Interactive Clustering of Linear Classes and Cryptographic Lower Bounds

Adam D. Lelkes

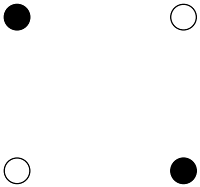
University of Illinois at Chicago

Joint work with Lev Reyzin

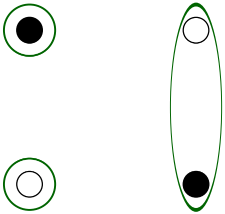
The Model

- Model introduced by Balcan and Blum (2008)
- Learner proposes hypothesis clustering; adversarial teacher replies with
  - `accept`: the proposed clustering is the target clustering,
  - `split`(c): c contains points from more than one target cluster (c is "impure"), or
  - `merge`(c, d): $c \cup d$ is a subset of a target cluster.

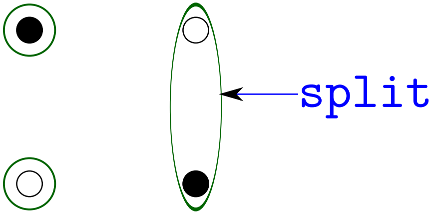
A Simple Example



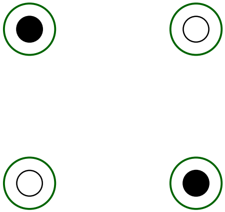
A Simple Example



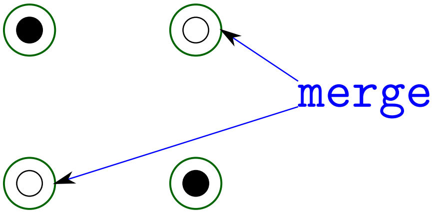
A Simple Example



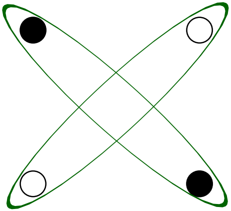
A Simple Example



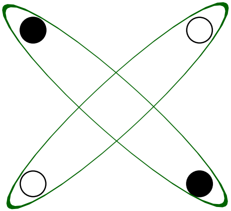
A Simple Example



A Simple Example



A Simple Example



accept

The Model

- Number of points: m (finite)
- The number of target clusters, k, is fixed and known to the clustering algorithm.
- Target clustering comes from a concept class C (known to the clustering algorithm).
  (Note: C is a subset of the set of partitions of m points, therefore finite.)

The Model

- Number of points: m (finite)
- The number of target clusters, k, is fixed and known to the clustering algorithm.
- Target clustering comes from a concept class C (known to the clustering algorithm).
  (Note: C is a subset of the set of partitions of m points, therefore finite.)

Definition

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

Previous Work

- Efficient algorithms for intervals, disjunctions, conjunctions (for constant k) (Balcan and Blum 2008)
- General (but inefficient) version space algorithm (Balcan and Blum 2008)
- Efficient algorithm for rectangles, noisy version of the model, improved general version space algorithm with query complexity $O(k \log |C|)$ (Awasthi and Zadeh 2010)

Our Contribution

- Efficient algorithm for parity (and, more generally, for linear functionals over finite fields)
- Efficient algorithm for hyperplanes in $\mathbb{R}^d$ (for constant d)
- Cryptographic lower bounds

Parity

- Points are from $\{0,1\}^n$
- Concept class: parity functions, i.e. functions of the form $v \mapsto x \cdot v$ (over $GF(2)$)
- Number of target clusters: 2

Parity

- Points are from $\{0, 1\}^n$
- Concept class: parity functions, i.e. functions of the form $v \mapsto x \cdot v$ (over $GF(2)$)
- Number of target clusters: 2

Idea: starting from all singletons, every `merge` request gives us a linear equation for x. In each round, we output the coarsest clustering we know to be pure.

