2012 IEEE 53rd Annual Symposium on Foundations of Computer Science

Population Recovery and Partial Identification

Avi Wigderson*
School of Mathematics
Institute for Advanced Study
Princeton, NJ
avi@math.ias.edu

Amir Yehudayoff†
Department of Mathematics
Technion-IIT
Haifa, Israel
amir.yehudayoff@gmail.com

***Abstract*—We study several problems in which an *unknown* distribution over an *unknown* population of vectors needs to be recovered from partial or noisy samples, each of which nearly completely erases or obliterates the original vector. For example, consider a distribution p over a population $V \subseteq \{0,1\}^n$. A noisy sample v' is obtained by choosing v according to p and flipping each coordinate of v with probability say 0.49 independently. The problem is to recover V, p as efficiently as possible from noisy samples. Such problems naturally arise in a variety of contexts in learning, clustering, statistics, computational biology, data mining and database privacy, where loss and error may be introduced by nature, inaccurate measurements, or on purpose. We give fairly efficient algorithms to recover the data under fairly general assumptions.**

Underlying our algorithms is a new structure we call a *partial identification (PID) graph* for an arbitrary finite set of vectors over any alphabet. This graph captures the extent to which certain subsets of coordinates in each vector distinguish it from other vectors. PID graphs yield strategies for dimension reductions and re-assembly of statistical information.

The quality of our algorithms (sequential and parallel runtime, as well as numerical stability) critically depends on three parameters of PID graphs: width, depth and cost. The combinatorial heart of this work is showing that *every* set of vectors posses a PID graph in which all three parameters are small (we prove some limitations on their trade-offs as well). We further give an efficient algorithm to find such near-optimal PID graphs for any set of vectors.

Our efficient PID graphs imply general algorithms for these recovery problems, even when loss or noise are just below the information-theoretic limit! In the learning/clustering context this gives a new algorithm for learning mixtures of binomial distributions (with known marginals) whose running time depends only quasi-polynomially on the number of clusters. We discuss implications to privacy and coding as well.

***Keywords*-information recovery; learning theory; noisy data;**

I. INTRODUCTION

Recovery of information from lossy or noisy data is common to many scientific and commercial endeavors. In this work we introduce a new structure we call a partial identification (PID) graph that helps in solving such recovery problems.

The use of IDs, features of individuals that *uniquely* identify them in a given population, is ubiquitous. Fingerprints and retinal scans, or simply names and addresses, are useful for identifying people. In other populations, various appropriate subsets of features are used[1]. In some populations, however, some IDs may be too long, and thus too expensive to maintain, e.g., when storage is limited. In these cases, *partial* IDs can be used: instead of a long list of features per individual, we can maintain a shorter list that is more efficient. This efficiency has a cost, as it may introduce ambiguity and "imposters." PID graphs represent the imposter-structure of a given choice of PIDs.

We provide fairly efficient algorithms for recovery and clustering of information from lossy/noisy data. These algorithms use PID graphs to identify useful local sub-systems of a given system, and combine local pieces of information to a global one. The complexity of our algorithms crucially depends on the PID graphs they use. A substantial part of our algorithms, therefore, is a procedure that builds efficient PID graphs.

Roadmap: As the introduction is long, here is a roadmap to it. In section I-A we give intuition and formally define population recovery problems, samplers and our main results. In section I-B we explain why a simpler "distribution recovery" problem is sufficient to solve. Sections I-C and I-D respectively explain how parameters of PID graphs control our algorithms, and how to efficiently achieve good parameters. Section I-E extensively discusses our work in various contexts which considered similar problems and methods.

A. Recovery problems

Let us start with two scenarios that motivate the kind of problems we consider in this paper. We then formulate them more precisely.

Recovery from lossy samples: Imagine that you are a paleontologist, who wishes to determine the population of dinosaurs that roamed the Earth before the hypothetical

* Research partially supported by NSF grants CCF-0832797 and DMS-0835373.

† Horev fellow – supported by the Taub Foundation. Research supported by ISF and BSF.

[1] It will be clear that in the settings we consider, hashing and related techniques are irrelevant, and one has to use actual features of individuals.

0272-5428/12 $26.00 © 2012 IEEE
DOI 10.1109/FOCS.2012.14

IEEE computer society

meteor made them extinct. Typical observations of dinosaurs consist of finding a few teeth of one here, a tailbone of another there, perhaps with some more luck a skull and several vertebrae of a third, and rarely a near complete skeleton of a fourth. Each observation belongs to one of several species. Using these fragments, you are supposed to figure out the population of dinosaurs, namely, a complete description of (say) the bone skeleton of each species, *and* the fraction that each species occupied in the entire dinosaur population. Even assuming that our probability of finding the remains of a particular species, e.g. Brontosaurus, is the same as its fraction in the population, the problem is clear: while complete features identify the species, fragments may be common to several. Even with knowledge of the complete description of each type (which we a priori do not have), it is not clear how to determine the distribution from such partial descriptions. A modern-day version of this problem is analyzing the partial Netflix matrix, where species are user types, features are movies, and each user ranked only relatively few movies.

Recovery from noisy samples: Imagine you are a social scientist who wants to discover behavioral traits of people and their correlations. You devise a questionnaire where each yes/no question corresponds to a trait. You then obtain a random sample of subjects to fill the questionnaire. Many of the traits, however, are private, so most people are unwilling to answer such questions truthfully, fearing embarrassment or social consequences if these are discovered[2]. To put your subjects at ease you propose that instead of filling the questionnaire truthfully, they fill it randomly: as follows: to each question, they flip the true answer with probability 0.49, independently of all others. This surveying method (typically applied when there is one question of interest) is known as "randomized response" and was initiated by Warner [26]. Is privacy achieved? Can one recover the original truthful information about the original sequences of traits and their distribution in the population? Observe that typical samples look very different than the sequences generating them. Indeed, can one synthesize sensitive databases for privacy by publishing very noisy versions of their records?

These examples illustrate some of the many settings in which lossy or noisy samples of concrete data is available, and from which one would like to efficiently recover a full description of the original data. Losses and noise may be an artifact of malicious or random actions, and the entity attempting recovery may be our friends or foes. We formalize below the general setting and show that, surprisingly, accurate recovery is possible, as long as samples bear minimal correlation with *some* data item. This is achieved even when *both* the dimension (number of "attributes") and the population size (number of "clusters") are large!

