Interactive Clustering of Linear Classes and Cryptographic Lower Bounds

Ádám D. Lelkes(✉) and Lev Reyzin

Department of Mathematics, Statistics, and Computer Science,
University of Illinois at Chicago, Chicago, IL 60607, USA
{alelke2,lreyzin}@uic.edu

Abstract. We study an interactive model of supervised clustering introduced by Balcan and Blum [7], where the clustering algorithm has query access to a teacher. We give an efficient algorithm clustering linear functionals over finite fields, which implies the learnability of parity functions in this model. We also present an efficient clustering algorithm for hyperplanes which are a natural generalization of the problem of clustering linear functionals over $\mathbb{R}^d$. We also give cryptographic hardness results for interactive clustering. In particular, we show that, under plausible cryptographic assumptions, the interactive clustering problem is intractable for the concept classes of polynomial-size constant-depth threshold circuits, Boolean formulas, and finite automata.

Keywords: Interactive clustering · Query learning · Parity function · Cryptographic lower bounds

1 Introduction

In this paper we consider the interactive clustering model proposed by Balcan and Blum [7]. This clustering (and learning) model allows the algorithm to issue proposed explicit clusterings to an oracle, which replies by requesting two of the proposed clusters "merge" or that an impure cluster be "split". This model captures an interactive learning scenario, where one party has a target clustering in mind and communicates this information via these simple requests.

Balcan and Blum [7] give the example of a human helping a computer cluster news articles by topic by indicating to the computer which proposed different clusters are really about the same topic and which need to be split. Another motivating example is computer-aided image segmentation, where an algorithm can propose image segmentations to a human, who can show the computer which clusters need to be "fixed up" – this is likely to be much simpler than having the human segment the image manually.

Many interesting results are already known for this model [5,7], including the learnability of various concept classes and some generic, though inefficient, algorithms (for an overview, see Sect. 3).

In this paper we extend the theory of interactive clustering. Among our main results:

K. Chaudhuri et al. (Eds.): ALT 2015, LNAI 9355, pp. 165–176, 2015.
DOI: 10.1007/978-3-319-24486-0_11

- We show efficient algorithms for clustering parities and, more generally, linear functionals over finite fields – parities are a concept class of central importance in most models of learning. (Section 4)
- We also give an efficient algorithm for clustering hyperplanes, a generalization of linear functionals over $\mathbb{R}^d$. These capture a large and important set of concept classes whose efficient clusterability was not known in this model. (Section 5)
- We prove lower bounds for the interactive clustering model under plausible cryptographic assumptions, further illustrating the richness of this model. (Section 6)

2 The Model

In this section we describe the interactive clustering model of Balcan and Blum [7]. In this model of clustering, no distributional assumptions are made about the data; instead, it is assumed that the teacher knows the target clustering, but it is infeasible for him to label each data point by hand. Thus the goal of the learner is to learn the target clustering by making a small number of queries to the teacher. In this respect, the model is similar to the foundational query learning models introduced by Angluin [2]. (As a consequence, the classes we consider in this paper might be more familiar from the theory of query learning than from the usual models of clustering.)

More specifically, the learner is given a sample S of m points, and knows the number of target clusters which is denoted as k. The target clustering is an element of a concept class C. In each round, the learner presents a hypothesis clustering to the teacher. The answer of the teacher to this query is one of the following: either that the hypothesis clustering is correct, or a `split` or `merge` request. If this hypothesis is incorrect, that means that at least one of the following two cases has to hold: either there are *impure* hypothesis clusters, i.e. hypothesis clusters which contain points from more than one target cluster, or there are more than one distinct hypothesis clusters that are subsets of the same cluster. In the first case, the teacher can issue a `split` request to an impure cluster, in the second case the teacher can issue a `merge` request to two clusters that are both subsets of the same target cluster. If there are several valid possibilities for `split` or `merge` requests, the teacher can arbitrarily choose one of them.

