Optimal Interpretable Clustering Using Oblique Decision Trees

Magzhan Gabidolla
mgabidolla@ucmerced.edu
Dept. Computer Science & Engineering
University of California, Merced
Merced, CA, United States

Miguel Á. Carreira-Perpiñán
mcarreira-perpinan@ucmerced.edu
Dept. Computer Science & Engineering
University of California, Merced
Merced, CA, United States

ABSTRACT

Recent years have seen a renewed interest in interpretable machine learning, which seeks insight into how a model achieves a prediction. Here, we focus on the relatively unexplored case of interpretable clustering. In our approach, the cluster assignments of the training instances are constrained to be the output of a decision tree. This has two advantages: 1) it makes it possible to understand globally how an instance is mapped to a cluster, in particular to see which features are used for which cluster; 2) it forces the clusters to respect a hierarchical structure while optimizing the original clustering objective function. Rather than the traditional axis-aligned trees, we use sparse oblique trees, which have far more modelling power, particularly with high-dimensional data, while remaining interpretable. Our approach applies to any clustering method which is defined by optimizing a cost function and we demonstrate it with two k-means variants.

CCS CONCEPTS

• **Computing methodologies → Cluster analysis**.

KEYWORDS

clustering, interpretability, decision trees, oblique decision trees

ACM Reference Format:
Magzhan Gabidolla and Miguel Á. Carreira-Perpiñán. 2022. Optimal Interpretable Clustering Using Oblique Decision Trees. In *Proceedings of the 28th ACM SIGKDD Conference on Knowledge Discovery and Data Mining (KDD '22), August 14–18, 2022, Washington, DC, USA.* ACM, New York, NY, USA, 11 pages. https://doi.org/10.1145/3534678.3539361

1 INTRODUCTION

The desire to understand the internal working of a predictive model is very old in statistics and machine learning (ML), but has achieved enormous prominence in recent years. This is due to the widespread deployment of ML in practical applications; to the black-box nature of the more accurate models (such as neural networks and random or boosted forests); and to upcoming regulations in many jurisdictions that require model or algorithmic decisions to be explainable in some way, so they can be trusted or audited (for bias, fairness, mistakes, etc.).

The vast majority of work on interpretable models has focused on classification and, to a less extent, regression. There, the model takes the form of a predictive mapping (of the label for an input instance), and multiple approaches exist that seek to interpret the mapping, globally or locally around a specific instance. Here, we focus on clustering, which has received far less attention. At a basic level, the output of a clustering method trained on a dataset is the set of clusters, i.e., an assignment variable for each training instance that indicates which cluster it belongs to (we do not consider soft clustering, where the assignment is probabilistic). However, we can consider that a clustering method also outputs an out-of-sample mapping that predicts for any instance (not just those in the training set) the cluster it belongs to. This is effectively a classifier, which we can then interpret, for example to see whether some input features are not selected, or which features affect which clusters, or whether the clusters exhibit a hierarchical structure. One simple way to obtain such a mapping is to fit a classifier to the clustering result, but this is suboptimal in that the clustering result and the explainer mapping should be learned jointly.

Our approach starts by defining a joint learning problem of the clustering and the explainer mapping. We provide an algorithm for that which alternates two intuitive steps, one a regularized clustering and the other a classification. We do this in an agnostic way with respect to the original clustering method; all we require is that it define a cost function over the assignments. However, for the classifier, we use an oblique tree: this strikes a good balance of interpretability and accuracy, and introduces a hierarchical structure in the clustering. This provides a principled way to turn a flat clustering algorithm such as k-means into a hierarchical clustering algorithm that is still driven by the original clustering cost function.

Next, we first discuss what it means for a clustering method to be interpretable and argue for our approach (section 3). We discuss the special case of k-means (section 4). Then, we describe the class of clustering methods we handle (section 5), define our interpretable clustering formulation (section 6) and give an algorithm to optimize it (section 7). This critically relies on a recent algorithm to train sparse oblique trees, Tree Alternating Optimization (TAO), which we review in section 8. Finally, in section 9 we demonstrate our approach on two variants of k-means and on datasets of different type (tabular, images, documents) and dimensionality.

2 RELATED WORK

We briefly review decision tree based methods for interpretable clustering. Subspace clustering techniques, which aim to find clusters in low-dimensional subspaces, can also provide interpretable results (such as those based on feature selection); we refer the reader to [20, 28] for a detailed review of this work and its applications.

The traditional, widely established approach to decision tree learning in supervised problems, as in CART [3] and C5.0 [29], is based on greedy top-down induction: by starting with a root node we greedily split the nodes based on some purity criterion until a stopping condition is met, and optionally we might prune the tree based on a validation metric. Several lines of work adapt greedy recursive partitioning into the domain of clustering [1, 13, 16, 23]. Based on clever heuristics, these papers define new measures of splitting criteria for clustering such as heterogeneity of the data based on covariance or density, and they also propose corresponding modifications to the pruning or merging step. Ref. [2] provides a mixed-integer optimization formulation of a clustering problem over an axis-aligned decision tree with one leaf per cluster for two unusual objective functions (the silhouette metric and the Dunn index). This approach is limited to very small datasets and trees. Recently, ref. [26] provides theoretical approximation bounds over the optimal k-means and k-medians clustering for an axis-aligned decision tree with one leaf per cluster, and proposes a greedy recursive partitioning algorithm to build the tree from the k-means result. Subsequent works [9, 15, 21, 24] further extend and improve these approximation bounds. All these papers consider axis-aligned trees with K leaves to map the K clusters, which is too restrictive a model. Indeed, their experiments are limited to small problems. Ref. [14] gives a top-down tree induction algorithm to build a tree with an arbitrary number of leaves to map the k-means clustering.

All the aforementioned works consider only axis-aligned trees, in which a decision node performs the split by thresholding a single feature. Oblique trees, which use a linear combination of features at a decision node, are much more flexible models than axis-aligned ones. To the best of our knowledge, there are no existing works that use (sparse) oblique trees in interpretable clustering.

3 WHAT IS AN "INTERPRETABLE" CLUSTERING?

In this section we discuss what we mean by interpretable or explainable clustering, and justify our approach. Throughout, our work applies generally to any clustering method that defines a cost function of the cluster assignments; an example which is useful to keep in mind is k-means.

Interpretability of the clustering out-of-sample mapping. Firstly, clustering is an exploratory data analysis technique, so it is itself used to gain insights about a dataset regarding the existence of clusters. The clustering interpretability or explainability we seek is different. It aims at explaining how an input instance $\mathbf{x} \in \mathbb{R}^D$ (not necessarily in the training set) is mapped or assigned to a particular cluster. We call this the out-of-sample mapping. For some clustering methods, an out-of-sample mapping is naturally implicit. For example, for k-means it is given by assigning the instance $\mathbf{x}$ to its closest centroid (which involves computing the distance of $\mathbf{x}$ to each centroid). However, this mapping is not very helpful in explaining how the input features in $\mathbf{x}$ determine the cluster, or even if all features are required or just a subset. Also, precisely characterizing the cluster regions (Voronoi cells in D dimensions!) is complicated. For other clustering methods (e.g. spectral clustering) a natural out-of-sample mapping is much harder to determine. For these reasons, we want to determine an out-of-sample mapping that is interpretable, and in a way that is agnostic to how the clustering cost is defined, so it is generally applicable.

What kind of mapping is most suitable? It should be flexible enough to model possibly complex clusters, but interpretable. This rules out black-box models such as neural networks or random forests. Some interpretable models are quite limited in modelling power (e.g. linear models or generalized additive models) and also do not provide a hierarchical clustering. Here we consider sparse oblique decision trees, where each split is a hyperplane using few features.

Sparse oblique decision trees as out-of-sample mapping. Decision trees have several attractive properties in our context. 1) They naturally handle multiple classes, by assigning one leaf (or more) to each class. 2) By using multiple leaves per class, they can model nonconvex and even disconnected classes. 3) They make the clustering hierarchical, i.e., they define a nested set of clusters. 4) As long as the number of nodes is not very large, they are globally interpretable by simple inspection of the nodes and the features they involve, without the need of any approximation or external explanation method. 5) Each leaf can be described by a rule (given by the root-leaf path). (Note we do not consider soft trees, where an instance traverses all the paths in the tree and the prediction is a weighted average of all the leaves, because this makes the tree hard to interpret.)

These comments apply to the traditional axis-aligned trees (where each decision node tests a single input feature). However, axis-aligned trees are inadequate mappings, particularly for high-dimensional data. This is well known because of the restrictive modelling assumption that each leaf must be a box [17]. A CART tree will typically have many leaves (hence being hard to interpret) and yet be quite inaccurate. For example, a CART tree trained on the MNIST dataset (10 digit classes) has hundreds of nodes and a test error of ≈11%, far worse than a linear classifier. But an even stronger limitation comes from the number of features the tree uses. A binary tree with L leaves has $L - 1$ decision nodes, each of which uses one feature. So the total number of features throughout the tree is at most $L - 1$ (usually less because the same feature may be used in multiple nodes), and the number of features along any one root-leaf path is even smaller ($\log_2 L$ for a complete tree). With high-dimensional data, i.e., when the number of features in an instance is large, this restriction is crippling—particularly if we use one leaf per cluster, as some recent papers do (see section 2)

In contrast, sparse oblique trees achieve much higher accuracy and yet the tree remains interpretable. The hyperplane splits are a good model for high-dimensional data, which often has correlated features. In practice, using the TAO algorithm (section 8) with a regularized penalty that sparsifies the weight vectors and prunes the tree, accurate sparse oblique trees can be learned that are quite small and use few features at each node, as seen in section 9.