1) Definitions and results: Fix an alphabet Σ, integer parameters n, k and an accuracy parameter $\alpha > 0$. Assume, as in the examples above, there is a population of vectors $V \subseteq \Sigma^n$ of size (at most) k, and a probability distribution π on the population V. In the following, we shall not know π or even V, but rather only get lossy or noisy samples to them. Our goal will be to fully recover both V and π.

The *population recovery problem* (PRP) is to (approximately) reconstruct V and π, up to the accuracy parameter α. Namely, given independent lossy or noisy samples, find a set V' containing every element $v \in V$ for which $\pi(v) > \alpha$, and a distribution π' on V' such that for every $v \in V'$ we have $|\pi(v) - \pi'(v)| < \alpha$.

We start by focusing on two processes for generating samples (which we will later generalize): lossy sampling and noisy sampling. This type of recovery problem was first formulated in [5], for the model of lossy sampling.

Lossy samples: Let $0 < \mu < 1$. A *μ-lossy* sample v' is generated randomly as follows. First, an element $v \in V$ is picked at random with probability $\pi(v)$. Then, independently at random, every coordinate v_i of v is replaced by the symbol "?" (assumed not to be in Σ) with probability $1 - \mu$, and is left untouched with probability μ. The result is v'. For example, a 0-lossy sample is a vector of question marks, and a 1-lossy sample gives full access to V.

Noisy samples: Let $-1 < \nu < 1$, and for simplicity assume $\Sigma = \{0, 1\}$. A *ν-noisy* sample v' is generated as follows. First, an element $v \in V$ is picked at random with probability $\pi(v)$. Then, independently at random, every bit v_i of v is flipped with probability $(1 - \nu)/2$, and left untouched with probability $(1 + \nu)/2$. The result is v'. For example, a 0-noisy sample is a uniform random vector from the discrete cube (regardless of V, π).

The nature of the recovery problem, namely, that we only have access to samples, requires that we allow algorithms to err with some small probability $\delta > 0$. The main question we study is "for which noise/loss parameters is there an efficient reconstruction algorithm, hopefully that runs in time polynomial in all other parameters $|\Sigma|$, n, k, $1/\alpha$, $\log(1/\delta)$?"

It is not hard to see that, in the noisy model, if $\nu = 0$ it is information theoretically impossible to solve the recovery problem. If $\nu = o(1)$ then it is impossible to solve the recovery problem in time polynomial in k, e.g., consider the case of distinguishing all even-weight strings from all odd-weight strings of length $n = \log k$. Similarly, for $\mu = 0$ the lossy recovery problem is information theoretically impossible, and for $\mu = o(1)$ it is impossible to solve it in time polynomial in k.

In all other cases, namely when $\mu, \nu > 0$ are constants, we design algorithms for these two problems, which run in time polynomial in $|\Sigma|, n, 1/\alpha, \log(1/\delta)$ and $k^{\log k}$. The exponent of the polynomial depends logarithmically on $1/\mu$, $1/\nu$, respectively. The question of reducing the dependence on k to polynomial is the main open problem. We now

[2] A possibly worse scenario is when looking for information concerning medical conditions, where leakage of the truth may cause one to lose medical insurance.

state the main results, starting with the lossy case and then describing the noisy case.

Theorem 1. *Let $0 < \mu < 1$. There is an algorithm that, given an integer k, $\alpha > 0$, $0 < \delta < 1$ and μ, runs in time* $\mathsf{poly}(|\Sigma|, n, 1/\alpha, \log(1/\delta), k^{\log k})$ *and solves the recovery problem for an n-dimensional database of size k over Σ, given μ-lossy samples, with error α and probability at least $1-\delta$.*

Independence of erasures of different coordinates is not necessary for the algorithm stated in the theorem to perform well. Indeed, the algorithm works as long as every set of size at most $\log k$ coordinates has a non-negligible probability (say β) of having no question marks. The running time in this case may be slowed down by a factor of $\mathsf{poly}(1/\beta)$. For example, as long as the question marks are $(\log k)$-wise independent, we have $1/\beta = (1/\mu)^{\log k} = \mathsf{poly}(k)$.

In [5] a completely different algorithm was given for the lossy case, which is fully polynomial in all parameters including k, but works only as long as $\mu > 0.365$. For smaller values of μ, however, exponential running time could not be ruled out. The motivation in [5] was DNF-learning in the *restriction access* PAC learning model (defined there). Via their reduction the above theorem implies a learning algorithm for all $\mu > 0$.

Corollary 2. *For any fixed $0 < \mu < 1$, there is an algorithm that, given k, n, α, δ, learns n-variate DNF formulas of size k in the restriction access model with[3] μ-lossy samples and probability at least $1-\delta$ in time* $\mathsf{poly}(n, 1/\alpha, \log(1/\delta), k^{\log k})$.

Our main result for recovery from the noisy samples is similar to the lossy one.

Theorem 3. *Let $0 < \nu < 1$. There is an algorithm that, given an integer k, $\alpha > 0$ and $0 < \delta < 1$ and[4] ν, runs in time* $\mathsf{poly}(n, 1/\alpha, \log(1/\delta), k^{\log k})$ *and solves the recovery problem for an n-dimensional database of size k over $\Sigma = \{0,1\}$ with ν-noisy samples, error α and probability at least $1-\delta$.*

The study of recovering information from noisy samples appears already in the work of Kearns, Mansour, Ron, Rubinfel, Schapire and Sellie [19]. There it is called "learning mixtures of hamming balls" and two algorithms are presented. One with a similar running time to our algorithm, but that only guarantees small divergence, that is, the population V is not reconstructed exactly but only up to hamming distance roughly $\sqrt{n}$. A second that basically guarantees small statistical distance, but runs in time exponentially in k.

With an appropriate definition of noise, one can generalize this theorem too to every alphabet Σ. Moreover, it can also be generalized to the case that a different (but known) noise probability is applied in every coordinate. As in the lossy case, we do not need the full independence of the noise but only $(\log k)$-wise independence. We do, however, require knowledge of the noise parameter ν (or at least some polynomially good estimate of it).