Definition 1. *An interactive clustering algorithm is called* ***efficient*** *if it runs in* $O(\text{poly}(k, m, \log|C|))$ *time and makes* $O(\text{poly}(k, \log m, \log|C|))$ *queries.*

Observe that allowing the learner to make m queries would make the clustering task trivial: by starting from the all singleton hypothesis clustering and merging clusters according to the teacher's requests, the target clustering can be found in at most m rounds.

3 Previous Work

Extensive research on clustering has yielded a plethora of important theoretical results, including traditional hardness results [17,18], approximation algorithms [3,4,9,13,16,20], and generative models [12,14]. More recently researchers have examined properties of data that imply various notions of "clusterability" [1]. An ongoing research direction has been to find models that capture real-world behavior and success of clustering algorithms, in which many foundational open problems remain [11].

Inspired by models where clusterings satisfy certain natural relations with the data, e.g. [6], Balcan and Blum [7] introduced the notion of interactive clustering we consider in this paper – the data assumption here, of course, is that a "teacher" has a clustering in mind that the data satisfies, while the algorithm is aware of the space of possible clusterings.

In addition to defining the interactive clustering model, Balcan and Blum [7] gave some of the first results for it. In particular, they showed how to efficiently cluster intervals, disjunctions, and conjunctions (the latter only for constant k). Moreover, they gave a general, but inefficient, version space algorithm for clustering any finite concept class using $O(k^3 \log |C|)$ queries. They also gave a lower bound that showed efficient clustering was not possible if if the algorithm is required to be proper, i.e. produce k-clusterings to the teacher. These results contrast with our cryptographic lower bounds, which hold for arbitrary hypothesis clusterings.

Awasthi and Zadeh [5] later improved the generic bound of $O(k^3 \log |C|)$ to $O(k \log |C|)$ queries using a simpler version space algorithm. They presented an algorithm for clustering axis-aligned rectangles.

Awasthi and Zadeh [5] also introduced a noisy variant of this model. In the noisy version, `split` requests are still only issued for impure clusters, but `merge` requests might have "noise": a `merge` request might be issued if at least an η fraction of the points from both hypothesis clusters belong to the same target cluster. Alternatively, a stricter version of the noisy model allows arbitrary noise: the teacher might issue a `merge` request for two clusters even if they both have only one point from some target cluster. Awasthi and Zadeh [5] gave an example of a concept class that cannot be learned with arbitrary noise, and presented an algorithm for clustering intervals in the η noise model. To the best of our knowledge, our algorithm for clustering linear functionals over finite fields, presented in Sect. 4, is the first algorithm for clustering a nontrivial concept class under arbitrary noise.

Other interactive models of clustering have, of course, also been considered [10,15]. In this paper, however, we keep our analysis to the Balcan and Blum [7] interactive model.

4 Clustering Linear Functionals

In this section we present an algorithm for clustering linear functionals over finite fields. That is, the instance space is $X = GF(q)^n$ for some prime power

q and positive integer n, where $GF(q)$ denotes the finite field of order q. The concept class is the dual space $(GF(q)^n)^*$ of linear operations mapping from $GF(q)^n$ to $GF(q)$. Thus the number of clusters is $k = q$. Recall that every linear functional in $(GF(q)^n)^*$ is of the form $v \mapsto x \cdot v$, thus clustering linear functionals is equivalent to learning this unknown vector x. For the special case of $q = 2$, we get the concept class of parity functions over $\{0,1\}^n$, where there are two classes/clusters (for the positively and negatively labeled points).

The idea of the algorithm is the following: in each round we output the largest sets of elements that are already known to be pure, thus forcing the teacher to make a `merge` request. A `merge` request for two clusters will yield a linear equation for the target vector which is independent from all previously learned equations. We use a graph on the data points to keep track of the learned linear equations. Since the algorithm learns a independent equation in each round, it finds the target vector in at most n rounds. The description of the algorithm follows.