An additional advantage of sparse oblique trees is that their inference is very fast. For k-means, using the natural out-of-sample mapping takes $\mathcal{O}(DK)$ time (to compute the distance to each centroid), while using a tree of depth Δ takes $\mathcal{O}(D\Delta)$ if dense and much less if sparse, which is a huge savings if $K \gg \Delta$ (for a complete tree, $\Delta = \lceil \log_2 K \rceil$).

Joint optimization of the clustering and the tree. One important point and a contribution of our paper is that the correct solution of our problem involves a joint optimization over the clustering assignments (according to the original clustering cost) and the tree parameters (which constrain the assignments). One shortcut to this joint optimization is to optimize the original clustering on its own and then directly fit a tree to that. While this may sometimes provide a reasonable solution, it is suboptimal because the tree will not be able to model the assignments perfectly, particularly when we constrain the tree to be small and use few parameters, to facilitate interpretability. Allowing the assignments and the tree to coadapt, as our algorithm does, results in a better solution, as demonstrated in our experiments

Tree size vs complexity of explanation. There is a tradeoff between accuracy and interpretability in the out-of-sample mapping. A perfectly accurate mapping can be easily obtained—even with axis-aligned trees—by simply growing a tree large enough that all instances in each leaf belong to the same cluster. However, as is well known in classification, this tree will be very large (hence hard to interpret) and overfit the data. In fact, such a tree behaves like a fast data structure to search for the leaf an instance belongs to, rather than like a classifier that generalizes. In classification, a supervised problem, this model selection issue is conveniently solved by cross-validation. In clustering, an unsupervised problem, we instead provide a hyperparameter λ that the user can tune, in an exploratory way, to find a tree that strikes a good compromise between prediction accuracy and explanation simplicity. As seen in section 8, with sparse oblique trees this is achieved with an ℓ_1 penalty $\lambda\|\mathbf{w}_i\|_1$ on the weight vector of each decision node. This has two useful effects: it sparsifies each hyperplane (using fewer features hence being more interpretable) and it automatically prunes the tree: if $\mathbf{w}_i = \mathbf{0}$ for a node i, that node is redundant (as it sends all instances to the same child) and can be removed at the end. By exploring a range of $\lambda \in [0, \infty)$ values (similarly to the regularization path of the Lasso [17]), we get trees of different sizes and so explanations of varying complexity—from highly precise trees that account for minutiae in the data, to less precise trees that give a higher-level, perhaps more fundamental, explanation.

Summary. For any given clustering method defined by a cost function, we seek to learn (jointly with the cluster assignments) an out-of-sample mapping that is a sparse oblique tree.

4 k-MEANS CASE: AN EXACT OBLIQUE TREE

Our paper is agnostic with respect to the clustering criterion. We seek a controllable tradeoff between faithfulness towards this criterion and interpretability of the tree out-of-sample mapping (in size of the tree and sparsity of its hyperplane splits). That said, we consider here the special case of k-means (the squared error distortion). We show that an exact representation of the clustering out-of-sample mapping can be done with a (deep) oblique tree but not with an axis-aligned tree, however deep.

Assume K centroids $\mathbf{c}_1, \dots, \mathbf{c}_K \in \mathbb{R}^D$ (obtained, say, by running k-means), and define the ideal out-of-sample mapping as assigning a point $\mathbf{x} \in \mathbb{R}^D$ to its closest centroid in Euclidean distance (breaking ties in some deterministic way). As is well known, this partitions $\mathbb{R}^D$ into K Voronoi cells, each a convex polytope whose sides are portions of bisector hyperplanes between pairs of centroids (see fig. 1, plot 1). Can we represent this partition exactly, for any $\mathbf{x} \in \mathbb{R}^D$, with a decision tree? That is, with a finite binary tree such that each leaf region is entirely contained in one Voronoi cell, and each Voronoi cell is the union of one or more leaf regions. The answer depends on what type of decision nodes we allow.

Arbitrary decision nodes. The answer is, trivially, yes. We can construct a one-sided tree (an IF-THEN-ELSE sequence) with K leaves, one per cell, and $K - 1$ decision nodes, each with a test of the form "$\mathbf{x}$ is in cell k". However, this is not very helpful.

Axis-aligned decision nodes. The answer is, in general, no, because the cell boundaries are oblique, as shown in fig. 1 (plot 1). This shows that approximating a Voronoi tessellation with an axis-aligned tree will either require a large tree (clumsy and hard to interpret) or incur a large approximation error. Several papers (see section 2) provide some form of theoretical guarantees for axis-aligned trees with K leaves (one per cluster), but this does not translate into good practical performance for the reasons described.

Oblique decision nodes. The answer is yes, always, as given by the following construction (see fig. 1 plots 2–3). The tree is complete of depth $K - 1$ and the decision nodes at depth $k \in \{0, \dots, K - 1\}$ all have tests of the form "$d_i < d_k$" where i is some other centroid, and $d_i = \|\mathbf{x} - \mathbf{c}_i\|_2$ is the Euclidean distance from the input $\mathbf{x}$ to centroid i. Some cells are given by a single leaf, others by more than one (each a subset of the Voronoi cell). In effect, the tree encodes all possible execution paths of computing $\min(d_1, \dots, d_K)$ by scanning the centroids sequentially from 1 to K. The tree is oblique because the test $d_i < d_k$ can be written linearly as $\mathbf{w}_{ik}^T\mathbf{x} + b_{ik}$ (with hyperplane weight vector $\mathbf{w}_{ik}$ and bias b_{ik}):

$$d_i^2 - d_k^2 = \|\mathbf{x} - \mathbf{c}_i\|_2^2 - \|\mathbf{x} - \mathbf{c}_k\|_2^2 = 2(\mathbf{c}_k - \mathbf{c}_i)^T\mathbf{x} + \|\mathbf{c}_i\|_2^2 + \|\mathbf{c}_k\|_2^2. \quad (1)$$

This result is also a corollary of theorem 3.1 in [32], which states that any K-class linear classifier can be exactly represented with an oblique tree of depth $K - 1$, by noting that a Euclidean distance classifier (which the Voronoi tessellation is) is a linear classifier (from eq. (1)).

Although the number of tests is $\binom{K}{2} = \frac{K(K-1)}{2}$, which is much smaller than $2^{K-1} - 1$ (the number of decision nodes), the same test $d_i < d_k$ may appear in multiple decision nodes, each corresponding to a specific portion of the hyperplane. The tree depends on the particular order of the centroids $1, \dots, K$, and different orders result in different trees, possibly of even different sizes. This is because some decision node tests can be redundant (always true or always false no matter the value of $\mathbf{x}$). For example, this would happen in fig. 1 (plots 2–3) if the centroids were in 1D rather than 2D. In general with K centroids, it is an interesting question to consider what the smallest tree is and how to compute it efficiently. However, this is unlikely to be practical if K is large, because even the smallest tree will be huge.

Fig. 1 (plots 4–5) shows a different construction: a much smaller oblique tree, having K leaves (one per cell), but a large approximation error. It is built by recursive partitioning as in CART, taking

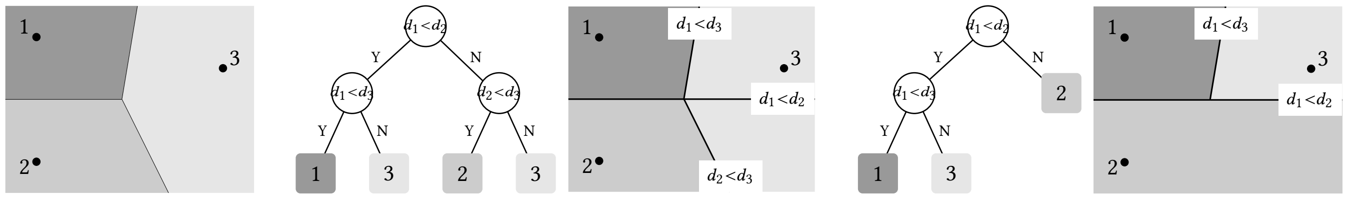

Figure 1: *Plot 1*: Voronoi tessellation induced by $K = 3$ centroids in 2 dimensions. *Plots 2 and 3*: oblique tree with 4 (= 2^{K-1}) leaves and the partition it induces, which is equivalent to the Voronoi tessellation. *Plots 4 and 5*: oblique tree with 3 (= K) leaves and the partition it induces, which is not equivalent to the Voronoi tessellation.

the K centroids as a training set, but where each decision node defines an oblique, bisector split $d_i < d_j$ between two centroids i and j. The resulting tree has a depth between $\lceil \log_2 K \rceil$ and $K - 1$.

This brings us back to our goal: to trade off optimally the clustering accuracy (according to a specific clustering criterion) with the interpretability of the clustering out-of-sample mapping, having the form of a small oblique tree with sparse hyperplane splits.

5 CLUSTERING PROBLEM FORMULATION

Consider a clustering algorithm defined by optimizing problem:

$$\min_{\mathbf{Z},\mathbf{\Psi}} E(\mathbf{Z},\mathbf{\Psi}) \quad \text{s.t.} \quad \mathbf{Z}^T\mathbf{1} = \mathbf{1}, \quad \mathbf{Z} \in \{0,1\}^{K\times N} \tag{2}$$

given a training set $\mathbf{X}_{D\times N} = (\mathbf{x}_1, \dots, \mathbf{x}_N)$ and seeking K clusters, where $\mathbf{1}$ is a vector of ones of suitable dimension. Here, $E(\mathbf{Z},\mathbf{\Psi})$ is a cost function defining the goodness of a particular clustering. The assignment variables $\mathbf{Z}_{K\times N} = (\mathbf{z}_1, \dots, \mathbf{z}_N)$ indicate which cluster each instance $\mathbf{x}_n$ is assigned to, encoded as one-hot vectors. The variables $\mathbf{\Psi}$ include any other variables learnt by the algorithm, for example the cluster centroids in k-means. Any clustering algorithm must produce as output the assignment variables, but not all algorithms need learn additional variables $\mathbf{\Psi}$. Our formulation can also handle soft clustering, where $\mathbf{z}_n \in [0,1]^K$, or other constraints in problem (2), e.g. must-link or cannot-link constraints between pairs of instances, but for simplicity we ignore this here.