More general samplers: A third sampler one can consider is a *hybrid sampler* combining the lossy and noisy ones. In the hybrid model, a sample is generated by first erasing all but a μ fraction of the coordinates, and then flipping each one of the remaining coordinates with probability $(1-\nu)/2$. A fourth sampler is a noisy sampler where every coordinate i is flipped with probability $(1-\nu_i)/2$ that depends on i. Even in these more complex situations we can achieve the same bounds on recovery (in general knowledge of the noise model is required for our algorithm to perform well). In fact, when proving the above theorems, we consider even a more general type of samplers that we define in Section III-A below.

Our algorithms use the following strategy. Due to the partial, distorted and inconsistent information provided by samples, we focus on small *windows* (subsets of coordinates). The strategy consists of two phases. In the first phase, we collect statistics for many judiciously chosen windows. This phase may be viewed as a collection of dimension reductions of the original problem. Since windows are small (of logarithmic size), we can afford running time that is exponential in the window size, and so we can get pretty accurate information for each of the chosen windows. However, as many vectors may agree on their projection on a small window, we need to "decouple" the statistics thus obtained. This is done in the second phase, where special care is needed to control accumulation of errors. The implementation of this strategy is guided by a PID graph. To build the PID graph though, we need to know the underlying population (which is actually part of what we want to learn in PRP). So, we should first explain why we can assume knowledge of the population (but not of the underlying distribution).

B. Recovery when vectors are known

The *distribution recovery problem* (DRP) is a sub-problem of PRP, in which the set V of vectors in the population is actually given to the algorithm, and the only problem is estimating the unknown probability distribution π on V.

An important reduction provided in [5] shows that PRP can be efficiently reduced to DRP: if from V we can deduce π, we can find V as well[5]. In other words, we may assume

[3] In the notation of [5], μ-lossy samples are called "set distribution Ind_μ."

[4] In the full version we shall explain how to estimate the noise parameter when $n > \log k$. This enables the algorithm to perform well even without knowing ν.

[5] In follow-up work, Impagliazzo [12] shows that recovering only (a super-set of) V, but not π, can be done in polynomial time from lossy samples and quasi-polynomial time from noisy ones.

V is known even though it is not! This somewhat counter-intuitive statement has a very simple proof and moreover is quite general. It works for all *uniform*[6] samplers of V, in particular, the ones we discussed above. We note that it is very similar in spirit to the prefix constructions in other learning settings, such as [10], [15]. For completeness (and since it is not stated in this generality in [5]), we state it here and provide the formal definition of a uniform sampler and a proof sketch in the appendix.

For uniform samplers, solving PRP is as easy as solving n DRPs, as the following theorem shows. In the reduction, there is no need to consider probability of success, but setting an error probability δ/n for the DRP solver yields an error probability δ for the PRP solver by a union bound.

Theorem 4 ([5]). *Assume f is a lossy/noisy/hybrid sampler. There is an algorithm that given f and parameters Σ,k,n,α, makes n oracle calls to a DRP algorithm using f and parameters Σ,k,i,$\alpha/2$ for every $i \in [n]$, and if all n calls produce a correct answer, the algorithm solves PRP.*

We are, therefore, left with solving DRP from lossy and noisy samples. That is, we can assume we know V and only need to estimate π. The approach taken here, which is very different than that of [5], is to first efficiently construct a "succinct" representation for vectors in V. To motivate it, let us first observe that DRP is much easier if the number n of "features" of an individual was small, say $n \leq O(\log k)$ where $|V| = k$. This allows us to use algorithms that run in exponential time in n. The following proposition states that exponential running time suffices for solving the problem (and, in fact, for a more general family of samplers; see Section III-A for details).

Proposition 5. *DRP can be solved in both sample models in time exponential in n.*

How can we make use of the proposition when n is large (as it typically is)? For this we introduce a generic way to reduce the dimension: use PID graphs.

C. Dimension reductions and PID graphs

At this point V is available to us and we can preprocess it. Our approach is to pick, for every individual $v \in V$, a small representative set of features $S_v \subseteq [n]$, of size $|S_v| \leq \log k$. Then, solve DRP on the window S_v, ignoring all other coordinates, in time $\mathsf{poly}(k)$ using the lemma above. For a vector v and a set S, denote by $v[S]$ the restriction of v to the entries in S. If $v[S_v]$, the pattern of v in coordinates S_v, *uniquely* identifies v in V (namely it is an ID), then we directly get an estimate of $\pi(v)$. Unfortunately, such short IDs do not exist for all vectors in a general populations V. For some vectors v the pattern on S_v is shared by other *imposters* $u \in V$. Thus the set S_v is only a *partial ID* (PID) for v, and the statistics obtained from the DRP solver on S_v counts the total mass of π of all imposters u of v. Recovering π from this information is clearly a linear system. This linear system is not always invertible, so we first need to ensure that it is. But, even when it is invertible, recall that we only have estimates, and inverting the system can exponentially blow up the accuracy parameter.

To study the cost of this computation, it is natural to capture "imposter relations" in a graph. A *PID graph* $G = G(V, \mathcal{S})$ is defined for any set of vectors $V \subseteq \Sigma^n$ and any set of PIDs $\mathcal{S} = \{S_v \subseteq [n] : v \in V\}$. The vertices of G are the elements of V. Directed edges go from every vertex v to every imposter u of v, namely, $u \neq v$ such that $u[S_v] = v[S_v]$.

The set of *imposters* $I(v)$ of v contains v itself and all vertices u such that (v, u) is an edge in G. Let $p(v) = \sum_{u \in I(v)} \pi(u)$. When G is acyclic, as we will ensure, the linear system expressing p in terms of π is *triangular*. So, knowing p, we can compute π by backwards substitution. But, as we shall only have an approximation p' of p, we need to ensure that the initial error of p' is small enough so that accumulated error in π' will be small as well.

The accumulation of errors in this computation is affected by $\mathsf{depth}(G)$, the length of the longest directed path in it. This motivates using a graph G of smallest possible depth. The depth also dominates the running time of any parallel algorithm for this task. This is a secondary reason for minimizing the depth.