Algorithm 1. Cluster-Functional

initialize $G = (V, \emptyset)$, with $|V| = m$, each vertex corresponding an element from the sample.
initialize $Q = \emptyset$.
repeat
  find the connected components of G and output them as clusters.
  on a `merge` request to two clusters:
  for each pair a, b of points in the union **do**
    if $(a - b) \cdot x = 0$ is independent from all equations in Q **then**
      add $(a - b) \cdot x = 0$ to Q.
    end if
  end for
  for each non-edge (a, b), add (a, b) to G if $(a - b) \cdot x = 0$ follows from the equations in Q.
until the target clustering is found

Theorem 1. *Algorithm 1 finds the target clustering using at most n queries and $O(m^2 n^4)$ time. Moreover, the query complexity of the algorithm is optimal: every clustering algorithm needs to make at least n queries to find the target clustering.*

Proof. We claim that in each round we learn a linear equation that is independent from all previously learned equations, thus in n rounds we learn the target parity.

Assume for contradiction that there is a round where no independent equations are added. All hypothesis clusters are pure by construction so they can never be split. If two clusters are merged, then let us pick an element a from one of them and b from the other. Then $(a - b) \cdot x = 0$ has to be independent from Q

since otherwise the edge (a, b) would have been added in a previous round and the two elements would thus belong to the same cluster.

Thus after at most n rounds G will consist of two marked cliques which will give the correct clustering. Finding the connected components and outputting the hypothesis clusters takes linear time. To update the graph, $O(m^2)$ Gaussian elimination steps are needed. Hence the total running time is $O(m^2n^4)$.

To show that at least n queries are necessary, notice that `merge` and `split` requests are equivalent to linear equations and inequalities, respectively. Since the dimension of the dual space is n, after less than n queries there are at least two linearly independent linear functionals, and therefore at least two different clusterings, that are consistent with all the queries. □

Observe that for $q > 2$ this is in fact an efficient implementation of the generic halving algorithm of Awasthi and Zadeh [5]. Every subset of element is either known to be pure, in which case it is consistent with the entire version space, or is possible impure, in which case a `split` request would imply that the target vector satisfies a disjunction of linear equations. Thus in the latter case the set is consistent with at most a $\frac{1}{q} < \frac{1}{2}$ fraction of the version space.

There are two other notable properties of the algorithm. One is that it works without modification in the noisy setting of Awasthi and Zadeh [5]: if any pair of elements from two pure sets belong to the same target cluster, then it follows immediately by linearity that both sets are subsets of this target cluster.

The other notable property is that the algorithm never outputs impure hypothesis clusters. This is because it is always the case that every subset of the sample is either known to be pure, or otherwise it is consistent with at most half of the version space. Any concept class that has a similar gap property can be clustered using only pure clusters in the hypotheses. The following remark formalizes this idea.

Remark 1. Consider the following generic algorithm: in each round, output the maximal subsets of S that are known to be pure, i.e. are consistent with the entire version space. The teacher cannot issue a `split` request since every hypothesis cluster is pure. If there is an $\varepsilon > 0$ such that in each round every subset $h \subseteq S$ of the sample is consistent with either the entire version space or at most a $(1-\varepsilon)$ fraction of the version space, then on a `merge` request, by the maximality of the hypothesis clusters, we can eliminate an ε fraction of the version space. Therefore this algorithm finds the target clustering after $k \log_{\frac{1}{1-\varepsilon}} |C|$ queries using only pure clusters in the hypotheses.

5 Efficient Clustering of Hyperplanes

Now we turn to a natural generalization of linear functions over $\mathbb{R}^d$, k hyperplanes. Clustering geometric concept classes was one of the proposed open problems by Awasthi and Zadeh [5]; hyperplanes are an example of a very natural

geometric concept class. The data are points in $\mathbb{R}^d$ and they are clustered $(d-1)$-dimensional affine subspaces. Every point is assumed to lie on exactly one of k hyperplanes.

First, observe that this is a nontrivial interactive clustering problem: even for $d = 2$ the cardinality of the concept class can be exponentially large as a function of k. For example, let k be an odd integer, and consider $m - 2(k-1)$ points on a line and $2(k-1)$ points arranged as vertices of n squares such that no two edges are on the same line. Then it is easy to see that the number of different possible clusterings is at least 3^k. Hence, if $k = \omega(\text{polylog}(m))$, the target clustering cannot be efficiently found by the halving algorithm of Awasthi and Zadeh [5]: since the cardinality of the initial version space is superpolynomial in m, the algorithm cannot keep track of the version space in polynomial time.