Parity

0110 0000

1011

1101

1111

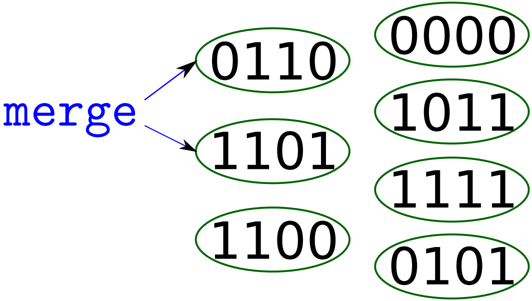
1100

0101

Parity

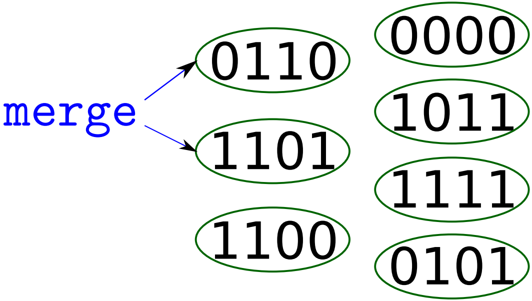
0110	0000
1101	1011
1100	1111
	0101

Parity



Parity

$x \cdot 1011 = 0$



Parity

$x \cdot 1011 = 0$

0110
1101

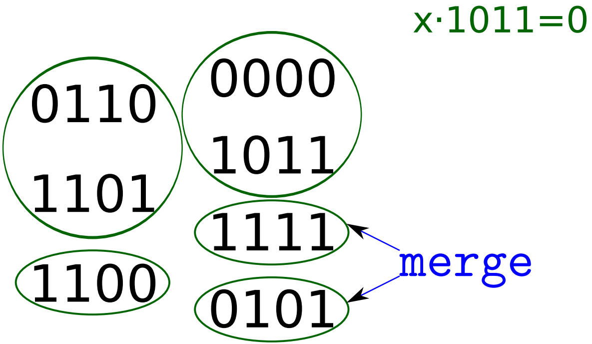
1100

0000
1011

1111

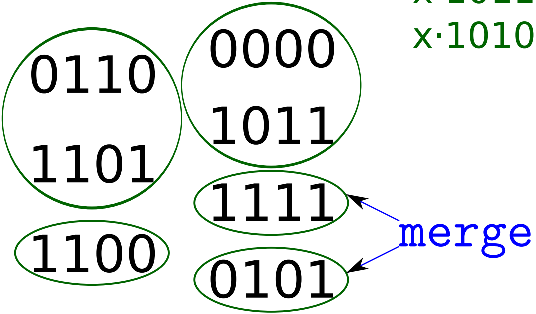
0101

Parity



Parity

$x \cdot 1011 = 0$
$x \cdot 1010 = 0$



Parity

0110
1101
1100

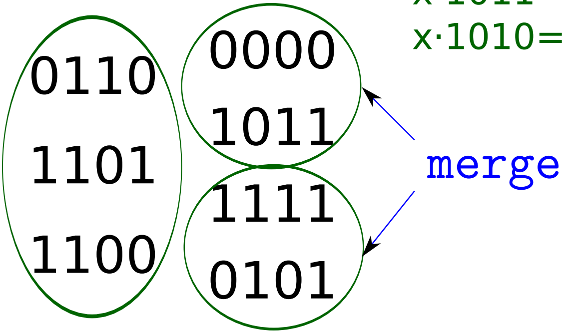
0000
1011

1111
0101

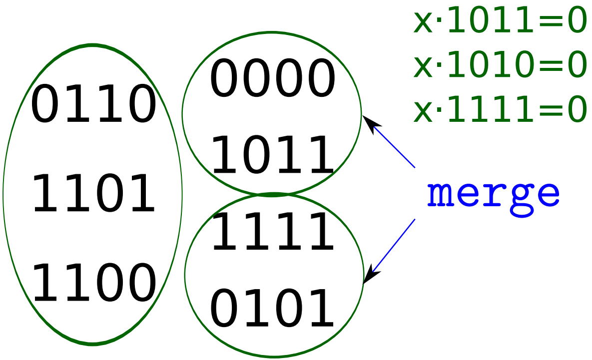
$x \cdot 1011 = 0$
$x \cdot 1010 = 0$

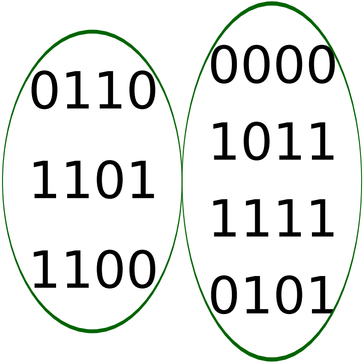
Parity

$x \cdot 1011 = 0$
$x \cdot 1010 = 0$



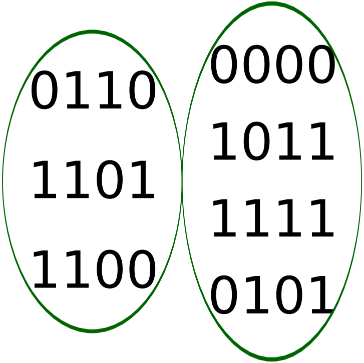
