Abstract. In this paper, we initiate a theoretical study of the problem of clustering data under interactive feedback. We introduce a query-based model in which users can provide feedback to a clustering algorithm in a natural way via split and merge requests. We then analyze the “clusterability” of different concept classes in this framework — the ability to cluster correctly with a bounded number of requests under only the assumption that each cluster can be described by a concept in the class — and provide efficient algorithms as well as information-theoretic upper and lower bounds.

1 Introduction

Clustering is often a highly under-specified problem: given a set of data items, there may be many different possible clusterings a user might be interested in. For instance, given a set of documents or news articles, should all those about sports go into a single cluster or should there be different clusters for football, baseball, hockey and so on? Should articles on salaries paid to sports figures be classified under sports or under business? Or perhaps, completely orthogonally, the user wants articles clustered by length or by writing style and not by topic. Most theoretical work “wishes away” this under-specification by making strong distributional assumptions, such as the data distribution being a mixture of Gaussians with each Gaussian as one of the clusters (e.g., [9, 3]). In this work, we instead embrace the idea that a given set of data might have multiple plausible clusterings, and consider the problem of clustering under feedback. That is, we imagine users are willing to help a clustering algorithm arrive at their own desired answer with a small amount of additional prodding, and ask what kinds of algorithms can take advantage of such feedback and under what conditions can they succeed. In this paper, we consider the problem of clustering under a quite natural type of feedback in the form of split and merge requests. Specifically, given a proposed clustering of the dataset, the user responds with either:

Split: The user identifies a cluster \( c \) in the algorithm’s clustering that contains points from multiple target clusters (and therefore should be split), or

Merge: The user identifies two clusters \( c, c' \) in the algorithm’s clustering that are both subsets of the same target cluster and therefore should be merged.
Or, if neither of these applies, the user responds that the algorithm’s clustering is correct. Thus, this model is similar to the standard model of learning with equivalence queries [10], except that rather than return a misclassified data point, the user instead responds with the more vague request that, say, some cluster should be split (but without saying how that split should be done, or where in that cluster a mistake was made). We then show, perhaps surprisingly, that a number of interesting positive results can be had in this model under only the assumption that each cluster in the target is a member of some given concept class $C$, without any distributional assumptions on the data. In contrast, as mentioned above, most work on clustering has focused on generative models, such as mixtures of Gaussian or logconcave distributions, in which the underlying data distribution is effectively committing to only a single answer [1, 8, 7, 9, 14, 12].

Our model can be illustrated by the following simple example. Suppose our given dataset $S$ consists of $m$ points on the real line. We are told that each cluster is an interval, and let us say for simplicity of discussion we also are told there are only $k \leq 2$ clusters. In this case we could begin by proposing a single cluster with all the points and proposing it to the user. If we are incorrect, the user will respond with a split request, in which case we split the cluster into two intervals of $m/2$ points each and present the result to the user again. In general, if the user asks us to split a cluster, we partition it exactly in half (by cardinality), and if the user asks us to merge two clusters, we merge them. Since at most one of the algorithm’s intervals can contain points from both of the user’s intervals, and since each split request causes the number of points in the offending interval to drop by a factor of 2, the total number of split requests is at most $\log m$. Therefore, the total number of merge requests is at most $\log m$ as well, and so the overall number of requests at at most $2\log m$.

Note that clustering in this model with $m$ requests is trivial for any concept class $C$: just begin with each point in its own cluster and merge as requested. So, our goal will be to develop algorithms whose query complexity is logarithmic in $m$ and polynomial in parameters such as $\log |C|$ and (ideally) $k$. Note also that if we strengthened the model to allow the algorithm to specify which cluster the user should focus on, then we could simulate membership queries [2, 11]; indeed, one of the key difficulties in our model will be designing algorithms that can make progress no matter which clusters are asked to be split or merged.

The main results we show in this model are as follows (here, $m$ is the total number of data points and $k$ is the number of clusters in the target):

1. For the case of points on the line and the class of intervals, we give a simple algorithm that requires only $O(k \log m)$ requests to cluster correctly.

---

1 Suppose inductively we have determined points $x_1, \ldots, x_{k'}$ that are all known to be in different clusters. Then, in this strengthened model, given a new point $x$ we could query the sequence of clusters $\{x, x_1\}, \{x, x_2\}, \ldots$. The first one of these that does not produce a split request is the label of $x$ (or if all produce a split request, we assign $x$ to label $k' + 1$).