Nevertheless, the case of $d = 2$ can be solved by the following trivial algorithm: start with the all-singleton hypothesis, and on a `merge` request, merge all the points that are on the line going through the two points. This algorithm will find the target clustering after k queries. However, this idea does not even generalize to $d = 3$: the teacher might repeatedly tell the learner to merge pairs of points that define parallel lines. In this case, it is not immediately clear which pairs of lines span the planes of the target clustering, and there can be a linear number of such parallel lines.

On the other hand, in the case of $d = 3$, coplanar lines either have to be in the same target cluster, or they all have to be in different clusters. Therefore if we have $k + 1$ coplanar lines, by the pigeonhole principle we know that the plane containing them has to be one of the target planes. Moreover, since all points are clustered by the k planes, it follows by the pigeonhole principle that after $k^2 + 1$ `merge` requests for singleton pairs we will get $k + 1$ coplanar lines. This observation gives an algorithm of query complexity $O(k^3)$, although it is not immediately clear how the coplanar lines can be found efficiently.

Algorithm 2, described below, is an efficient clustering algorithm based on a similar idea which works for arbitrary dimension.

Algorithm 2. Cluster-Hyperplanes

```
let H = S.
for i = 1 to d − 1 do
  for each affine subspace F of dimension i do
    if at least k^i + 1 elements of H are subsets of F then
      replace these elements in H by F.
    end if
  end for
end for
repeat
  output elements of H as hypothesis clusters.
  on a merge request, merge the two clusters in H.
until the target clustering is found
```

Theorem 2. *Algorithm 2 finds the target clustering using at most* $O(k^{d+1})$ *queries and* $O(d \cdot m^{d+1})$ *time.*

Proof. We claim that in each iteration of the for loop, it holds for every F that every subset of $k^{i-1}+1$ elements of H that lie on F spans F. The proof is by induction. For $i = 1$ this is clear: all pairs of points on a line span the line. Assume that the claim hold for $i-1$. Consider $k^{i-1}+1$ elements of H on an affine subspace F of dimension i. If they spanned an affine subspace of dimension less than i, then they would have been merged in a previous iteration. Hence they have to span F.

Now if $k^i + 1$ elements of H line on an i-dimensional affine subspace F for $i < d$, then they have to be in the same target cluster. If they were not, no hyperplane could contain more than k^{i-1} of the elements, and therefore the k target hyperplanes could cover at most k^i elements contained by F, which contradicts the assumption that all points belong to a target cluster.

Hence, at the start of the repeat loop there can be at most k^{d+1} elements in H: if there were more than $k^{d+1}+1$ elements in H, by the pigeonhole principle there would be a target cluster containing $k^d + 1$ of them. However, this is not possible since those $k^d + 1$ elements would have been merged previously.

Therefore in the repeat loop we only need k^{d+1} queries to find the target clustering. In each iteration of the outer for loop, we have to consider every affine subspace of a certain dimension. Since every at most $(d-1)$-dimensional subspace is defined by d points, there are at most $\binom{m}{d}$ subspaces. For each of them, we have to count the elements that are contained by them, this takes m time. Thus the total running time is $O\left(d \cdot \binom{m}{d} \cdot m\right) = O(d \cdot m^{d+1})$. □

Hence, for constant d, this is an efficient clustering algorithm.

6 Cryptographic Lower Bounds for Interactive Clustering

In this section, we show cryptographic lower bounds for interactive clustering. In particular, we prove that, under plausible cryptographic assumptions, the class of constant-depth polynomial-size threshold circuits and polynomial-size Boolean formulas are not learnable in the interactive clustering model. These lower bounds further go to show the richness of this model, which allows for both positive and negative clusterability results.