Parity


0110
1101
1100
0000
1011
1111
0101
x·1011=0
x·1010=0
x·1111=0
merge



x·1011=0
x·1010=0
x·1111=0

Parity



x·1011=0
x·1010=0
x·1111=0

accept

Hyperplanes

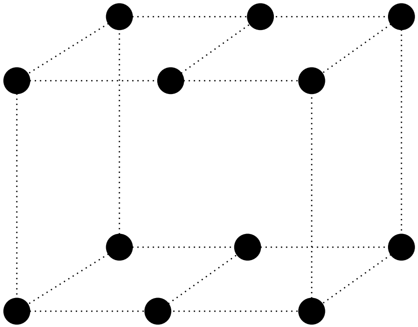
- Clusters are k hyperplanes in $\mathbb{R}^d$

Hyperplanes

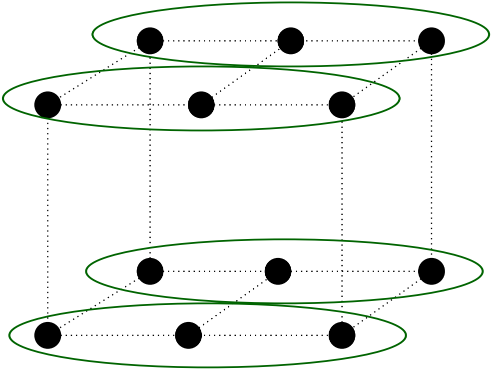
- Clusters are k hyperplanes in $\mathbb{R}^d$
- Observation: if there are $k+1$ collinear points, they have to be in the same target cluster by pigeonhole principle; we can merge them into one line.
- We can then use the pigeonhole principle iteratively to merge higher dimensional subspaces.
- After this, we only need to follow a few merge requests to get the target clustering.