For example, the following cost function:

$$E(\mathbf{Z},\mathbf{\Psi}) = \textstyle\sum_{n=1}^N \sum_{k=1}^K z_{kn}\, d(\mathbf{x}_n, \boldsymbol{\psi}_k), \tag{3}$$

where $\boldsymbol{\psi}_1, \dots, \boldsymbol{\psi}_K \in \mathbb{R}^D$ and d is a distance function, corresponds to centroid-based methods, such as k-means (where d is the squared Euclidean distance), spherical k-means (d is the cosine distance, or equivalently the negative dot product, assuming instances and centroids are normalized), k-medoids ($\boldsymbol{\psi}_k$ is constrained to be a training instance), k-modes [7, 30], k-medians, k-centers, etc. [17]. All of these clustering methods can be solved (approximately, at least) via various algorithms. The most popular one is alternating optimization:

- **Assignment step** $\min_{\mathbf{Z}} \sum_{n=1}^N \sum_{k=1}^K z_{kn}\, d(\mathbf{x}_n, \boldsymbol{\psi}_k)$ s.t. $\mathbf{Z}^T\mathbf{1} = \mathbf{1}$, $\mathbf{Z} \in \{0,1\}^{K\times N}$: set $\mathbf{z}_n$ to the cluster k having smallest $d(\mathbf{x}_n, \boldsymbol{\psi}_k)$, separately for each instance $\mathbf{x}_1, \dots, \mathbf{x}_N$.
- **Centroid step** $\min_{\mathbf{\Psi}} \sum_{n=1}^N \sum_{k=1}^K z_{kn}\, d(\mathbf{x}_n, \boldsymbol{\psi}_k)$: set $\boldsymbol{\psi}_k = \arg\min \sum_{n:z_{nk}=1} d(\mathbf{x}_n, \boldsymbol{\psi}_k)$. For example, for $d = \ell_2^2$ and d = cosine distance this sets $\boldsymbol{\psi}_k$ to the mean of the instances in cluster k (normalized, for the cosine distance).

Other examples use a cost function defined on a graph where each instance is one vertex, such as spectral clustering, correlation clustering, etc.

Most clustering methods use a cost function E, but not all. One is mean-shift clustering, where the number of clusters is implicitly defined by the modes of a kernel density estimate. Another is agglomerative or divisive clustering, which iteratively merge instances or split groups, respectively (although some work has sought to define a cost function for these algorithms [10]).

6 INTERPRETABLE CLUSTERING PROBLEM FORMULATION

We now solve problem (2) but demand that the cluster assignments $\mathbf{z}_n$ be produced by an out-of-sample mapping $\mathbf{T}(\mathbf{x}_n;\mathbf{\Theta})$, a classification tree with parameters $\mathbf{\Theta}$. That is:

$$\min_{\mathbf{\Psi},\mathbf{\Theta}} E(\mathbf{T}(\mathbf{X};\mathbf{\Theta}),\mathbf{\Psi}) + \lambda\,\phi(\mathbf{\Theta}) \tag{4}$$

where $\mathbf{T}(\cdot;\mathbf{\Theta})\colon \mathbb{R}^D \rightarrow \{1,\dots,K\}$ (one-hot encoded) and the notation $\mathbf{T}(\mathbf{X};\mathbf{\Theta})$ stands for $\mathbf{T}(\mathbf{x}_1;\mathbf{\Theta}), \dots, \mathbf{T}(\mathbf{x}_N;\mathbf{\Theta})$. We consider either an axis-aligned tree or an oblique tree, depending on whether the decision nodes use a single feature or a (sparse) linear combination of them. As for the leaf predictors, we consider either a class label or a histogram over classes (where the most frequent class is the final prediction). The regularization term $\phi(\mathbf{\Theta})$ with user parameter $\lambda \geq 0$ controls the tree complexity. This is explained in section 8.

This is a difficult optimization problem because the tree $\mathbf{T}$ (which makes hard decisions at its nodes) is not a differentiable function of $\mathbf{\Theta}$, and besides $\mathbf{T}$ appears as an argument of the nonlinear function E. To make (4) amenable to iterative optimization, we apply the *method of auxiliary coordinates* [6, 8]. We first rewrite (4) as a constrained problem by introducing the assignment variables:

$$\begin{aligned} &\min_{\mathbf{Z},\mathbf{\Psi},\mathbf{\Theta}} \; E(\mathbf{Z},\mathbf{\Psi}) + \lambda\,\phi(\mathbf{\Theta}) \\ &\text{s.t.} \quad \mathbf{Z} = \mathbf{T}(\mathbf{X};\mathbf{\Theta}), \quad \mathbf{Z}^T\mathbf{1} = \mathbf{1}, \quad \mathbf{Z} \in \{0,1\}^{K\times N}. \end{aligned} \tag{5}$$

7 INTERPRETABLE CLUSTERING: OPTIMIZATION ALGORITHM

We apply a penalty method to the equality constraints in (5) that involve $\mathbf{T}$ (leaving the other constraints in place) and define the problem (similar to that in [5, 31] for dimensionality reduction):

$$\begin{aligned} &\min_{\mathbf{Z},\mathbf{\Psi},\mathbf{\Theta}} \; E(\mathbf{Z},\mathbf{\Psi}) + \lambda\,\phi(\mathbf{\Theta}) + \mu\, P(\mathbf{Z},\mathbf{T}(\mathbf{X};\mathbf{\Theta})) \\ &\text{s.t.} \quad \mathbf{Z}^T\mathbf{1} = \mathbf{1}, \quad \mathbf{Z} \in \{0,1\}^{K\times N} \end{aligned} \tag{6}$$

where $\mu \geq 0$ is a penalty parameter and P is a penalty function satisfying $P(\mathbf{z},\mathbf{z}) = 0$ and $P(\mathbf{z},\mathbf{z}') > 0$ if $\mathbf{z} \neq \mathbf{z}'$. The notation $P(\mathbf{Z},\mathbf{T}(\mathbf{X};\mathbf{\Theta}))$ stands for $P(\mathbf{z}_1,\mathbf{T}(\mathbf{x}_1;\mathbf{\Theta})) + \cdots + P(\mathbf{z}_N,\mathbf{T}(\mathbf{x}_N;\mathbf{\Theta}))$. If $\mu \rightarrow \infty$ then (5) and (6) have the same solutions (in fact, as seen later, this happens if $\mu > \mu^*$ for some finite μ^*). The objective of (6) becomes progressively ill-conditioned (hence harder to optimize numerically) as μ increases. Thus, rather than optimizing (6) directly for a very large value of μ, we follow a path of solutions starting from small μ, as is common with quadratic-penalty and other homotopy methods [27].

For a fixed value of μ, we optimize (6) using alternating optimization over the clustering variables $(\mathbf{Z},\mathbf{\Psi})$ and the tree parameters $\mathbf{\Theta}$. This results in an intuitive algorithm with the following steps:

- **Clustering step** (over $\mathbf{Z},\mathbf{\Psi}$ given $\mathbf{\Theta}$):

$$\min_{\mathbf{Z},\mathbf{\Psi}} E(\mathbf{Z},\mathbf{\Psi}) + \mu \sum_{n=1}^{N} P(\mathbf{z}_n,\overline{\mathbf{z}}_n) \qquad \text{s.t.} \quad \mathbf{Z}^T\mathbf{1} = \mathbf{1}, \quad \mathbf{Z} \in \{0,1\}^{K\times N} \tag{7}$$

  where $\overline{\mathbf{z}}_n = \mathbf{T}(\mathbf{x}_n;\mathbf{\Theta})$ is a constant vector for $n = 1,\dots,N$. This can be seen as the original clustering problem (2) but with a regularization term that pulls the assignments $\mathbf{Z}$ towards $\overline{\mathbf{Z}}$. This step can be usually solved using a modified version of the algorithm for (2). For example, for centroid clustering as in section 5, the centroid step does not change, and the assignment step takes the following form (separately for each $n = 1,\dots,N$): $\min_{\mathbf{z}_n} \sum_{k=1}^{K} \delta_{kn} z_{kn}$ s.t. $\mathbf{1}^T\mathbf{z}_n = 1$, $\mathbf{z}_n \in \{0,1\}^K$, where $\delta_{kn} = d(\mathbf{x}_n,\boldsymbol{\psi}_k) + \mu\, p_{kn} \geq 0$ and p_{kn} is the penalty value $P(\mathbf{z}_n,\overline{\mathbf{z}}_n)$ when $\mathbf{z}_n$ picks cluster k. The solution is to set $\mathbf{z}_n$ to the cluster k having smallest δ_{kn}.
- **Tree step** (over $\mathbf{\Theta}$ given $\mathbf{Z},\mathbf{\Psi}$):

$$\min_{\mathbf{\Theta}} \sum_{n=1}^{N} P(\mathbf{z}_n,\mathbf{T}(\mathbf{x}_n;\mathbf{\Theta})) + \frac{\lambda}{\mu}\phi(\mathbf{\Theta}). \tag{8}$$

  This takes the form of a classification problem with loss P, tree classifier $\mathbf{T}$ and regularization ϕ, which we can solve using the TAO algorithm (section 8). Importantly, we can use warm-start, i.e., initialize TAO using the tree parameters $\mathbf{\Theta}$ of the previous iteration. Note the tree structure is the same throughout the algorithm, and given in the first iteration. At the end of the algorithm, we can prune nodes with $\mathbf{w}_i = \mathbf{0}$.

