The Parity Problem in the Presence of Noise, Decoding Random Linear Codes, and the Subset Sum Problem ⋆ (Extended Abstract)

Vadim Lyubashevsky

University of California at San Diego, La Jolla CA 92093, USA
vlyubash@cs.ucsd.edu

Abstract. In [2], Blum *et al.* demonstrated the first sub-exponential algorithm for learning the parity function in the presence of noise. They solved the length-n parity problem in time $2^{O(n/\log n)}$ but it required the availability of $2^{O(n/\log n)}$ labeled examples. As an open problem, they asked whether there exists a $2^{o(n)}$ algorithm for the length-n parity problem that uses only $poly(n)$ labeled examples. In this work, we provide a positive answer to this question. We show that there is an algorithm that solves the length-n parity problem in time $2^{O(n/\log\log n)}$ using $n^{1+\epsilon}$ labeled examples. This result immediately gives us a sub-exponential algorithm for decoding $n \times n^{1+\epsilon}$ random binary linear codes (i.e. codes where the messages are n bits and the codewords are $n^{1+\epsilon}$ bits) in the presence of random noise. We are also able to extend the same techniques to provide a sub-exponential algorithm for dense instances of the random subset sum problem.

1 Introduction

In the length-n parity problem with noise, there is an unknown to us vector $c \in \{0,1\}^n$ that we are trying to learn. We are also given access to an oracle that generates examples a_i and labels l_i where a_i is uniformly distributed in $\{0,1\}^n$ and l_i equals $c \cdot a_i (\text{mod } 2)$ with probability $\frac{1}{2}+\eta$ and $1-c\cdot a_i(\text{mod } 2)$ with probability $\frac{1}{2}-\eta$. The problem is to recover c. In [2], Blum, Kalai, and Wasserman demonstrated the first sub-exponential algorithm for solving this problem. They gave an algorithm that recovers c in time $2^{O(n/\log n)}$ using $2^{O(n/\log n)}$ labeled examples for values of η is greater than 2^{-n^δ} for any constant $\delta < 1$. An open problem was whether it was possible to have an algorithm with a sub-exponential running time when only given access to a polynomial number of labeled examples. In this work, we show that by having access to only $n^{1+\epsilon}$ labeled examples, we can recover c in time $2^{O(n/\log\log n)}$ for values of η greater than $2^{-(\log n)^\delta}$. So the penalty we pay for using fewer examples is both in the time and the error tolerance.

⋆ Research supported in part by NSF grant CCR-0093029

The parity problem in the presence of noise is equivalent to the problem of decoding random binary linear codes in the presence of random noise. Suppose that A is a random $n \times m$ boolean matrix, and let $l = cA$ for some binary string c of length n. Now flip each bit of l with probability $\frac{1}{2} - \eta$, and call the resulting bit string l'. The goal is to recover c given the matrix A and the string l'. Notice that we can just view every column of A as an example a_i and view the ith bit of l' as the value of $c \cdot a_i (\text{mod } 2)$ which is correct with probability $\frac{1}{2} + \eta$. So this is exactly the length n parity problem in the presence of noise where we are given m labeled examples. In this application, being able to solve the length n parity problem with fewer examples is crucial because we don't really have an oracle that will provide us with as many labeled examples as we want. The number of labeled examples we get is exactly m, the number of columns of A. Our result for learning the parity function provides the first algorithm which can decode $n \times n^{1+\epsilon}$ random binary linear codes in sub-exponential time ($2^{O(n/\log\log n)}$) in the presence of random noise. Unfortunately, the techniques do not extend to providing a sub-exponential algorithm for decoding $n \times \alpha n$ random binary linear codes for constant $\alpha > 1$, and we leave that as an open problem.

We then show how to apply essentially the same techniques to obtain a sub-exponential time algorithm for high density random instances of the subset sum problem. In the random subset sum problem, we are given n random integers $(a_1, \ldots, a_n)$ in the range $[0, M)$, and a target t. We are asked to produce a subset of the a_i's whose sum is t modulo M. We say that the instance of the problem is high density if $M < 2^n$. For such values of M, a solution to the problem exists with extremely high probability. It has been shown that solving worst case instances of some lattice problems reduces to solving random instances of high density subset sum [1],[10]. Thus it is widely conjectured that solving random high density instances of subset sum is indeed a hard problem.