It was first observed by Valiant [24] that the existence of certain cryptographic primitives implies unlearnability results. Later, Kearns and Valiant [19] showed that, assuming the intractability of specific problems such as inverting the RSA function, some natural concept classes, for example the class of constant-depth threshold circuits, are not efficiently PAC learnable.

The hardness results for PAC learning are based on the following observation: if f is a trapdoor one-way function, and there is an efficient learning algorithm which, after seeing polynomially many labeled examples of the form $(f(x), x)$, can predict the correct label $f^{-1}(y)$ of a new unlabeled data point y, then that learning algorithm by definition breaks the one-way function f.

This observation doesn't apply to interactive clustering since here the learner doesn't have to make predictions about new examples and the teacher can give information about any of the elements in the sample. Indeed, if the learner were allowed to make a linear number of queries to the teacher, the clustering task would be computationally trivial. Instead, our proofs are based on the following counting argument: if the concept class is exponentially large in the size of the sample, then there is an immediate information-theoretic exponential lower bound on the required number of queries; therefore on average a learner would have to make an exponential number of queries to learn a randomly chosen clustering. If there exist certain pseudorandom objects, then one can construct concept classes of subexponential size such that a randomly chosen concept from the smaller class is computationally indistinguishable from a randomly chosen concept from the exponential-size class. However, on the smaller concept class the learner is only allowed to make a subexponential number of queries; consequently, this smaller class is not efficiently learnable.

First let us recall some relevant definitions.

A *one-way function* is a function $f : \{0,1\}^* \to \{0,1\}^*$ such that for every polynomial time algorithm A, every positive integer α, and every sufficiently large integer n it holds that

$$\Pr(f(A(f(x))) = f(x)) < n^{-\alpha}$$

where x is chosen uniformly at random from $\{0,1\}^n$.

A *pseudorandom function family* of hardness $h(n)$ is a sequence F_n of sets of efficiently computable functions $A_n \to B_n$ such that for every $h(n)$ time-bounded probabilistic algorithm A with oracle access to a function $A_n \to B_n$ and every $\alpha > 0$ it holds that

$$|\Pr(A^{f_n}(1^n) = 1) - \Pr(A^{r_n}(1^n) = 1)| < n^{-\alpha}$$

where f_n and r_n are chosen uniformly randomly from F_n and the set of all functions $A_n \to B_n$, respectively; and there is a probabilistic poly(n)-time algorithm that on input 1^n returns a uniformly randomly chosen element of F_n.

In this paper, we will pick $A_n = \{0,1\}^n$ and $B_n = \{0,1\}$. Sometimes it is useful to consider *keyed* pseudorandom functions, i.e. pseudorandom function families where $F_n = \{f_K : K \in \{0,1\}^n\}$ (and the sampling algorithm simply chooses a uniform $K \in \{0,1\}^n$ and returns f_K). The existence of one-way functions implies the existence of pseudorandom function families of polynomial hardness.

We will use the following information-theoretic lower bound to prove our hardness result.

Lemma 1. *For $k = 2$, every clustering algorithm has to make at least $\Omega\left(\frac{\log |C|}{\log m}\right)$ queries to find the target clustering.*

Proof. There are $\log |C|$ bits are needed to describe the clustering. To each query, the answer is `split` or `merge` and the identifier of at most two clusters. Since

there are at most m clusters in any hypothesis, this means that the teacher gives at most $2\log m + 1$ bits of information per query. Thus the required number of queries is $\Omega\left(\frac{\log|C|}{\log m}\right)$. □

We remark that Theorem 9 of Balcan and Blum [7] implies a worst-case lower bound of $\Omega(\log|C|)$. However, this weaker bound of $\Omega\left(\frac{\log|C|}{\log m}\right)$ holds for teachers that are not necessarily adversarial.

As we noted above, the existence of pseudorandom function families that can fool any polynomial time-bounded distinguishers is implied by the existence of one-way functions. Unfortunately, this hardness does not seem enough to imply a lower bound for interactive clustering for the following reason. If we take a sample of size m from $\{0,1\}^n$, then if $m = O(\mathrm{poly}(n))$, the learner is allowed to make m queries which makes the clustering problem trivial. On the other hand, if m is superpolynomial in n, the learner is allowed to take superpolynomial time, therefore it might break pseudorandom functions that can only fool polynomial-time adversaries.