Choice of penalty function and type of tree leaf. There are multiple choices of penalty function in (6) and of what form of label the tree leaves should use. The following two possibilities are most convenient algorithmically:

- P = 0/1 loss ($P(\mathbf{z},\mathbf{z}) = 0$ and $P(\mathbf{z},\boldsymbol{\theta}) = 1$ if $\mathbf{z} \neq \boldsymbol{\theta}$) and leaf i has a constant label (one-hot encoded) $\boldsymbol{\theta}_i \in \{0,1\}^K$ (with $\mathbf{1}^T\boldsymbol{\theta}_i = 1$), which gives the class label it predicts.
- P = squared error ($P(\mathbf{z},\boldsymbol{\theta}) = \|\mathbf{z}-\boldsymbol{\theta}\|_2^2$) and leaf i has a constant histogram $\boldsymbol{\theta}_i \in [0,1]^K$ (with $\mathbf{1}^T\boldsymbol{\theta}_i = 1$), which gives the probability of each class in that leaf.

The above means that the tree step (which is always a classification problem with inputs $\mathbf{X}$ and "ground-truth" class labels $\mathbf{Z}$) uses either the 0/1 loss or the squared error, respectively. TAO can handle both types of loss function besides the regularization term ϕ.

```
input $\mathbf{X}_{D\times N} = \{\mathbf{x}_1,\cdots,\mathbf{x}_N\}$, $\lambda \geq 0$, $a > 0$, $\mu_0 > 0$
      initial tree structure and random $\mathbf{\Theta}$
$\mathbf{Z},\mathbf{\Psi} \leftarrow \arg\min E(\mathbf{Z},\mathbf{\Psi})$  s.t.  $\mathbf{Z}^T\mathbf{1}=\mathbf{1}$, $\mathbf{Z}\in\{0,1\}^{K\times N}$      Free clustering
$\mathbf{\Theta} \leftarrow \begin{cases} \arg\min P(\mathbf{Z},\mathbf{T}(\mathbf{X};\mathbf{\Theta})), & \lambda = 0 \\ 0, & \lambda > 0 \end{cases}$      Direct tree fit
$\mu \leftarrow \mu_0$
repeat
    $\mathbf{Z},\mathbf{\Psi} \leftarrow \arg\min E(\mathbf{Z},\mathbf{\Psi}) + \mu P(\mathbf{Z},\mathbf{T}(\mathbf{X};\mathbf{\Theta}))$      Clustering step
        s.t.  $\mathbf{Z}^T\mathbf{1} = \mathbf{1}$, $\mathbf{Z} \in \{0,1\}^{K\times N}$
    $\mathbf{\Theta} \leftarrow P(\mathbf{Z},\mathbf{T}(\mathbf{X};\mathbf{\Theta})) + \frac{\lambda}{\mu}\phi(\mathbf{\Theta})$      Tree step
    $\mu \leftarrow \mu \cdot a$
until $\mathbf{Z} = \mathbf{T}(\mathbf{X};\mathbf{\Theta})$ and no parameter change
return tree $\mathbf{T}(\cdot;\mathbf{\Theta})$ and $\mathbf{Z},\mathbf{\Psi}$
```

Figure 2: Pseudocode of the joint optimization framework for interpretable clustering

Beginning of the solution path. Eq. (6) has an intuitive solution $(\mathbf{Z}^*(0),\mathbf{\Psi}^*(0),\mathbf{\Theta}^*(0))$ for $\mu \rightarrow 0^+$: $\mathbf{Z}^*(0)$ and $\mathbf{\Psi}^*(0)$ are the minimizers of (2), without the constraint on $\mathbf{T}$, which we call the *free clustering*; and $\mathbf{\Theta}^*(0)$ is either $\mathbf{0}$ if $\lambda > 0$ (which corresponds to a single-leaf tree, and a single cluster) or, if $\lambda = 0$, the minimizer of $P(\mathbf{Z}^*(0),\mathbf{T}(\mathbf{X};\mathbf{\Theta}))$. The latter corresponds to fitting a classification tree to a training set $(\mathbf{X},\mathbf{Z}^*(0))$ that maps each instance to its free clustering cluster. As noted earlier, this is suboptimal.

End of the solution path and stopping criterion. As noted earlier, when $\mu \rightarrow \infty$ then the constraints $\mathbf{Z} = \mathbf{T}(\mathbf{X};\mathbf{\Theta})$ in (5) must be satisfied and the solution(s) of (5) and (6) coincide. In fact, this is true when $\mu > \mu^*$ for a certain finite $\mu^* > 0$. This is because both the assignment variables $\mathbf{z}_n$ and the tree outputs $\boldsymbol{\theta}_i$ at the leaves take values from a finite set. This happens because either the variables are discrete, or (with histogram leaves) they are real but the total number of possible histograms is finite (because the total number of training instance subsets at any leaf is finite). Hence, the total number of possible values for the penalty $P(\mathbf{Z},\mathbf{T}(\mathbf{X};\mathbf{\Theta}))$ is also finite, so when μ is large enough, the penalty term $\mu P(\mathbf{Z},\mathbf{T}(\mathbf{X};\mathbf{\Theta}))$ must jump to 0 and make $\mathbf{Z} = \mathbf{T}(\mathbf{X};\mathbf{\Theta})$. We could compute an upper bound for the value μ^* when this happens, but it is not necessary. We can simply stop the algorithm when, after one iteration (both clustering and tree steps) $\mathbf{Z} = \mathbf{T}(\mathbf{X};\mathbf{\Theta})$ and the parameters did not change (or changed less than a set tolerance).

Schedule of μ. To increase μ progressively, we use a multiplicative schedule of the form $\mu_t = \mu_0\, a^t$ for $t = 0,1,2\dots$ with $a > 1$ and $\mu_0 > 0$. The slower we increase μ, the better we follow the solution path, but the more computation. In practice we take $a \in (1,2)$. Pseudocode in fig. 2 summarizes the proposed algorithm.

8 REVIEW OF TREE ALTERNATING OPTIMIZATION (TAO)

As a subproblem of our algorithm, we have to train a sparse oblique classification tree with some specific loss function and a regularization term (eq. (8)). This means learning the structure of the tree and the parameters at the nodes. Besides, it should be able to take an initial tree and improve over it, so the tree step decreases the overall objective function in (6) (i.e., warm-start). Finally, it should be computationally efficient. A recent algorithm, *Tree Alternating Optimization (TAO)*, satisfies all of the above, and we use it here. The traditional, recursive partitioning algorithms, such as CART [3] or C4.5 [29], are inadequate because they grow a tree greedily from scratch (at each split using a proxy purity criterion) rather than improving a given tree globally with respect to a specific loss function. They are also quite suboptimal, particularly with oblique trees, and so are typically used with axis-aligned trees. Other algorithms are based on a brute-force search via branch-and-bound or mixed-integer optimization and have a worst-case complexity that is higher than exponential, so they are impractical.

The underlying mechanism of TAO [4] is to take a parametric tree of fixed structure (here, complete of depth Δ), and perform optimization steps in turn over the parameters of a single node (decision node or leaf) while keeping the rest of the parameters fixed. This succeeds because of the two theorems that we describe below. It works quite similar to how one would optimize a neural network, but instead of gradients (which do not apply) TAO uses alternating optimization on a fixed tree structure. This results in iteratively updating all the parameters in the tree (decision node hyperplanes and leaf output values), with a monotonic decrease of the objective function at each iteration over all nodes and convergence to a local optimum.

TAO handles naturally the objective function provided over the tree step in (8). Let us formally define $\mathbf{T}(\mathbf{x};\Theta)$ as a binary decision tree of some predetermined structure with parameters $\Theta = \{(\mathbf{w}_i, w_{i0})\}_{i\in\mathcal{D}} \cup \{\boldsymbol{\theta}_i\}_{i\in\mathcal{L}}$, decision nodes in set $\mathcal{D}$ and leaves in set $\mathcal{L}$. The prediction of $\mathbf{T}(\mathbf{x};\Theta)$ is obtained by routing $\mathbf{x}$ from the root to exactly one leaf and outputting the cluster label or histogram $\boldsymbol{\theta}_i \in \mathbb{R}^K$. At a decision node i we apply a decision function $f_i(\mathbf{x};\mathbf{w}_i, w_{i0})$: $\mathbb{R}^D \rightarrow \{\texttt{left}_i, \texttt{right}_i\} \subset \mathcal{D}\cup\mathcal{L}$ denoting "go to the right child if $\mathbf{w}_i^T\mathbf{x} + w_{i0} \geq 0$, else go to the left child". In this formulation axis-aligned trees are a special case of oblique trees, where $\mathbf{w}_i$ is all zeros but one at the threshold feature index and $-w_{i0}$ serves as a threshold value. As a regularization term in (8) we use an ℓ_1 penalty: $\phi(\Theta) = \sum_{i\in\mathcal{D}} \|\mathbf{w}_i\|_1$. For oblique trees this encourages sparsifying the hyperplanes. For axis-aligned trees, the regularization is interpreted as equalling λ if using one feature and 0 if using no features. This penalty for both types of trees can make decision nodes redundant (when $\mathbf{w}_i = \mathbf{0}$) so they can be pruned at the end.

