(DOMESTIC) + (HAS EGGS) $>$ 1] AND (HAS TEETH)

In this example the first rule in the rule set is a scorecard (both conditions need to be met). The rule sets require more effort to understand than a single rule, but still provide a clear explanation that can be understood by practitioners. For datasets with non-binerized features, the explanations remain rule sets but each rule is a more general linear condition. One cluster explanation for the wine dataset with $\beta = 2, M = 3$ is:

[3*(CITRIC ACIDITY) + 9*(DENSITY) $<$ 1.42]
AND
[1*(pH LEVEL) + 2*(CHLORIDES) $\geq$ 0.58]

Note that while the explanation is no longer easy to understand intuitively, it still remains audit-able which is sufficient for a number of applications. To improve the interpretability of explanations for datasets with real valued features, users can use binarized features for polytope construction (see Remark 1). Another benefit of the MPC framework is that it provides pairwise comparisons between clusters. To compare clusters with a decision tree, users have to traverse a tree looking at different nodes which can be further complicated by tree-based explanations with multiple leaf nodes per cluster. In contrast, the hyperplane separating each pair of clusters acts as a pairwise comparison. A pairwise comparison between two clusters in the zoo dataset (for $M = \beta = 1$) is:

IF (HAS HAIR) : Cluster 3 ELSE Cluster 4

5 Conclusion

In this paper we introduce a novel algorithm for interpretable clustering that describes clusters using polytopes. We formulate the problem of jointly clustering and explaining clusters as a MINLP that optimizes both silhouette coefficient and representation error. A novel IP formulation for finding separating hyperplanes is used to enforce interpretability considerations on the resulting polytopes. To approximate a solution to the MINLP we leverage a two-phase optimization approach that first generates an initial set of clusters and polytopes using alternating minimization, then improves clustering performance using coordinate descent. Compared to state of the art uninterpretable and interpretable clustering algorithms our approach is able to find high quality clusters while preserving interpretability.

References

Arthur, D.; and Vassilvitskii, S. 2006. k-means++: The advantages of careful seeding. Technical report, Stanford.

Basak, J.; and Krishnapuram, R. 2005. Interpretable hierarchical clustering by constructing an unsupervised decision tree. *IEEE transactions on knowledge and data engineering*, 17(1): 121–132.

Bertsimas, D.; and Dunn, J. 2017. Optimal classification trees. *Machine Learning*, 106(7): 1039–1082.

Bertsimas, D.; Orfanoudaki, A.; and Wiberg, H. 2021. Interpretable clustering: an optimization approach. *Machine Learning*, 110(1): 89–138.

Bertsimas, D.; and Van Parys, B. 2020. Sparse high-dimensional regression: Exact scalable algorithms and phase transitions. *The Annals of Statistics*, 48(1): 300–323.

Boser, B. E.; Guyon, I. M.; and Vapnik, V. N. 1992. A training algorithm for optimal margin classifiers. In *Proceedings of the fifth annual workshop on Computational learning theory*, 144–152.

Chen, J. 2018. *Interpretable Clustering Methods*. Ph.D. thesis, Northeastern University.

Chen, J.; Chang, Y.; Hobbs, B.; Castaldi, P.; Cho, M.; Silverman, E.; and Dy, J. 2016. Interpretable clustering via discriminative rectangle mixture model. In *2016 IEEE 16th International Conference on Data Mining (ICDM)*, 823–828. IEEE.

Dash, S.; Günlük, O.; and Wei, D. 2018. Boolean decision rules via column generation. *arXiv preprint arXiv:1805.09901*.

De Koninck, P.; De Weerdt, J.; et al. 2017. Explaining clusterings of process instances. *Data mining and knowledge discovery*, 31(3): 774–808.

De Raedt, L.; and Blockeel, H. 1997. Using logical decision trees for clustering. In *International Conference on Inductive Logic Programming*, 133–140. Springer.

Ester, M.; Kriegel, H.-P.; Sander, J.; Xu, X.; et al. 1996. A density-based algorithm for discovering clusters in large spatial databases with noise. In *kdd*, volume 96, 226–231.

Fraiman, R.; Ghattas, B.; and Svarc, M. 2013. Interpretable clustering using unsupervised binary trees. *Advances in Data Analysis and Classification*, 7(2): 125–145.

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

Ghattas, B.; Michel, P.; and Boyer, L. 2017. Clustering nominal data using unsupervised binary decision trees: Comparisons with the state of the art methods. *Pattern Recognition*, 67: 177–185.

Hastie, T.; Tibshirani, R.; and Friedman, J. 2009. Unsupervised learning. In *The elements of statistical learning*, 485–585. Springer.

Jain, A. K.; Murty, M. N.; and Flynn, P. J. 1999. Data clustering: a review. *ACM computing surveys (CSUR)*, 31(3): 264–323.

Jin, D.-S.; and Chang, J.-W. 2001. A Cell-based Clustering Method for Large High-dimensional Data in Data Mining. *Journal of KIISE: Databases*, 28(4): 558–567.

Kauffmann, J.; Esders, M.; Montavon, G.; Samek, W.; and Müller, K.-R. 2019. From clustering to cluster explanations via neural networks. *arXiv preprint arXiv:1906.07633*.

Lakkaraju, H.; Bach, S. H.; and Leskovec, J. 2016. Interpretable decision sets: A joint framework for description and prediction. In *Proceedings of the 22nd ACM SIGKDD international conference on knowledge discovery and data mining*, 1675–1684.

Lawless, C.; and Gunluk, O. 2021. Fair Decision Rules for Binary Classification. *arXiv preprint arXiv:2107.01325*.

Liu, B.; Xia, Y.; and Yu, P. 2000. Clustering Through Decision Tree Construction. *In Proceedings of the ACM International Conference on Information and Knowledge Management*.

Moshkovitz, M.; Dasgupta, S.; Rashtchian, C.; and Frost, N. 2020. Explainable k-Means and k-Medians Clustering. In III, H. D.; and Singh, A., eds., *Proceedings of the 37th International Conference on Machine Learning*, volume 119 of *Proceedings of Machine Learning Research*, 7055–7065. PMLR.

Pelleg, D.; and Moore, A. 2001. Mixtures of rectangles: Interpretable soft clustering. In *ICML*, volume 2001, 401–408.

Tibshirani, R. 1996. Regression shrinkage and selection via the lasso. *Journal of the Royal Statistical Society: Series B (Methodological)*, 58(1): 267–288.

Ultsch, A.; and Lötsch, J. 2020. The fundamental clustering and projection Suite (FCPS): A dataset collection to test the performance of clustering and data projection algorithms. *Data*, 5(1): 13.

Ustun, B.; and Rudin, C. 2017. Optimized risk scores. In *Proceedings of the 23rd ACM SIGKDD international conference on knowledge discovery and data mining*, 1125–1134.

Ustun, B.; Traca, S.; and Rudin, C. 2013. Supersparse linear integer models for interpretable classification. *arXiv preprint arXiv:1306.6677*.

Wang, T.; and Rudin, C. 2015. Learning optimized Or's of And's. *arXiv preprint arXiv:1511.02210*.

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. *The Journal of Machine Learning Research*, 18(1): 2357–2393.

Yasami, Y.; and Mozaffari, S. P. 2010. 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.