However, if there exist pseudorandom functions that can fool distinguishers that have slightly superpolynomial time, a hardness result for interactive clustering follows. Candidates for pseudorandom functions or permutations used in cryptographic practice are usually conjectured to have this property.

Theorem 3. *If there exist strongly pseudorandom permutations that can fool distinguishers which have $n^{\omega(1)}$ time, then there exists a concept class C which is not learnable in the interactive clustering model with $\mathrm{poly}(\log m, \log|C|)$ queries and $\mathrm{poly}(m, \log C)$ time.*

Proof. Let $f_K : \{0,1\}^n \to \{0,1\}$ be a keyed pseudorandom function that can fool distinguishers which have $t(n)$ time for some time-constructible $t(n) = n^{\omega(1)}$. Without loss of generality, assume that $t(n) = o(2^n)$. Let us fix a time-constructible function $m(n)$ such that $m(n) = n^{\omega(1)}$ and $\mathrm{poly}(m(n)) = o(t(n))$. Let S be a subset of $\{0,1\}^n$ of cardinality $m = m(n)$ and let $k = 2$. Let U_n be a set of all functions $\{0,1\}^n \to \{0,1\}$, $F_n = \{f_K : K \in \{0,1\}^n\}$.

Let us assume for contradiction that there is an efficient interactive clustering algorithm A for the concept class $C = F_n$. Since $|C| = 2^n$, this learner has to make at most $\mathrm{poly}(n, \log m(n)) = \mathrm{poly}(n)$ queries and has $\mathrm{poly}(n, m(n)) = \mathrm{poly}(m(n))$ time. Let us assume that the learner finds the target clustering after $O(n^\alpha)$ queries.

Let B be the following algorithm: given oracle access to a function $f : \{0,1\}^n \to \{0,1\}$, pick a sample S of size $m = m(n)$ from $\{0,1\}^n$, label the sample vectors according to the value of f, and simulate the learner A for at most $n^{\alpha+1}$ queries. Accept if the learner finds the target clustering and reject otherwise.

Since $\mathrm{poly}(m(n)) = o(t(n))$, B runs in time $t(n)$. If f is chosen from F_n, B will accept with probability 1. On the other hand, if f is chosen from U_n, then since $|U_n| = 2^{2^n}$, by Lemma 1, we have a query lower bound of $\frac{\log|U_n|}{\log m} = \frac{2^n}{\log m(n)} = \omega(n^{\alpha+1})$. Therefore after $n^{\alpha+1}$ queries there are at least two different

clusterings in the version space, therefore B will reject with probability at least $\frac{1}{2}$. This contradicts the $t(n)$-hardness of f_K. □

Naor and Reingold [22] constructed pseudorandom functions with one-bit output that are not only as secure as factoring Blum integers, but also computable by TC^0 circuits. Since $\log |TC^0| = \text{poly}(n)$, this, together with Theorem 3, implies the following corollary:

Corollary 1. *If factoring Blum integers is hard for $h(n)$-time bounded algorithms for some $h(n) = n^{\omega(1)}$ then the class TC^0 of constant-depth polynomial-size threshold circuits and the class of polynomial-size Boolean formulas are not learnable in the interactive clustering model.*

Proof. By Theorem 3, learning a pseudorandom function family of superpolynomial hardness is hard in the interactive clustering model. If factoring Blum integers is superpolynomially hard, then by the construction of Naor and Reingold [22], TC^0 contains such a pseudorandom function family. Furthermore, $\log |TC^0| = \text{poly}(n)$, the learner is still only allowed to have $\text{poly}(n, \log m)$ queries and $\text{poly}(n, m)$ time, therefore the Theorem 3 also applies to TC^0. In fact, this holds for TC^0 circuits of size at most n^α for some constant α (determined by the size of the circuits implementing the pseudorandom function). The set of languages computable TC^0 circuits of size n^α is in turn a subset of the languages computable by Boolean formulas of size at most n^β for some other constant β. Thus our cryptographic lower bound also holds for polynomial-sized Boolean formulas. □