TAO is based on two theorems. First, eq. (8) *separates over any subset of non-descendant nodes* (e.g. all the nodes at the same depth); this follows from the fact that the tree makes hard decisions. All such nodes may be optimized in parallel. Second, optimizing over the parameters of a single node i simplifies to a well-defined *reduced problem* over the instances that currently reach node i (the *reduced set* $\mathcal{R}_i \subset \{1,\dots,N\}$). The form of the reduced problem depends on the type of node:

Decision node It is a *weighted 0/1 loss binary classification problem*, where the two classes correspond to the left and right child, which are the only possible outcomes for an instance. Child $\texttt{left}_i$ ($\texttt{right}_i$) incurs a loss (weight) given by the prediction of the leaf reached from the left (right) child's subtree. Thus, each instance is assigned as *pseudolabel* the child with lower loss. The reduced problem takes the form (where $\overline{L}$ is the said loss):

$$\min_{\mathbf{w}_i, w_{i0}} \sum_{n\in\mathcal{R}_i} \bar{L}(\mathbf{z}_n, f_i(\mathbf{x};\mathbf{w}_i, w_{i0})) + \lambda\,\|\mathbf{w}_i\|_1. \tag{9}$$

For oblique nodes this problem is NP-hard but can be well approximated with a convex surrogate; we use ℓ_1-regularized logistic regression where each instance is weighted by the loss difference between the winner child and the other child, and solve it using LIBLINEAR [11]. We can guarantee a monotonic decrease in the objective by only accepting this update if it improves over the previous step. For axis aligned splits the optimal solution is found by enumeration.

Leaf The reduced problem consists of optimizing the original loss but over the leaf classifier on its reduced set:

$$\min_{\boldsymbol{\theta}_i} \sum_{n\in\mathcal{R}_i} P(\mathbf{z}_n, \boldsymbol{\theta}_i). \tag{10}$$

For the two penalty functions that we consider the solution is either a majority label or a normalized histogram.

Given an initial tree structure with initial parameter values, the resulting algorithm repeatedly visits nodes in reverse breadth-first search order. Each iteration trains all nodes at the same depth (in parallel) from the leaves to the root, by solving either an ℓ_1-regularized logistic regression for oblique splits or by enumeration in axis-aligned case, or the above exact solution at each leaf.

9 EXPERIMENTS

Our experimental findings illustrate that 1) The proposed optimization framework produces more optimal trees for k-means and spherical k-means objectives than just naively fitting a tree to free clustering for both axis-aligned, and especially for oblique trees; 2) The resulting oblique trees are sparse, shallow and easily interpretable, while distorting very little the cluster quality as measured by the objective function, and one can directly tune the tradeoff between these with a single sparsity parameter λ.

We first demonstrate the effectiveness of the proposed algorithm in terms of objective function improvement, followed by quantitative comparison of axis-aligned and oblique trees. Then we demonstrate the interpretability of sparse oblique trees on image and document data. For experiments with oblique trees, we obtain the initial tree parameters by recursive k-means bipartitioning: starting from the root we split the points into two clusters (left and right children) using k-means with $K = 2$, and continue this recursively until we construct a complete tree of given depth Δ. More details on the experiments, including datasets and implementation, can be found in appendix A.

9.1 The benefit of joint optimization

To see whether the proposed joint optimization framework can produce more optimal clustering trees than just directly fitting a tree to free clustering assignments, in fig. 3 we plot how the value of the clustering objective function changes as we iterate through the μ schedule. The beginning of the path corresponds to the direct fit of an oblique TAO tree to free clustering assignments, where the initial tree is a complete tree of depth Δ=6 for MNIST and Δ=4 for Web of Science, and has random initial parameters. The downward trend of the curves clearly demonstrate the effectiveness of

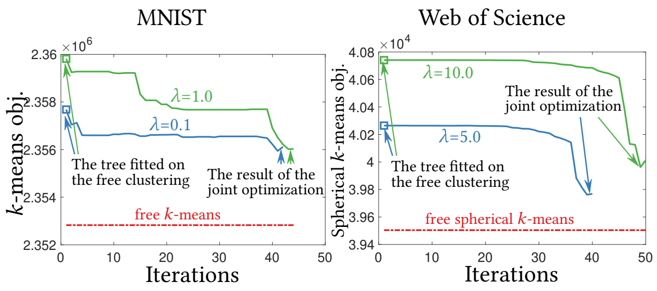

Figure 3: The plots of the k-means and spherical k-means objectives for MNIST and Web of Science datasets as our joint optimization algorithm progresses over the μ schedule. Each point shows the clustering objective when the cluster assignments are given by a sparse oblique tree. The initial points are the direct fit of TAO to the free clustering assignments.

Method	cost (%)	Δ	#leaves
CART	9.77	4	10
CART	5.58	7	20
CART	4.61	8	40
TAO	4.34	7	10
IMM	4.34	7	10
ExGreedy	4.25	7	10
ExKMC	2.57	10	20
TAO	2.53	10	20
ExKMC	1.95	12	40
TAO	1.70	12	39

FIFA19 (16k, 80, 10)

Method	cost (%)	Δ	#leaves
CART	3.30	5	10
CART	2.54	6	20
IMM	1.19	9	10
TAO	0.77	9	10
CART	0.66	9	40
ExGreedy	0.59	9	10
ExKMC	0.49	12	20
TAO	0.48	12	20
ExKMC	0.08	13	40
TAO	0.00	12	37

Adult (30k, 103, 10)

Table 1: Quantitative comparison of axis-aligned trees for two tabular datasets. "TAO" refers to the result of joint optimization with an axis-aligned tree. The dataset name is followed (N, D, k), where N is the number of points, D feature dimension, and k is the number of clusters. The cost (%) is the percentage increase from the reference k-means clustering. We also report the depth Δ of the tree and the number of leaves. Sorted by the decreasing k-means objective value.

the proposed joint optimization framework, and that the margin of improvement can be quite significant.

9.2 Quantitative comparison

Axis aligned trees. For two tabular datasets we evaluate the performance of axis-aligned trees in Table 1. For methods that allow trees of various size we report multiple of those in order to compare the objective in terms of different tree sizes. The ExKMC algorithm is specifically designed for k-means objective and can induce trees with an arbitrary number of leaves. For these tabular datasets it performs much better than a naive CART tree fitted to cluster assignments. To further improve the ExKMC cluster quality, we initialize axis-aligned TAO trees from ExKMC at the beginning of the path ($\mu = 0^+$), and proceed with alternating optimizations over the μ schedule. As the results in Table 1 show, in most cases the proposed algorithm can improve the objective, and in some cases produce even smaller trees (because some leaves might become dead and so pruned). But as we will see later, for other types of data, such as image pixels, axis-aligned trees are not very suitable to optimally represent the clusters (without being very large).

	Method	cost (%)	#parameters	#features/node	Δ	#leaves
MNIST (60k,784,10)	IMM	14.34	28	1	9	10
	Ex-Greedy	12.48	28	1	8	10
	CART	11.54	28	1	4	10
	TAO	7.90	199	23	4	9
	CART	1.87	3070	1	16	1024
	ExKMC	1.81	3070	1	29	1024
	TAO	1.50	753	66	5	12
	TAO	0.94	1372	96	4	15
FashMNIST (60k,784,10)	IMM	28.34	28	1	9	10
	CART	23.33	28	1	4	10
	Ex-Greedy	17.28	28	1	7	10
	TAO	5.22	452	43	5	11
	ExKMC	2.49	3070	1	59	1024
	CART	1.75	3070	1	17	1024
	TAO	1.74	825	80	5	11
	TAO	0.44	2081	146	4	15
Letter (20k,16,26)	IMM	35.02	76	1	25	26
	CART	30.61	76	1	10	26
	ExGreedy	27.78	76	1	21	26
	TAO	9.94	523	15	5	32
	TAO	4.06	516	8	6	52
	CART	2.89	3070	1	25	1024
	ExKMC	2.91	3070	1	39	1024
	TAO	2.75	858	12	6	64

Table 2: Like Table 1, but now "TAO" refers to the joint optimization with an oblique tree. We also report the number of parameters and the average number of features used at a decision node over the whole tree, which is constant (=1) for axis-aligned splits and varies in (sparse) oblique splits.

	Method	cost (%)	#parameters	#features/node	Δ	#leaves
WebSc. (47k,22k,7)	random	5.81				
	CART	2.89	19	1	6	7
	CART	1.32	190	1	37	64
	CART	0.99	766	1	70	256
	TAO	0.80	365	59	6	7
	CART	0.62	3070	1	111	1024
	TAO	0.41	859	121	6	8
	TAO	0.35	1209	199	4	7
Amaz. (46k,13k,12)	random	13.13				
	CART	2.14	34	1	10	12
	CART	0.56	766	1	73	256
	TAO	0.56	734	59	4	13
	TAO	0.29	532	42	6	13
	CART	0.28	3070	1	125	1024
	TAO	0.18	673	54	6	13

Table 3: Like Tables 1 and 3, but for spherical k-means. TAO refers to an oblique tree resulted from the proposed algorithm for joint optimization. The cost is measured in terms of the spherical k-means objective. Random refers to the result from random assignment of cluster labels. It is included to show the significance of small improvements in percentage, because the scale of the cosine distance is quite small.

Oblique trees. We test the proposed algorithm for oblique trees on 3 simple image datasets where features are grayscale pixels (MNIST, FashionMNIST) or statistical moments (Letter). As an initial tree we take a complete tree of depth Δ and random node parameters. For comparison we also evaluate the performance of axis-aligned trees. Table 2 summarizes the results. Clearly, sparse oblique trees perform much better than any other axis-aligned tree but still remaining shallow and using only a small subset of features at decision nodes. For example, on FashionMNIST an oblique with 11 leaves matches the performance of axis-aligned trees with 1024 leaves, while using on average only 10% of features at a decision node. In section 9.3 we visualize these trees to show their interpretability.

Spherical k-means. The literature on text mining suggests that a distance based on cosine similarity is a more appropriate measure in clustering documents than a Euclidean distance [12]. The corresponding change to the k-means objective is to use a cosine distance (1.0 − cosine similarity) in place of Euclidean, which results in a spherical k-means objective. The assignment step from the regular k-means will now be based on a cosine distance, but the centroid step will still use the mean vector. Given the modularity of our proposed optimization framework, there will also be a minimal change to our algorithm: the assignment and centroid steps will correspondingly adjust to the spherical k-means objective, but the step over the tree will still be the minimization of squared error plus regularization.