Depth is just a crude parameter controlling error accumulation. A more precise one is $\mathsf{cost}(G)$, defined recursively for acyclic G according to its topological order by $\mathsf{cost}(v) = 1$ on every sink v in G, and

$$\mathsf{cost}(v) = 1 + \sum_{u \in I(v) : u \neq v} \mathsf{cost}(u).$$

Note that $\mathsf{cost}(v)$ is simply the number of directed paths (of all lengths) emanating at v. Define

$$\mathsf{cost}(G) = \max\{\mathsf{cost}(v) : v \in V\}.$$

Trivially,

$$\mathsf{cost}(G) \leq |V|^{\mathsf{depth}(G)}.$$

As we already explained, the runtime of solving each dimension-reduced DRP problem on S_v is exponential in $|S_v|$. This aspect is capture by the *width* of G,

$$\mathsf{width}(G) = \max\{|S_v| : v \in V\}.$$

We conclude by showing how the runtime of the full DRP algorithm depends on these three parameters.

Theorem 6. *Let G be a PID graph for a population V of size $|V| = k$.*

- *Given G, the distribution recovery problem with accuracy $\alpha > 0$ for V can be solved, with probability*

[6] Due to space limitations, we do not formally define uniform samplers. The definition appears in the full version.

at least $1-\delta$, using μ-lossy samples in time $T = \mathsf{poly}(nk|\Sigma|\log(1/\delta)/\alpha)\cdot\mathsf{cost}(G)\cdot(1/\mu)^{\mathsf{width}(G)}$. Moreover, it can be solved in parallel using T processors in time $\mathsf{depth}(G)\cdot\mathsf{poly}\log T$.

- *Given G, the distribution recovery problem with accuracy $\alpha > 0$ for V can be solved, with probability at least $1-\delta$, using ν-noisy samples in time $T = \mathsf{poly}(nk\log(1/\delta)/\alpha)\cdot\mathsf{cost}(G)\cdot(1/\nu)^{\mathsf{width}(G)}$. Moreover, it can be solved in parallel using T processors in time* $\mathsf{depth}(G)\cdot\mathsf{poly}\log T$.

D. Efficient PID graphs

We are now faced with constructing, for any population V, a PID graph with the smallest possible width and depth (and cost). Minimizing only one of these parameters is easy; considering the following examples will help getting some intuition.

To minimize depth, pick $S_v = [n]$ for all v. Since these PIDs are actually IDs, there are no imposters and the depth is zero. The width, however, is n, which would lead to an exponential time algorithm. Note that if we insist on IDs, the width may have to be n, e.g. when V consists of all unit vectors e_i as well as the all zeros vector $\bar{0}$. While each e_i has an ID of size one, the only ID for the vector $\bar{0}$ is $[n]$.

On the other extreme, let us see how we can easily make the width at most logarithmic. In this case, however, the depth is possibly k, which again leads to an exponential running time (due to error-accumulation). First observe[7] that in any V, there must be at least one member v with an ID S_v of size at most $\log|V|$. We can now remove v from the population, and consider the set $\tilde{V} = V\setminus\{v\}$. We can now find a vector $\tilde{v}$ which has a short ID in $\tilde{V}$. Even though $S_{\tilde{v}}$ is an ID of $\tilde{v}$ in $\tilde{V}$, the vector v may still be an imposter of $\tilde{v}$ in V. Continuing in this fashion results in a set of PIDs, each of which has size at most $\log k$, but may yield a graph of depth k. Indeed, if our set of vectors V is simply all $n+1$ vectors of the form $1^i 0^{n-i}$, then all PIDs thus obtained have size at most 1, but their graph is a path of length $n+1$. This graph happens to have small cost, but similar examples in which the cost is exponential can be given as well.

We prove that width, depth (and thus cost) can simultaneously be made small. Moreover, we can find such PIDs efficiently.

Theorem 7. *There is an algorithm that, given V of size k, runs in time* $\mathsf{poly}(k)$*, and finds a PID S_v for every v in V so that the corresponding PID is so that* $\mathsf{width}(G)\le\log k$ *and* $\mathsf{depth}(G)\le\log k$*. Specifically,* $\mathsf{cost}(G)\le k^{\log k}$.

Theorems 1 and 3 follow immediately from the above theorems. Observe that finding the smallest possible ID for a given v in V is exactly the hyper-graph vertex-cover problem which is NP-hard to compute.

It would be preferable to obtain short PIDs whose graph has a polynomial upper bound on the cost. Such a bound would automatically yield a polynomial running time for our algorithms. Unfortunately, such a bound can not be obtained.

Proposition 8. *For every k there is a databases V of size k over $\Sigma = \{0,1\}$ so that* every *PID graph for V of width at most $O(\log k)$ has super-polynomial cost, at least $k^{\Omega(\log\log k)}$.*

This example shows that in order to solve DRP in polynomial time new ideas are required. We note, however, that for most databases V, the cost of a PID graph is much smaller. So, for an "average input" our algorithm runs faster.

E. Discussion and related work

Our work addresses situations in which a database of vectors in high dimension undergoes heavy obliteration and needs to be recovered. We consider situations where noise or loss are close to the information-theoretic limit; slightly increasing their levels completely destroys all information about the original data. We, nevertheless, provide (reasonably) efficient algorithms that completely recover all the data with arbitrary high probability. Improving the efficiency (e.g. to be polynomial in k) and relaxing the assumptions (e.g. the knowledge of noise level) are the obvious next steps to make these results more useful in practice.

In this section we discuss our results and our methods in comparison to similar ones in the literature. There are numerous works dealing with a wide variety of recovery from lossy and noisy information, and we can't hope to survey them all. These arise in various fields, and sometimes with different motivations, assumptions and terminology, and we try to relate them to ours. The partitions to topics below are somewhat arbitrary and certainly overlapping.

Learning: The population recovery problem from lossy samples was introduced in [5]. It arose naturally from the analysis of PAC-learning algorithms in a general non-black-box learning framework they introduced, called *restriction access*. Here we introduced noisy (and more general) sample-models. The algorithms we describe suggest that the restriction access model can be generalized to allow noise as well as loss. We note that our recovery algorithms are very different than the ones in [5].

Other works in which a set of vectors (in this case, representing high Fourier coefficients of a function given by an oracle) is discovered from samples include the famous algorithms for learning decision trees and DNFs [10], [15], [13]. These works differ in several aspects; in particular, membership queries are allowed, as opposed to only samples from the distribution.