Planes

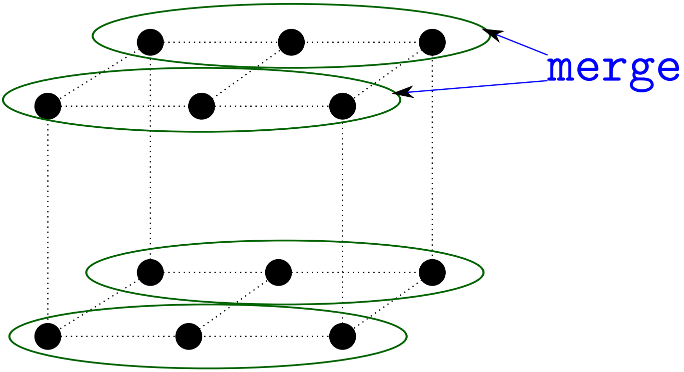
k=2



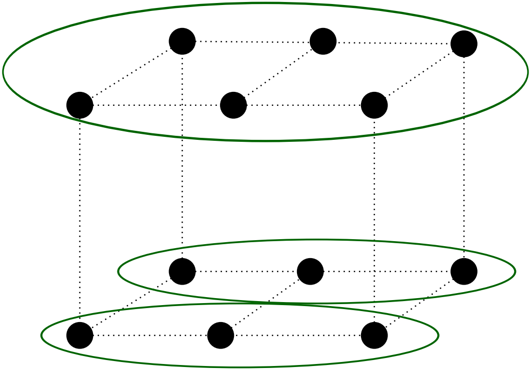
Planes



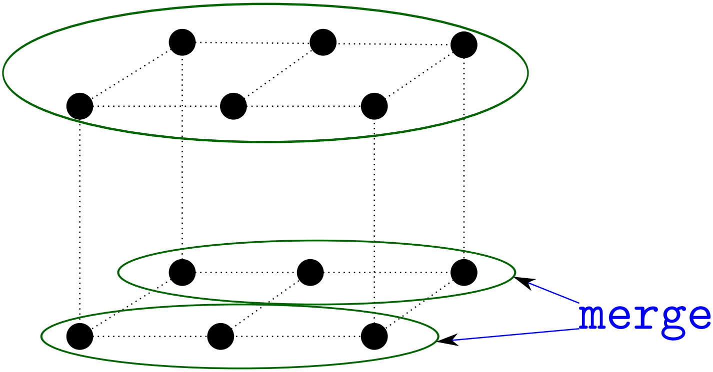
Planes



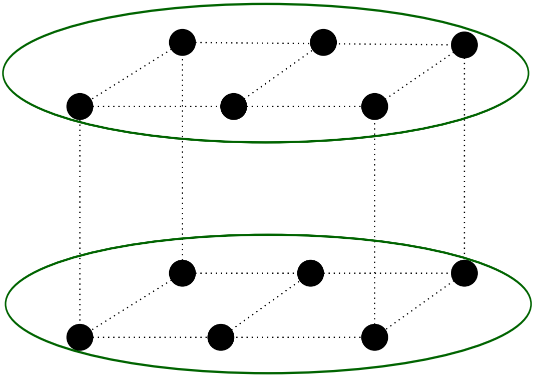
Planes



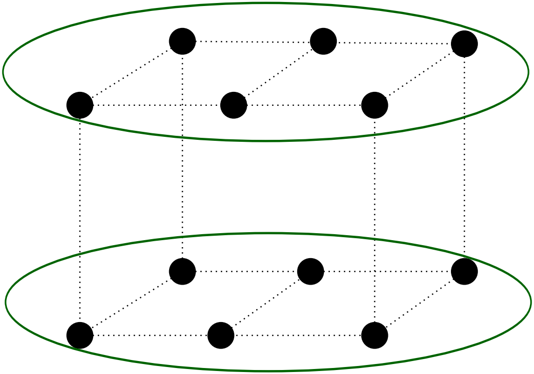
Planes



Planes



Planes



accept

Lower Bounds

How can we prove lower bounds?

Unlike in the PAC model, we don't have to generalize to new data.

Lower Bounds

How can we prove lower bounds?

Unlike in the PAC model, we don't have to generalize to new data.

Information-theoretic lower bound:

Lemma

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.

Lower Bounds

Idea: let us have two concept classes.

1. Big class: lemma gives query lower bound.

2. Small one: the definition of efficient clustering requires a small number of queries.

Lower Bounds

Idea: let us have two concept classes.

1. Big class: lemma gives query lower bound.

2. Small one: the definition of efficient clustering requires a small number of queries.

We put all function into the big class, a pseudorandom function family in the small class. If we could cluster the small class, that would break the pseudorandom function family.

Lower Bounds

Big class:

Instance space: $\{0,1\}^n$

Concept class: all functions $\{0,1\}^n \to \{0,1\}$

Size of concept class: 2^{2^n}

Number of points: $m(n) = n^{\omega(1)}$ chosen carefully

Query lower bound from lemma: superpolynomial

Lower Bounds

Small class:

Instance space: $\{0,1\}^n$

Concept class: an $t(n) = n^{\omega(1)}$-hard keyed pseudorandom function family with seed length n

Size of concept class: 2^n

Number of points: $m(n) = n^{\omega(1)}$ chosen carefully

Query upper bound from definition: $\mathrm{poly}(n, \log m(n)) = \mathrm{poly}(n)$

Lower Bounds

If we could cluster the small class efficiently, we could distinguish the pseudorandom function family from the uniform distribution on all functions.

Lower Bounds

Results:

1. 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 hard to cluster.
2. Corollary 1: if factoring is $n^{\omega(1)}$-hard, then TC^0, polynomial-size Boolean formulas are not clusterable.
3. Corollary 2: if there are $n^{\omega(1)}$-hard pseudorandom functions in logspace, polynomial-size DFAs are also not clusterable.

Open Problems

1. Better algorithm for hyperplanes
2. Half-spaces

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

Hyperplanes

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