We evaluate the proposed optimization approach for spherical k-means objective with oblique decision trees for two document datasets, where the features are tf-idf transformed unigram bag-of-words. Table 3 summarizes the quantitative results. For comparison we also show axis-aligned CART trees fitted to cluster assignments. The other algorithms designed specifically for regular Euclidean k-means are not evaluated here, because it might not be appropriate to compare them in terms of the spherical k-means objective. Similar to image datasets, oblique trees achieve more optimal clustering objective while still being small and using on the order of 0.5% features at a decision node. Section B shows a visualization and interpretation of one such sparse oblique tree for the Amazon Reviews dataset.

9.3 Interpreting k-means clustering for Fashion MNIST

Fashion MNIST is a dataset of normalized grayscale images of different types of clothes and shoes with 10 number of classes. Running the free k-means with 10 clusters and visualizing the average of those clusters (see the right column of fig. 4) reveal that in most cases clusters are dominated by items with a similar shape. An exception is cluster 7, which has mostly Sandals ($\approx$ 50%) mixed with Shirts, T-shirts and Dresses ($\approx$ 25%). On the left column of fig. 4 we visualize 2 sparse oblique trees obtained from our proposed algorithm with different levels of sparsity parameter: $\lambda = 10$ and $\lambda = 100$. The sparser top tree has 11 leaves, from which it follows that 2 leaves (numbered 14 and 24) are predicting the same cluster, which is, quite interestingly, turns out to be the mixed cluster 7. Looking at the mean image of these two leaves we could see that now Sandals are better separated in leaf 24 ($\approx$ 70%) along with the other shoe (Sneakers $\approx$ 10%), while Shirts/T-Shirts/Dresses dominate the leaf 14 ($\approx$ 70%). From the tree we can see that a separation between these leaves takes place at a decision node 3, which uses pixels in the middle right (corresponding to the ankle side of shoes) to send points to the subtree with Sandals. Looking at the other leaves of that same subtree (rooted at node 6) we see that it is mostly dominated by shoe images and the decision nodes learn to distinguish them based on specific characteristics such as height or heel position. The tree at the bottom has denser decision nodes and is more accurate in terms of k-means objective. In the fig. 4 we provide additional annotations to provide some possible interpetation of clustering. In summary, the clustering oblique tree can serve as a helpful exploratory tool to gain more insight about the data.

10 CONCLUSION

We have proposed a way to redefine any clustering method defined by a cost function of the cluster assignments, by constraining the latter to be produced by an interpretable out-of-sample mapping. The mapping is given by a sparse oblique decision tree, which is far more powerful than the usual axis-aligned trees, particularly with high-dimensional data. The tree makes it possible to explain how a prediction was arrived at by simple inspection. The resulting problem involves a joint optimization over the original clustering parameters and the tree. We give an algorithm that iteratively alternates a step that optimizes over the cluster assignments with a step that fits a classification tree until both eventually agree. The complexity of the tree is controlled by a hyperparameter, which allows a user to explore a tradeoff between accuracy of the clustering and simplicity of the explanation. In our experiments we have demonstrated this with k-means-type methods, but the approach applies to other clustering methods defined by a cost function.

Acknowledgments. Work supported by NSF award IIS−2007147.

REFERENCES

[1] Jayanta Basak and Raghu Krishnapuram. 2005. Interpretable Hierarchical Clustering by Constructing an Unsupervised Decision Tree. *IEEE Trans. Knowledge and Data Engineering* 17, 1 (Jan. 2005), 121–132.
[2] Dimitris Bertsimas, Agni Orfanoudaki, and Holly Wiberg. 2021. Interpretable Clustering: An Optimization Approach. *Machine Learning* 110, 1 (Jan. 2021).
[3] Leo J. Breiman, Jerome H. Friedman, R. A. Olshen, and Charles J. Stone. 1984. *Classification and Regression Trees.*
[4] Miguel Á. Carreira-Perpiñán and Pooya Tavallali. 2018. Alternating Optimization of Decision Trees, with Application to Learning Sparse Oblique Trees. In *(NEURIPS)*, S. Bengio, H. Wallach, H. Larochelle, K. Grauman, N. Cesa-Bianchi, and R. Garnett (Eds.), Vol. 31. 1211–1221.
[5] Miguel Á. Carreira-Perpiñán and Max Vladymyrov. 2015. A Fast, Universal Algorithm to Learn Parametric Nonlinear Embeddings. In *(NIPS)*, C. Cortes, N. D. Lawrence, D. D. Lee, M. Sugiyama, and R. Garnett (Eds.), Vol. 28. 253–261.
[6] Miguel Á. Carreira-Perpiñán and Weiran Wang. 2012. Distributed Optimization of Deeply Nested Systems. (Dec. 24 2012). arXiv:1212.5921.
[7] Miguel Á. Carreira-Perpiñán and Weiran Wang. 2013. The K-Modes Algorithm for Clustering. (April 23 2013). arXiv:1304.6478.
[8] Miguel Á. Carreira-Perpiñán and Weiran Wang. 2014. Distributed Optimization of Deeply Nested Systems. In *Proc. of the 17th (AISTATS 2014)*, Samuel Kaski and Jukka Corander (Eds.). Reykjavik, Iceland, 10–19.
[9] Moses Charikar and Lunjia Hu. 2022. Near-Optimal Explainable k-Means for All Dimensions. In *Proc. of the 33rd ACM-SIAM Symposium on Discrete Algorithms (SODA 2022)*, Joseph Naor and Niv Buchbinder (Eds.). Virtual, 2580–2606.
[10] Sanjoy Dasgupta. 2016. A Cost Function for Similarity-Based Hierarchical Clustering. In *Proc. of the 48th ACM symposium on Theory of Computing (STOC 2016).*
[11] Rong-En Fan, Kai-Wei Chang, Cho-Jui Hsieh, Xiang-Rui Wang, and Chih-Jen Lin. 2008. LIBLINEAR: A Library for Large Linear Classification. *J. Machine Learning Research* 9 (Aug. 2008), 1871–1874.
[12] Ronen Feldman and James Sanger. 2006. *The Text Mining Handbook. Advanced Approaches in Analyzing Unstructured Data.* Cambridge University Press.
[13] Ricardo Fraiman, Badih Ghattas, and Marcela Svarc. 2013. Interpretable Clustering Using Unsupervised Binary Trees. *Advances in Data Analysis and Classification* 7, 2 (June 2013), 125–145.
[14] Nave Frost, Michal Moshkovitz, and Cyrus Rashtchian. 2020. ExKMC: Expanding Explainable k-Means Clustering. (July 2 2020). arXiv:2006.02399.
[15] Buddhima Gamlath, Xinrui Jia, Adam Polak, and Ola Svensson. 2021. Nearly-Tight and Oblivious Algorithms for Explainable Clustering. In *(NEURIPS)*,

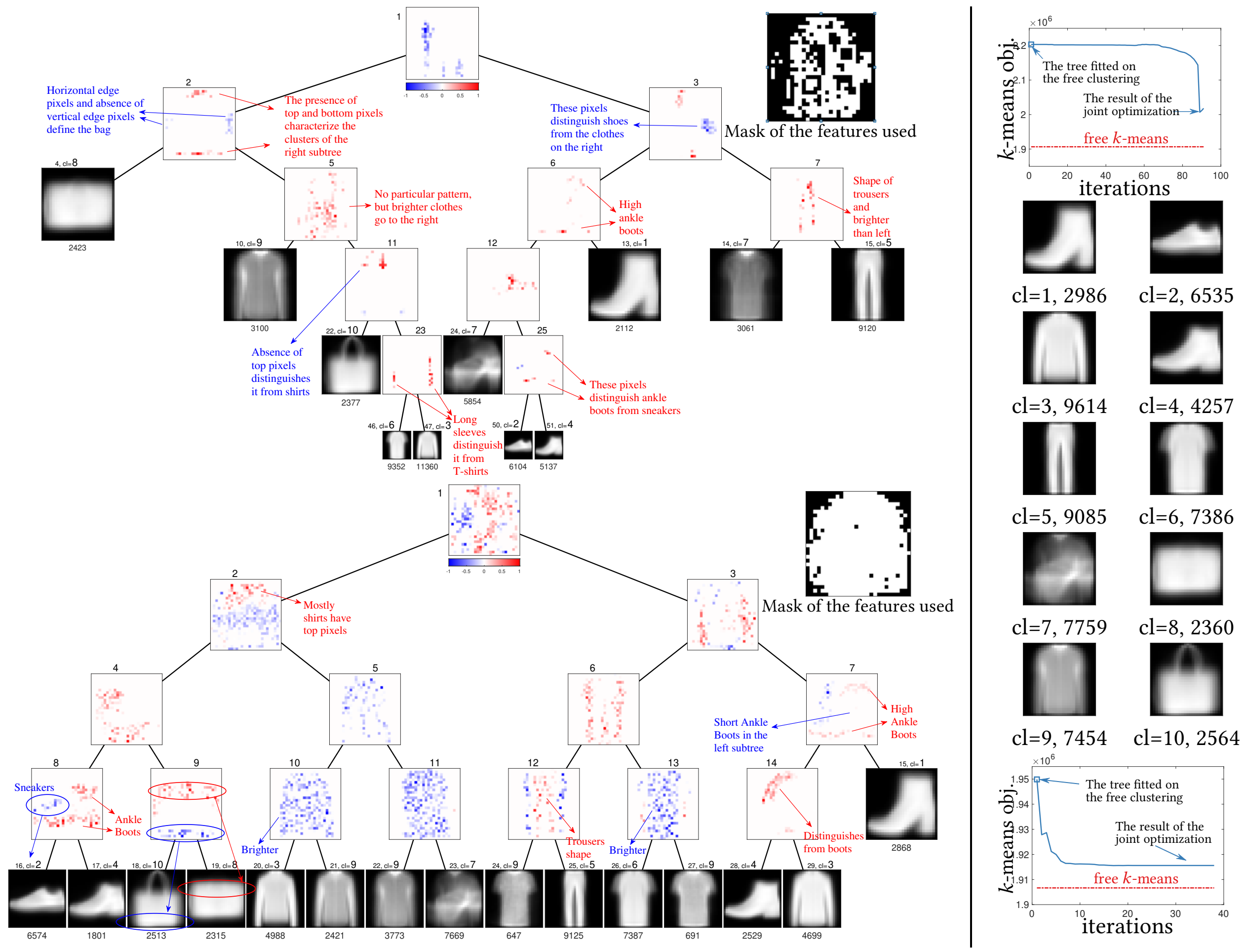
Horizontal edge pixels and absence of vertical edge pixels define the bag
The presence of top and bottom pixels characterize the clusters of the right subtree
These pixels distinguish shoes from the clothes on the right
Mask of the features used
No particular pattern, but brighter clothes go to the right
High ankle boots
Shape of trousers and brighter than left
Absence of top pixels distinguishes it from shirts
These pixels distinguish ankle boots from sneakers
Long sleeves distinguish it from T-shirts
Mostly shirts have top pixels
Mask of the features used
Short Ankle Boots in the left subtree
High Ankle Boots
Sneakers
Ankle Boots
Brighter
Trousers shape
Brighter
Distinguishes from boots
k-means obj.
The tree fitted on the free clustering
The result of the joint optimization
free k-means
iterations
cl=1, 2986
cl=2, 6535
cl=3, 9614
cl=4, 4257
cl=5, 9085
cl=6, 7386
cl=7, 7759
cl=8, 2360
cl=9, 7454
cl=10, 2564
k-means obj.
The tree fitted on the free clustering
The result of the joint optimization
free k-means
iterations