Clustering: One natural way to view the noisy model is as follows. The unknown vectors v in our database are k

[7] Indeed, to find such v, S_v take the left-most (non-constant) coordinate in $[n]$, and keep only vectors in V that have the least popular value in this coordinate. Now do it again, and again, until you are left with only one vector v. This process chose at most $\log|V|$ coordinates, which serve as an ID for v.

cluster *centers*, and the noise generates a mixture of *binomial* distributions around these centers, with unknown mixture coefficients determined by π. This problem was considered when the binomial noise is *unknown*, and [27], [4], [9] give algorithms to learn such mixtures. A different closely related problem is the well-studied learning mixtures of Gaussians problem (see [18] and references therein). The most crucial difference between these works and ours is that we assume to know the binomial distributions around the cluster, and previous works do not. This is, of course, less general, but allows us to achieve a better dependence on k, the number of centers. For example, previous algorithms are polynomial only for constant k, but ours is polynomial even for k polynomial in $2^{\sqrt{\log n}}$. Another difference is that in the Gaussian noise model the distribution is continuous and the vectors naturally reside in high dimensional Euclidean space, whereas the binomial noise is discrete and vectors reside in high dimensional combinatorial cube. The Gaussian distribution is spherically symmetric and Euclidean space allows projections in all directions, whereas in the Boolean case symmetry is restricted and allows only projection on coordinates.

Codes: Yet another, quite different view of our result is as a surprising robustness of repetition codes. Imagine the following way to encode a *multi-set* of k binary vectors of length n: simply make many (say m) copies of each. We show that this simple repetition code tolerates the following vicious kind of attack. First, the km vectors are arbitrarily permuted. Next, in each vector, a random 99% of the coordinates are erased (replaced by question marks). Finally, the surviving bits are each flipped with probability 49%. It is not clear that this leaves enough information to recover the original data. We show that for rate $m = \mathsf{poly}(n, k^{\log k})$, recovery is not only information theoretically possible but can be performed in polynomial time. Of course, determining the minimal rate for which this is possible is an intriguing question (relevant to the next item as well).

Privacy: Resolving the conflict between individual privacy in large databases and their usability for statistical, medical and social research is an important research area. Again, we can do no justice to the volumes of work on the subject, and we resort to generalities. One general principle that is used is to obliterate individual records and perturb the data by applying to it random noise of some form, while allowing access to some statistics or aggregate information of reasonable quality. In most settings under consideration, a family of queries to the database is allowed, and then noise can be applied to the data itself, or to the result of the query, or to both (see examples in [20], [21], [3], [25], [2], [7], [8], [6] and their many references). An important parts of these works include defining notions of individual privacy, finding mechanisms which ensure privacy under general families of queries and/or proving that no such mechanisms exist for other families of queries. Most impossibility results are about noise applied to query answers, as e.g. in [7] who show how (almost all) the database can be recovered if the noise is not sufficiently high to be practically useless. Our work also shows that under some (quite general) conditions information can be recovered.

Dimension reduction: High dimension of data is often considered a curse for efficient algorithms. Often the best known (or even best possible) running time is exponential in the dimension. One common remedy used in numerous algorithms for such data is *dimension reduction*. The most famous one is the Johnson-Lindenstrauss reduction [14], showing that a random linear projection of k points in Euclidean space to a subspace of dimension $O(\log k)$ nearly preserves all their pairwise Euclidean distances. Similar reductions in many settings and problems allow much faster processing of the data in the lower-dimensional space, and then "lifting" the solution to the original space, often losing something in its quality. Typically, reductions are random, and are done once (though there are exceptions to both: sometimes reductions move to a subspace determined by some top singular eigenvectors, and e.g. in the aforementioned Gaussian mixtures learning, many 1-dimensional projections are used). Also in our algorithms we use many different dimension reductions, projecting the data onto very different small subsets of the coordinates. But for the problems we deal with random projections are useless in general, and judicious choice is needed to guarantee all properties of our PID graphs.

Identification: Unique identifiers, IDs for short, are prevalent for people, animals, cars, computers and even clothing and food items today. Often these IDs are artificial *additions* to the individuals, like driver licenses and barcodes. But sometimes such additions are impossible or inconvenient. Another possibility is to let IDs consist of a genuine subset of the features of the individual. One example is fingerprints and retinal scans used for people. A second one is data records in databases, which typically have their IDs being a single key feature of a record. When space is not an issue, or when the data is structured in advance to have short IDs, the use of unique IDs is easy and fully justified. But in situations where the set of individuals is unstructured, short IDs for every individual, distinguishing it from all others may not always be possible. In this paper we introduce the idea of using *partial IDs*, or PIDs, which are subsets of features per individual that may not distinguish it completely from all others, but in which some control is available about its set of imposters. In this work we had specific requirements (arising from analyzing our algorithms) from our set of PIDs, and we could satisfy them all with very short PIDs (even when no short IDs exist). In different settings one may need different properties, and we just put forth the possibility that PIDs may be of use elsewhere. A particular setting of large collections of unstructured (and noisy/lossy) set of sequences one may consider are those arising in computational biology, e.g. from some sequencing or phylogeny of large collections of large molecules like DNA and proteins. Is there any use of PIDs for the algorithmic problems such data gives rise to? A related work is [16] where certain small-dimensional witnesses are used to study phylogenetic mixtures.

Reconstruction: Reconstruction problems various settings abound, and we just mention a few. The famous graph reconstruction conjecture of Kelly and Ulam (see e.g. [24]), now open for half a century, states that an unlabeled n-vertex graph is always uniquely defined by its (unordered multi-set of) n induced subgraphs obtained by removing each of the vertices. A far less famous but very intriguing is Mossell's subsequence problem [22] arising from computational biology [11], asking if an n-bit sequence is uniquely identified (with, say, a noticeable gap of $1/n$) by the probability measure on it subsequences, obtained by independently removing each symbol with probability $1/2$ (and removing the gaps!). Both problems are information theoretic about reconstruction of data from lossy models *different* than the ones considered here, which seem difficult even if we allow inefficient algorithms! A different reconstruction problem, in which efficiency is important, is Blum's junta learning problem, which can be cast as a recovery from a noisy model different than ours.