Remark 2. After Naor and Reingold's first construction of pseudorandom functions in TC^0, several others showed that it is possible to construct even more efficient PRFs, or PRFs based on different, possibly weaker cryptographic assumptions. For example, we refer the reader to the work of Lewko and Waters [21] for a construction under the so-called "decisional k-linear assumption" which is weaker than the assumption of Naor and Reingold [22], or to Banerjee et al. [8] for a construction based on the "learning with errors" problem, against which there is no known attack by efficient quantum algorithms.

Kearns and Valiant [19] used the results of Pitt and Warmuth [23] about prediction-preserving reductions to show that in the PAC model, their cryptographic hardness result for NC^1 circuits also implies the intractability of learning DFAs. Despite the fact the problem of interactive clustering is fundamentally different from prediction problems, we show that the ideas of Pitt and Warmuth [23] can be applied to show that DFAs are hard to learn in this model as well. We use the following theorem:

Theorem 4 (Pitt and Warmuth [23]). *Let k be a fixed positive constant. If T is a single-tape Turing machine of size at most s that runs in space at most $k \log n$ on inputs of length n, then there exist polynomials p and q such that for all positive integers n there exists a DFA M of size $q(s, n)$ such that M accepts $g(w) = 1^{|w|}0w^{p(|w|,s,n)}$ if and only if T accepts w.*

This theorem implies a hardness result for interactive clustering.

Corollary 2. *If there are $n^{\omega(1)}$-hard pseudorandom function families computable in logarithmic space, then polynomial-size DFAs are not efficiently learnable in the interactive clustering model.*

Proof. Let $f_K : \{0,1\}^n \to \{0,1\}$ be an $n^{\omega(1)}$-hard keyed pseudorandom function. If $S \subset \{0,1\}^n$ has cardinality $m(n)$ as defined in Theorem 3 and the concept class is $\{f_K : K \in \{0,1\}\}^n$, the interactive clustering task is hard.

For all $K \in \{0,1\}^n$, let T_K be a Turing machine of size at most s that runs in space $k \log n$ and, given w as an imput, computes $f_K(w)$. It is easy to see that there exists functions g, p and q defined as in Theorem 4 that work for T_K for all K. Consider the sample $S' = g(S)$ and the concept class C of DFAs of size $q(s,n)$. Since $|S'| = m(n)$ and $\log |C| = \text{poly}(n)$, the hardness result of Theorem 3 holds here as well. □

7 Conclusion

In this paper we studied a model of clustering with interactive feedback. We presented efficient clustering algorithms for linear functionals over finite fields, of which parity functions are a special case, and hyperplanes in $\mathbb{R}^d$, thereby showing that these two natural problems are learnable in the model. On the other hand, we also demonstrated that under a standard cryptographic assumptions, constant-depth polynomial-size threshold circuits, polynomial-size Boolean formulas, and polynomial-size deterministic finite automata are not learnable.

We propose the following open problems.

1. It would be interesting to see if the exponential dependence on d in the complexity of Algorithm 2 for clustering hyperplanes can be reduced.
2. Clustering half-spaces remains a natural and important open problem.

References

1. Ackerman, M., Ben-David, S.: Clusterability: A theoretical study. In: Proceedings of the Twelfth International Conference on Artificial Intelligence and Statistics, AISTATS 2009, Clearwater Beach, Florida, USA, April 16–18, pp. 1–8 (2009)
2. Angluin, D.: Queries and concept learning. Machine Learning **2**(4), 319–342 (1988)
3. Arora, S., Raghavan, P., Rao, S.: Approximation schemes for euclidean-medians and related problems. In: STOC, pp. 106–113 (1998)
4. Arya, V., Garg, N., Khandekar, R., Meyerson, A., Munagala, K., Pandit, V.: Local search heuristics for k-median and facility location problems. SIAM J. Comput. **33**(3), 544–562 (2004)
5. Awasthi, P., Zadeh, R.B.: Supervised clustering. In: Advances in Neural Information Processing Systems, pp. 91–99 (2010)
6. Balcan, M., Blum, A., Vempala, S.: A discriminative framework for clustering via similarity functions. In: Proceedings of the 40th Annual ACM Symposium on Theory of Computing, Victoria, British Columbia, Canada, May 17–20, pp. 671–680 (2008)