Figure 4: Visualization of trees resulted from the proposed optimization algorithm for FashionMNIST dataset with different sparsity parameters. The top tree results from $\lambda = 100.0,\ \Delta = 5$ and has a distortion of 5.22% from the free k-means clustering objective. The lower tree is obtained with $\lambda = 10.0,\ \Delta = 4$, and is only worse by 0.44% than the free clustering. The weight vector of a decision node is visualized as a 28×28 image. Red/blue pixels contribute sending points to the right/left. A leaf visualizes the mean of images that reach it. On the top right of the trees, we show the mask of all the features used by a tree. On the right we visualize the free clustering by plotting the mean image and the number of points. On the right we also plot the corresponding objective function during the run the algorithm.

M. Ranzato, A. Beygelzimer, P.S. Liang, J. W. Vaughan, and Y. Dauphin (Eds.), Vol. 34. 28929–28939.

[16] Badih Ghattas, Pierre Michel, and Laurent Boyer. 2017. Clustering Nominal Data Using Unsupervised Binary Decision Trees. *Pattern Recognition* 67, C (July 2017).

[17] Trevor J. Hastie, Robert J. Tibshirani, and Jerome H. Friedman. 2009. *The Elements of Statistical Learning—Data Mining, Inference and Prediction* (second ed.).

[18] Ruining He and Julian McAuley. 2016. Ups and Downs: Modeling the Visual Evolution of Fashion Trends with One-Class Collaborative Filtering. In *Proc. of the 25th (WWW'2016)*. 507–517.

[19] Kamran Kowsari, Donald E. Brown, Mojtaba Heidarysafa, Kiana Jafari Meimandi, Matthew S. Gerber, and Laura E. Barnes. 2017. HDLTex: Hierarchical Deep Learning for Text Classification. In *16th Int. Conf. Machine Learning and Applications (ICMLA)*. 364–371.

[20] Hans-Peter Kriegel, Peer Kröger, and Arthur Zimek. 2009. Clustering High-Dimensional Data: A Survey on Subspace Clustering, Pattern-Based Clustering, and Correlation Clustering. *ACM Trans. Knowledge Discovery from Data* 3, 1 (March 2009), 1.

[21] Eduardo Laber and Lucas Murtinho. 2021. On the Price of Explainability for Some Clustering Problems, See [25], 5915–5925.

[22] M. Lichman. 2013. UCI Machine Learning Repository. http://archive.ics.uci.edu/ml.

[23] Bing Liu, Yiyuan Xia, and Philip S. Yu. 2000. Clustering through Decision Tree Construction. In *9th ACM (CIKM 2000)*. 20–29.

[24] Konstantin Makarychev and Liren Shan. 2021. Near-Optimal Algorithms for Explainable k-Medians and k-Means, See [25], 7358–7367.

[25] Marina Meila and Tong Zhang (Eds.). 2021. *Proc. of the 38th (ICML 2021)*. Online.

[26] Michal Moshkovitz, Sanjoy Dasgupta, Cyrus Rashtchian, and Nave Frost. 2020. Explainable k-Means and k-Medians Clustering. In *Proc. of the 37th (ICML 2020)*, Hal Daumé III and Aarti Singh (Eds.). Online, 7055–7065.

[27] Jorge Nocedal and Stephen J. Wright. 2006. *Numerical Optimization* (second ed.).

[28] Lance Parsons, Ehtesham Haque, and Huan Liu. 2004. Subspace Clustering for High Dimensional Data: A Review. *SIGKDD Explorations* 6, (June 2004), 90–105.

[29] J. Ross Quinlan. 1993. *C4.5: Programs for Machine Learning*. Morgan Kaufmann.

[30] Weiran Wang and Miguel Á. Carreira-Perpiñán. 2014. The Laplacian K-Modes Algorithm for Clustering. (June 15 2014). arXiv:1406.3895.

[31] Arman Zharmagambetov and Miguel Á. Carreira-Perpiñán. 2022. Learning Interpretable, Tree-Based Projection Mappings for Nonlinear Embeddings. In *Proc. of the 25th (AISTATS 2022)*. Online, 9550–9570.

[32] Arman Zharmagambetov, Magzhan Gabidolla, and Miguel Á. Carreira-Perpiñán. 2021. Softmax Tree: An Accurate, Fast Classifier When the Number of Classes Is Large. In *Proc. Conf. (EMNLP 2021)*, Marie-Francine Moens, Xuanjing Huang, Lucia Specia, and Scott Wen-tau Yih (Eds.). Online, 10730–10745.

A REPRODUCIBILITY

A.1 Datasets

Amazon Reviews We select a small subset of text reviews from the large collection in [18][1]. We use the version from year 2014, and use the full reviews (listed under Per category files). To have some hierarchy of products, we select the reviews from the following categories: {Toys and Games: [Electronics for Kids, Arts & Crafts, Baby & Toddler Toys, Learning & Education], Beauty: [Skin Care: (Body, Face), Hair Care: (Styling Tools, Styling Products)], Grocery & Gourmet Food: [Candy Chocolate, Snack Foods, Breakfast, Beverages] }. From each category, except the "Breakfast", we select the longest (in terms of number of words) 4000 reviews. The "Breakfast" category has only 2358 reviews, from which we include all. We then extract unigrams from raw texts using scikit-learn's `CountVectorizer` class. For `stop_words` we use a built-in English stop word list, and we set `min_df=0.0002` and `max_df=0.2` to remove too frequent or very rare words. Then we apply tf-idf transformation using the `TfidfTransformer` class. We set the number of clusters equal to the number of categories (=12) in spherical k-means.

Web of Science Dataset of abstracts of scientific papers [19][2]. From the raw texts of 7 categories we extract the tf-idf feature vectors using the same procedure as in Amazon Reviews dataset. We set the number of clusters equal to the number of categories (=7) in spherical k-means.

MNIST and Fashion MNIST We use only the training set (N=60000) and scale pixel values to [0,1]. We set the number of clusters equal to the number of classes (=10).

FIFA19 The dataset of football player attributes from a popular simulation game FIFA 2019 [3]. We exclude the following features: {ID, Name, Photo, Nationality, Club, Club Logo, Special, Loaned From, Contract Valid Until, Release Clause, Flag, Position, Jersey Number, Body Type, Real Face, Joined}, because they do not describe the actual physical characteristics of a player. For categorical features we perform one-hot encoding. We then transform the features to make the mean 0 and variance 1 (feature-wise). We set the number of clusters to 10 (arbitrary choice).

Adult The dataset of 1994 Census participants available in the UCI Machine Learning repository [22]. We exclude the "fnlwgt" feature, as it does not really describe the person. We perform one-hot encoding of categorical features, and transform the features to have mean 0 and variance 1 (feature-wise). We set the number of clusters to 10 (arbitrary choice).

Letter English letter recognition task, available in the UCI Machine Learning Repository [22]. The features are edge counts and statistical moments. We transform the features to make mean 0 and variance 1 (feature-wise). We set the number of clusters equal to the number of classes (=26).

[1]https://jmcauley.ucsd.edu/data/amazon/
[2]https://data.mendeley.com/datasets/9rw3vkcfy4/6
[3]https://www.kaggle.com/karangadiya/fifa19

A.2 Implementation

To obtain the initial free k-means clustering, we use a scikit-learn's implementation. For spherical k-means, we implement it ourselves. We set the number of repeats `n_init=50`, and the number of iterations `max_iter=500` to obtain good reference clustering. For regular Euclidean k-means for initial centroid assignment we use k-means++, but for spherical k-means the initial centroids are selected randomly. The reference k-means clusters are the same for all the following decision trees.