F. Organization

The technical part of paper essentially has two parts. The first part, Section II, contains the full discussion of PID graphs: their construction with good parameters, and a tradeoff showing near optimality of the parameters we achieve. The second part, Section III, contains our recovery algorithms and their analysis.

Due to space limitations many proofs are not included in this version of the paper. The full version can be found on the authors' homepages.

II. PID GRAPHS

In this section we study PID graphs. We first describe our main result, an efficient algorithm for constructing PID graphs of logarithmic width and depth for any set of vectors. As noted, these bounds only imply a quasi-polynomial upper bound on the cost. We next describe a set of vectors for which any logarithmic width PID graph has super-polynomial cost.

Our construction of an efficient PID graph is iterative: Start with a PID graph of logarithmic width in a similar fashion to as described in Section I-D. Then, iteratively improve it. Each iteration consists of some local improvement based on the procedure "extend" below. Finally, when no more extensions are possible, the overall structure is (perhaps surprisingly) as desired.

We start with setting some useful notation. Fix a set $V \subseteq \Sigma^n$ of size $|V| = k$. We construct PIDs S_v for the vectors $v \in V$ by an iterative procedure, and so we redefine the set of imposters $I(v, S)$ with respect to any subset $S \subseteq [n]$ to explicitly depend on S. As before it is the set of all $u \in V$ (including v itself) which have the same pattern on S as v does, namely

$$I(v, S) = \{u \in V : u[S] = v[S]\}.$$

By convention, $v[\emptyset] = u[\emptyset]$ for all v, u.

A. Extension of a PID

A key procedure in the algorithm is a "greedy extension" of a PID: For $v \in V$ and $S \subseteq [n]$, we seek to successively add coordinates to S which shrink the number of imposters by a factor of 2, as long as we can. Formally, this procedure $\mathsf{Extend}(v, S)$ produces a superset of S and is defined recursively as follows (in extend V is a fixed set of vector that the procedures knows).

<u>Extend algorithm</u>

```
Input: A vector $v \in V$ and a set $S \subseteq [n]$.

Recursion base:
    Let $J$ be the set of $i$ in $[n] \setminus S$ so that
        $|I(v, S \cup \{i\})| \leq |I(v, S)|/2.$
    If $J = \emptyset$, output
        $\mathsf{Extend}(v, S) = S.$

Recursive step:
    Otherwise, let[8] $i = \min J$, and compute
        $\mathsf{Extend}(v, S) = \mathsf{Extend}(v, S \cup \{i\}).$
```

Call S *maximal* for v if $\mathsf{Extend}(v, S) = S$. A simple but crucial claim summarizes the properties of this procedure.

Claim 9 (Properties of extension)**.** *For every $v \in V$ and $S \subseteq [n]$, if $T = \mathsf{Extend}(v, S)$ then*

- *$S \subseteq T$,*
- *$|I(v, T)| \leq |I(v, S)| \cdot 2^{|S| - |T|}$,*
- *T is maximal for v, and*
- *If $u \neq v$ and $u \in I(v, T)$, then T is* not *maximal for u.*

Let us give some intuition for the algorithm constructing the PID graph before formally describing it. Our algorithm will perform many such Extend sub-routines. We initialize it by setting $S_v = \mathsf{Extend}[v, \emptyset]$ for all v. Then we repeatedly and judiciously choose a vector u and a subset S (not necessarily its PID), and assign the new S_u to be $\mathsf{Extend}(u, S)$. We insist that whenever we activate $\mathsf{Extend}(v, S)$, we have

[8] Any choice of i in J will do. Minimum is chosen for concreteness.

$|I(v,S)| \le k2^{-|S|}$. This guarantees that each S_v during the running of the algorithm, and especially the final ones, has size at most $\log k$ and so the graph produced has small width.

To take care of the depth we ensure that eventually, for every v, *every* imposter $u \neq v$ has a PID S_u that is strictly larger than S_v. Thus along every directed path PID-size increases, and as it has maximum value $\log k$, no path can be longer than $\log k$. This condition also ensure that no cycles exist in the graph. The main question is how to ensure this "monotonicity" property (which certainly does not hold initially). The answer is simple: fix it whenever it is violated. Specifically, as long as some imposter $u \neq v$ in $I(v, S_v)$ with $|S_u| \le |S_v|$ exists, replace S_u with $\mathsf{Extend}(u, S_v)$. The properties of extension in the claim above guarantee a new S_u which is longer than the current S_v (and also longer than the previous S_u).

This fixings continue as long as necessary, so all we need to argue is termination and efficiency. The overall PID-size strictly increases with every such step. Since their total size cannot exceed $k \log k$ by the width bound, we get a $k \log k$ upper bound on the number of invocations of Extend. The bound on the running time of the algorithm easily follows[9].

We now formally define the algorithm and analyze it.

PID construction algorithm

```
Input:
    A set $V \subseteq \Sigma^n$.
Initialize:
    For every $v \in V$ set $S_v = \mathsf{Extend}(v, \emptyset)$.
Iterate:
    While there exists $v$ and $u \neq v$ with $u \in I(v, S_v)$
    and $|S_u| \le |S_v|$ set
        $S_u = \mathsf{Extend}(u, S_v)$.
Output:
    The set of final PIDs $S_v$ and their graph $G$.
```

Theorem 10. *The algorithm above terminates in at most $k \log k$ iterations, and produces a PID graph G with $\mathsf{width}(G) \le \log k$ and $\mathsf{depth}(G) \le \log k$ (and so also $\mathsf{cost}(G) \le k^{\log k}$).*

The theorem follows from the following claim, that summarizes the invariants maintained by the algorithm, and conclusions from them. The invariants follow from the properties of Extend in Claim 9 above, and can be easily proved by induction on the iterations of the algorithm.