7. Balcan, M.-F., Blum, A.: Clustering with interactive feedback. In: Freund, Y., Györfi, L., Turán, G., Zeugmann, T. (eds.) ALT 2008. LNCS (LNAI), vol. 5254, pp. 316–328. Springer, Heidelberg (2008)
8. Banerjee, A., Peikert, C., Rosen, A.: Pseudorandom functions and lattices. In: Pointcheval, D., Johansson, T. (eds.) EUROCRYPT 2012. LNCS, vol. 7237, pp. 719–737. Springer, Heidelberg (2012)
9. Bartal, Y., Charikar, M., Raz, D.: Approximating min-sum-clustering in metric spaces. In: STOC, pp. pp. 11–20 (2001)
10. Basu, S., Banerjee, A., Mooney, R.J.: Active semi-supervision for pairwise constrained clustering. In: Proceedings of the Fourth SIAM International Conference on Data Mining, Lake Buena Vista, Florida, USA, April 22–24, pp. 333–344 (2004)
11. Ben-David, S.: Computational feasibility of clustering under clusterability assumptions. CoRR abs/1501.00437 (2015)
12. Brubaker, S.C., Vempala, S.I.: PCA and affine-invariant clustering. In: 49th Annual IEEE Symposium on Foundations of Computer Science, FOCS 2008, pp. 25–28, Philadelphia, PA, USA, pp. 551–560 (October 2008)
13. Charikar, M., Guha, S., Tardos, É., Shmoys, D.B.: A constant-factor approximation algorithm for the k-median problem. J. Comput. Syst. Sci. **65**(1), 129–149 (2002)
14. Dasgupta, A., Hopcroft, J., Kannan, R., Mitra, P.: Spectral clustering by recursive partitioning. In: Azar, Y., Erlebach, T. (eds.) ESA 2006. LNCS, vol. 4168, pp. 256–267. Springer, Heidelberg (2006)
15. Dasgupta, S., Ng, V.: Which clustering do you want? inducing your ideal clustering with minimal feedback. J. Artif. Intell. Res. (JAIR) **39**, 581–632 (2010)
16. de la Vega, W.F., Karpinski, M., Kenyon, C., Rabani, Y.: Approximation schemes for clustering problems. In: STOC, pp. 50–58 (2003)
17. Guha, S., Khuller, S.: Greedy strikes back: Improved facility location algorithms. J. Algorithms **31**(1), 228–248 (1999)
18. Jain, K., Mahdian, M., Saberi, A.: A new greedy approach for facility location problems. In: STOC, pp. 731–740. ACM (2002)
19. Kearns, M., Valiant, L.: Cryptographic limitations on learning boolean formulae and finite automata. Journal of the ACM (JACM) **41**(1), 67–95 (1994)
20. Kumar, A., Sabharwal, Y., Sen, S.: Linear-time approximation schemes for clustering problems in any dimensions. J. ACM **57**, 2 (2010)
21. Lewko, A.B., Waters, B.: Efficient pseudorandom functions from the decisional linear assumption and weaker variants. In: Proceedings of the 16th ACM Conference on Computer and Communications Security, pp. 112–120. ACM (2009)
22. Naor, M., Reingold, O.: Number-theoretic constructions of efficient pseudo-random functions. Journal of the ACM (JACM) **51**(2), 231–262 (2004)
23. Pitt, L., Warmuth, M.K.: Prediction-preserving reducibility. Journal of Computer and System Sciences **41**(3), 430–467 (1990)
24. Valiant, L.G.: A theory of the learnable. Communications of the ACM **27**(11), 1134–1142 (1984)