TAO We implement TAO ourselves in Python. We initialize axis-aligned trees from ExKMC, but the oblique trees are initialized from a complete tree of depth Δ using bisecting k-means. Depending on the dataset, initial oblique trees have the following depths $\Delta \in \{4, 5, 6\}$. We consider several values of sparsity parameter $\lambda \in \{100, 10, 5, 1, 0.5, 0.1\}$ to obtain trees with different levels of complexity and cost. We estimate the starting value of μ_0 in the quadratic penalty schedule as the smallest value of μ needed to change the assignment of points in the $\mathbf{Z}$ step. Then we use the following μ schedule $\mu = \mu_0 \times 1.1^i$ for $i \in 0, 1, 2, \dots, 100$. The algorithm might terminate early if the penalty term vanishes. When fitting the initial tree (direct fit) we set the number of TAO iterations I=20. After that, at each μ step we just perform I=5 TAO iterations. Once we obtain the final tree, we prune the nodes that do not receive any points.

ExKMC The algorithm presented in [14] where a tree can have an arbitrary number of leaves. We use the authors implementation[4]. We set the `base_tree` to IMM, and consider the following number of leaves for tabular datasets: $\{k, 2k, 4k\}$, where k is the number of clusters. For other datasets we consider trees with the number of leaves up to 1024.

IMM Iterative Mistake Minimization algorithm from [26]. We use the authors implementation provided in ExKMC. We set the `base_tree` to IMM and `max_leaves` to the number of clusters k, as IMM only constructs trees with k leaves.

ExGreedy The algorithm from [21]. We use the authors implementation[5]. The algorithm constructs trees with exactly k leaves.

CART We use a scikit-learn's implemention of decision tree to fit the cluster labels of k-means as a classification problem. We do not restrict `max_depth` and control the tree size through `max_leaf_nodes` $\in \{k, 2k, 4k\}$ for tabular datasets, where k is the number of clusters. For other datasets we consider trees with `max_leaf_nodes` up to 1024.

B INTERPRETING SPHERICAL k-MEANS CLUSTERING FOR AMAZON REVIEWS DATASET

We select a subset of Amazon Reviews from 4 types of products of the following high level categories: Beauty, Food, and Toys. The details about the selection process can be found in the appendix A.1. In total our dataset has about 46k text reviews from 12 product categories. Having extracted tf-idf transformed bag-of-words

[4]https://github.com/navefr/ExKMC
[5]https://github.com/lmurtinho/ExKMC

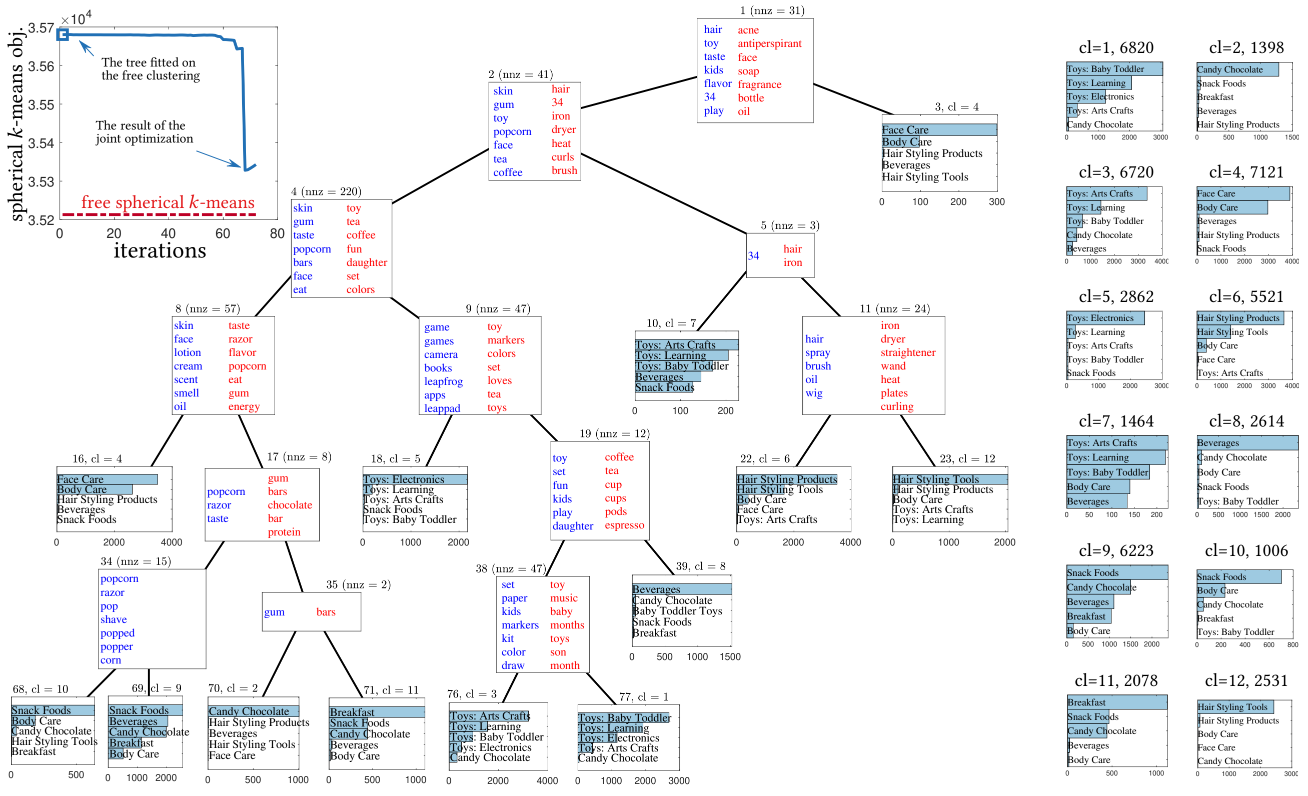
spherical k-means obj.
×10^4
The tree fitted on the free clustering
The result of the joint optimization
free spherical k-means
iterations
1 (nnz = 31)
hair toy taste kids flavor 34 play
acne antiperspirant face soap fragrance bottle oil
2 (nnz = 41)
skin gum toy popcorn face tea coffee
hair 34 iron dryer heat curls brush
3, cl = 4
4 (nnz = 220)
skin gum taste popcorn bars face eat
toy tea coffee fun daughter set colors
5 (nnz = 3)
34
hair iron
8 (nnz = 57)
skin face lotion cream scent smell oil
taste razor flavor popcorn eat gum energy
9 (nnz = 47)
game games camera books leapfrog apps leappad
toy markers colors set loves tea toys
10, cl = 7
11 (nnz = 24)
hair spray brush oil wig
iron dryer straightener wand heat plates curling
16, cl = 4
17 (nnz = 8)
popcorn razor taste
gum bars chocolate bar protein
18, cl = 5
19 (nnz = 12)
toy set fun kids play daughter
coffee tea cup cups pods espresso
22, cl = 6
23, cl = 12
34 (nnz = 15)
popcorn razor pop shave popped popper corn
35 (nnz = 2)
gum
bars
38 (nnz = 47)
set paper kids markers kit color draw
toy music baby months toys son month
39, cl = 8
68, cl = 10
69, cl = 9
70, cl = 2
71, cl = 11
76, cl = 3
77, cl = 1
cl=1, 6820
cl=2, 1398
cl=3, 6720
cl=4, 7121
cl=5, 2862
cl=6, 5521
cl=7, 1464
cl=8, 2614
cl=9, 6223
cl=10, 1006
cl=11, 2078
cl=12, 2531
Face Care
Body Care
Hair Styling Products
Beverages
Hair Styling Tools
Snack Foods
Candy Chocolate
Breakfast
Toys: Arts Crafts
Toys: Learning
Toys: Baby Toddler
Toys: Electronics
Baby Toddler Toys

Figure 5: Visualization of a tree resulted from the proposed algorithm for spherical k-means objective for the subset of Amazon Reviews dataset. On the top left we plot the curve of the objective function during the algorithm run. We show histograms (horizontally) of the product categories of the clusters obtained by free spherical k-means on the right column. We visualize decision nodes by showing the words corresponding to the most positive/negative top 7 weights in red/blue. The words are sorted by the weight magnitude with higher weights appearing on top. We report the number of nonzero weights (nnz) at the top of a decision node. At a leaf we plot the histogram of product categories of the points reaching that leaf.

features, we set the number of clusters to 12 and run the free spherical k-means and visualize the clusters by plotting the histograms in the right column of fig. 5. In general the clusters are dominated by products from a single category or a few similar categories with the exception of cluster 7, which has products mostly from Toys, but also from Body Care and Beverages. We run our joint optimization approach for spherical k-means objective with an oblique tree of depth $\Delta = 6$ and $\lambda = 5$ initialized randomly. As the curve on the top left plot of fig. 5 shows, the tree considerably improves over the direct fit, and has a distortion of only 0.29% over the free clustering, and uses on average only 42 features at a decision node. Plotting the words corresponding to the most significant weights (both positive and negative) we can visualize the tree in fig. 5. We can notice some hierarchical structure of the nodes with leaves of similar cluster categories appearing together. For example, the children of a decision node 11 are about hair care products, where one is dominated by styling products and the other by styling tools, and similarly for the children of decision nodes 34, 35, and 38. The cluster 7 with mixed product categories are predicted by a leaf numbered 10. Strangely, the parent of it uses a single word "34" to distinguish this cluster. A closer look at the text reviews in cluster 7 reveals that it contains documents with the word "&#34;". This is a decimal encoding of a quotation mark in Unicode format. Because of using stop words in bag-of-words feature extraction, "34" appears as a significant word characterizing these documents. This case clearly illustrates the value of an interpretable model such as a decision tree in debugging or finding artifacts in data, which is not in general possible from the free clustering result.