Claim 11 (Properties of the PID algorithm)**.**

- *If $u \neq v$ in V were chosen in some iteration, then $|\mathsf{Extend}(u, S_v)| > |S_v| \ge |S_u|$.*
- *The total length of PIDs, $\sum_{v \in V} |S_v|$, strictly increases at every iteration.*
- *For every $v \in V$ and every S_v that is obtained while the algorithm runs, $1 \le |I(v, S_v)| \le 2^{-|S_v|} \cdot |V|$. Specifically, the size of S_v never exceeds $\log k$.*
- *The total length of PIDs never exceeds $k \log k$.*
- *The algorithm halts after at most $k \log k$ iterations.*
- *Let G be the PID graph the algorithm computed. Then, along every path the size of the corresponding PIDs strictly increases.*

III. Recovery algorithms

In this section we describe two distribution recovery algorithms, for lossy and for noisy samples (and also remark on recovery from the hybrid model combining loss and noise). That is, we prove Theorem 6. In both cases, there is some known database V and we wish to estimate a probability distribution π on it, using random imperfect samples. We assume that the algorithms get a PID graph G of V, and specifically a PID S_v for all v in V.

The two algorithms for the two cases share a two-phase structure: an estimation phase and an aggregation phase. In the estimation phase, the PID graph is used to extract a list of quantities that are useful to estimate, and these estimations are obtained. In the aggregation phase, the estimations obtained are numerically combined into an estimation of π.

We now discuss the two phases in more detail. Both use the PID graph G, and specifically will estimate π by first estimating an auxillary vector p, which for every v in V is defined by

$$p(v) = \sum_{u \in I(v)} \pi(u),$$

where $I(v)$ is the set of imposters of u in G. We shall see how all parameters of the PID graph, width, depth and cost affect the performance of these two phases.

In the *estimation phase*, the algorithm obtains an (L_∞) estimate p' of p. As we shall see, these estimates amount to solving k distribution recovery problems in small dimension (at most $\mathsf{width}(G)$), by projecting the data to the coordinates of each of the PIDs S_v. As $\mathsf{width}(G) \le \log k$, these recovery algorithms can afford to use time exponential in the dimension! Computing p' from lossy samples in this regime is simple, and uses only the well-known Hoeffding bound. Computing p' from noisy samples, even in this regime, is more elaborate (and, in fact, a priori may be impossible). In both cases, a simple parallel implementation of the algorithm is possible. In both cases, the quality of the approximation p' of p needed (which determines running time and number of samples) follows from the discussion of the aggregation phase below.

In the aggregation phase, the estimate p' of p, is used to compute an estimate π' of π. Clearly, the vector p is obtained from π by a linear transformation: there exists a $k \times k$ real matrix $M = M(G)$ so that

$$p = M\pi.$$

[9] An obvious implementation gives a running time of $(n+k)^2 \mathsf{poly} \log k$. It would be interesting to find a near-linear-time implementation.

Since G is acyclic, the matrix M is triangular with 1's on the diagonal: π can be computed from p by back substitution using induction on the structure of G,

$$\pi(v) = p(v) - \sum_{u \in I(v) : u \neq v} \pi(u). \tag{1}$$

This process is easily parallelized, and can be seen to take exactly $\mathsf{depth}(G)$ rounds. The main issue is, of course, that we do not have p available but rather an approximation p' of it. A priori such a matrix M may have exponentially small condition number (in k), and thus may require exponentially good approximation p' of p, which will, in turn, cause the first phase to require exponentially many samples. We show, however, that the error aggregation depends only on $\mathsf{cost}(G)$. Lemma 12 below states that using p' instead of p in (1) above yields an estimate π' satisfying

$$\|\pi - \pi'\|_\infty \leq \mathsf{cost}(G) \, \|p - p'\|_\infty . \tag{2}$$

Specifically, we only need to estimate p up to an additive factor $\alpha/\mathsf{cost}(G)$ in order to get an α approximation of π. When G has cost $k^{\log k}$, as we achieve, such estimates can be obtained in quasi-polynomial running time (instead of exponential).

Summarizing, the high-level description is:

Recovery algorithm

Input:

A database $V \subseteq \Sigma^n$ and a PID graph G for V.

Estimation:

Obtain a $(\alpha/\mathsf{cost}(G))$-estimate p' of p.

Aggregation:

Compute $\pi' = M^{-1}p'$, using (1) according to the topological order on G.

Output:

The estimate π' of a distribution π on V.

We note that $\mathsf{cost}(G)$ can be easily computed, and in many cases it can be much smaller than the worst-case super-polynomial one.

We now formally discuss the two phases. In the estimation phase (which is the only phase that depends on the underlying sampler), we consider each sampler model separately. The aggregation phase is the same for both.

A. The estimation phase

We now show how both algorithms compute an estimate p' of p: for every v in V, it will hold that

$$\Pr[|p(v) - p'(v)| \geq \varepsilon] \leq \delta,$$

with

$$\varepsilon = \alpha/\mathsf{cost}(G)$$

(a simple union bound implies that this estimate jointly holds for all v, with high probability).

In Section I-B of the introduction we intuitively explained how each of the lossy and noisy cases are handled when n is small. In this section we unify the two, and actually define and analyze a much more general sampler that extends both. This generality also helps to understand the essence of this phase of the algorithm.

A general sampler: We start by describing a general sampling procedure over Σ_2^n, given a distribution π on Σ_1^n. As we shall explain, the lossy and noisy cases are specific instances of it. The sampler is determined by n stochastic *transition* matrices $T_1, \ldots, T_n$, each in $\mathbb{R}^{\Sigma_1 \times \Sigma_2}$. These n matrices and π define a distribution on samples v' in Σ_2^n. First, choose v according to π. Then, for each i in $[n]$, if $v_i = \sigma_1$ in Σ_1, then v'_i is random: it is σ_2 in Σ_2 with probability $(T_i)_{\sigma_1,\sigma_2}$.

Let us consider the noisy case for example. In this case, $\Sigma_1 = \Sigma_2 = \{0, 1\}$, and all transition matrices $T_1, \ldots, T_n$ are equal to

$$T_{noise} = \begin{bmatrix} T_{0,0} & T_{0,1} \\ T_{1,0} & T_{1,1} \end{bmatrix} = \begin{bmatrix} (1+\nu)/2 & (1-\nu)/2 \\ (1-\nu)/2 & (1+\nu)/2 \end{bmatrix}. \tag{3}$$

A simple generalization yields, e.g., a noise model where each coordinate i in $[n]$ is perturbed with a different probability $(1 - \nu_i)/2$. As we shall see, our algorithm works in this case as well.

The estimation algorithm and its analysis: We now move to describe our algorithm for estimating p using a general sampler defined by $T_1, \ldots, T_n$. Due to space limitations, we shall not provide full proofs. Start by obtaining a large set V' of random samples $V' = \{v'_1, \ldots, v'_t\}$. The exact value of t will be determined later on. For each $v \in V$, use the PID S_v to perform dimension reduction on this sample to estimate $p(v)$. These computations can all be done in parallel. Fix v in V, and denote $S = S_v$. We describe how to estimate $p(v)$.

The transition matrix of vectors on the coordinates S is the tensor product

$$T = T^{(S)} = \otimes_{i \in S} T_i .$$

That is, for v_1 in Σ_1^S and v_2 in Σ_2^S, the number T_{v_1,v_2} is the probability of seeing v_2 given that v_1 was chosen (using π).

For every v_2 in Σ_2^S, let $q(v_2)$ be the probability that v_2 is the output of the sampler (when restricted to entries in S). As long as T is invertible, there is a unique r so that

$$q = Tr.$$

What is r? The number $r(v_1)$, for every v_1 in Σ_1^S, is the probability that a vector v that is chosen according to π is so that $v[S] = v_1$. Specifically,

$$p(v) = r(v[S]).$$

In other words, we just wish to estimate one entry in r. This can easily be done, by inverting T. And, as long as T does not have too small singular values, the number of samples t we need is not too large.

Comment: The same idea works also when the sampling procedure is only $\mathsf{width}(G)$-wise independent, since all calculations are based on local data (concerning sets of entries of size at most $\mathsf{width}(G)$).

B. The aggregation phase: error bounds

The only part that is missing is a bound on the increase in error. Let M be the matrix defined by the PID graph G for V as above. That is, $p = M\pi$ means

$$p(v) = \sum_{u \in I(v)} \pi(u).$$

We prove the following estimate on the increase of error M^{-1} can cause.

Lemma 12. *For any vector w,*

$$\left\|M^{-1}w\right\|_\infty \leq \mathsf{cost}(G) \left\|w\right\|_\infty .$$

The lemma clearly implies (2) and follows by induction.

Acknowledgements

We thank Zeev Dvir for helpful discussions. We thank Sanjeev Arora, Avrim Blum, Russell Impagliazzo, Dick Karp, Yishay Mansour and Elchanan Mossel for helpful comments on an earlier version of this work.

References

[1] D. Achlioptas and F. McSherry. On spectral learning of mixtures of distributions. COLT, pages 458–469, 2005.

[2] R. Agrawal, and R. Srikant. Privacy-preserving data mining. ACM SIGMOD Record 29 (2), pages 439–450, 2000.

[3] L. Beck. A security mechanism for statistical data bases. ACM Transactions of Databases 5 (3), pages 316–338, 1980.

[4] A. Blum, A. Coja-Oghlan, A. Frieze, and S. Zhou. Separating populations with wide data: A spectral analysis. Electron. J. Statist. 3, pages 76–113, 2009.

[5] Z. Dvir, A. Rao, A. Wigderson and A. Yehudayoff. Restriction access. ITCS, pages 19–33, 2012.

[6] C. Dwork, F. McSherry, K. Nissim, and A. Smith. Calibrating noise to sensitivity in private data analysis. TCC, pages 265–284, 2006.

[7] I. Dinur and K. Nissim. Revealing information while preserving privacy. PODS, pages 202–210, 2003.

[8] C. Dwork and A. Smith. Differential privacy for statistics: what we know and what we want to learn. NCHS/CDC Data Confidentiality Workshop, 2008.

[9] J. Feldman, R. O'Donnell and R. Servedio. Learning mixtures of product distributions over discrete domains SIAM Journal on Computing 37 (5), pages 1536–1564, 2008.

[10] O. Goldreich and L. Levin. A generic hardcore predicate for any one-way function. STOC, pages 25–30, 1989.

[11] T. Holenstein, M. Mitzenmacher, R. Panigrahy, and U. Wieder. Trace reconstruction with constant deletion probability and related results. SODA, pages 389–398, 2008.

[12] R. Impagliazzo. Private communication.

[13] J. Jackson. An efficient membership-query algorithm for learning DNF with respect to the uniform distribution. Journal of Computer and System Sciences, 55, pages 414–440, 1997.

[14] W. Johnson and J. Lindenstrauss. Extensions of Lipschitz mappings into a Hilbert space. Contemporary Mathematics 26, pages 189–206, 1984.

[15] E. Kushilevitz and Y. Mansour. Learning decision trees using the fourier spectrum. SIAM J. Computing 22 (6), pages 1331–1348, 1993.

[16] F. A. Matsen, E. Mossel and M. Steel. Mixed-up trees: the structure of phylogenetic mixtures. Bull Math Biol. 70 (4), pages 1115-39, 2008.

[17] A. Moitra and G. Valiant. Settling the polynomial learnability of mixtures of gaussians. FOCS, pages 93–102, 2010.

[18] A. Kalai, A. Moitra and G. Valiant. Disentangling Gaussians. Communications of the ACM 55 (2), pages 113–120, 2012.

[19] M. Kearns, Y. Mansour, D. Ron, R. Rubinfel, R. E. Schapire and L. Sellie. On the learnability of discrete distributions. STOC, pages 273–282, 1994.

[20] C. K. Liew, U. J. Choi and C. J. Liew. A data distortion by probability distribution. Communications of the ACM 42 (10), pages 89–94, 1999.

[21] E. Lefons, A. Silvestri and F. Tangorra. An analytic approach to statistical databases. International Conference on Very Large Data Bases, pages 260–274, 1983.

[22] E. Mossell. The subsequence problem. Unpublished

[23] E. Mossell and S. Roch. Phylogenetic mixtures: concentration of measure in the large-tree limit. Annals of Applied Probability, to appear, 2012.

[24] C. Nash-Williams. The reconstruction problem. Selected Topics in Graph Theory I, pages 205–236, 1978.

[25] J. Traub, Y.Yemini and H. Wozniakowski. The statistical security of a statistical database. ACM Transactions on Database Systems 9 (4), pages 672–679, 1984.

[26] S. L. Warner. Randomized response: a survey technique for eliminating evasive answer bias. Journal of the American Statistical Association 60, pages 63–69, 1965.

[27] G. R. Wood. Binomial mixtures: geometric estimation of the mixing distribution. Annals of Statistics 27 (5), pages 1706–1721, 1999.