Random subset sum instances where $M = 2^{\Omega(n^2)}$ have been shown to be solvable in polynomial time [8],[4], and the same techniques extend to provide better algorithms for values of $M > 2^{n^{1+\epsilon}}$. But the techniques do not carry over to high density instances where $M < 2^n$. A recent algorithm by Flaxman and Przydatek [3] gives a polynomial time solution for instances of random subset sum where $M = 2^{O(\log^2 n)}$. But for all other values of M between $2^{\log^2 n}$ and 2^n, the best algorithms work in time $O(\sqrt{M})$[11]. In our work, we provide an algorithm that works in time $M^{O(\frac{1}{\log n})}$ for all values of $M < 2^{n^\epsilon}$ for any $\epsilon < 1$. So our algorithm is a generalization of [3]. Technical difficulties do not allow us to extend the techniques to provide an algorithm that takes time $M^{O(\frac{1}{\log n})}$ when $M = 2^{\epsilon n}$ for $\epsilon < 1$, and we leave this as an open problem.

1.1 Conventions and Terminology

We will use the phrase "bit b gets corrupted by noise of rate $\frac{1}{2} - \eta$" to mean that the bit b retains its value with probability $\frac{1}{2} + \eta$ and gets flipped with probability $\frac{1}{2} - \eta$. And this probability is completely independent of anything else. Also, all logarithms are base 2.

1.2 Main Ideas and Main Results

In order for the Blum, Kalai, Wasserman algorithm to work, it must have access to a very large (slightly sub-exponential) number of labeled examples. What we will do is build a function to provide this very large number of labeled examples to their algorithm, and use their algorithm as a black box. We are given $n^{1+\epsilon}$ pairs (a_i, l_i) where a_i is uniformly distributed on $\{0,1\}^n$ and l_i is $c \cdot a_i (\text{mod } 2)$ corrupted by noise of rate $\frac{1}{2} - \eta$. Instead of treating this input as labeled examples, we will treat it as a random element from a certain family of functions that maps elements from some domain X to the domain of labeled examples. To be a little more precise, say we are given ordered pairs $(a_1, l_1), \ldots, (a_{n^{1+\epsilon}}, l_{n^{1+\epsilon}})$. Now consider the set $X = \{x \in \{0,1\}^{n^{1+\epsilon}} | \sum_i x_i = \lceil \frac{2n}{\epsilon \log n} \rceil\}$ (i.e all bit strings of length $n^{1+\epsilon}$ that have exactly $\lceil \frac{2n}{\epsilon \log n} \rceil$ ones). Our function works by selecting a random element $x \in X$ and outputting $a' = \bigoplus x_i a_i$ and $l' = \bigoplus x_i l_i$ (where $\oplus$ is the xor operation, and x_i is the ith bit of x).

What we will need to show is that the distribution of the a' produced by our function is statistically close to uniform on $\{0,1\}^n$, that l' is the correct value of $c \cdot a' (\text{mod } 2)$ with probability approximately $\frac{1}{2} + \eta^{\frac{2n}{\epsilon \log n}}$, and that the correctness of the label is almost independent of a'. This will allow us to use the function to produce a sub-exponential number of random looking labeled examples that the Blum, Kalai, Wasserman algorithm needs.

The idea of using a small number of examples to create a large number of examples is somewhat reminiscent of the idea used by Goldreich and Levin in [5]. In that work, they showed how to learn the parity function in polynomial time when given access to a "membership oracle" that gives the correct label for a $\frac{1}{2} + \frac{1}{poly(n)}$ fraction of the 2^n possible examples. A key step in their algorithm was guessing the correct labels for a logarithmic number of examples and then taking the xor of every possible combination of them to create a polynomial number of correctly labeled examples. The main difference between their problem and ours is that they were allowed to pick the examples with which to query the oracle, while in our formulation of the problem, the examples are provided to us randomly.

We prove the following theorem in Section 3:

Theorem 1. *We are given $n^{1+\epsilon}$ ordered pairs (a_i, l_i) where a_i are chosen uniformly and independently at random from the set $\{0,1\}^n$ and for some $c \in \{0,1\}^n$, $l_i = c \cdot a_i (mod\, 2)$ corrupted by noise of rate $\frac{1}{2} - \eta$. If $\eta > 2^{-(\log n)^\delta}$ for any constant $\delta < 1$, then there is an algorithm that can recover c in time $2^{O(n/\log\log n)}$. The algorithm succeeds with error exponentially small in n.*

In Section 4, we move on to the random subset sum problem. We can no longer use the Blum, Kalai, Wasserman algorithm as a black box, but we do use an algorithmic idea that was developed by Wagner [13], and is in some sense a more general version of an algorithm in [2]. Wagner considers the following problem: given n lists each containing $M^{\frac{1}{\log n}}$ independent integers uniformly distributed in the interval $[0, M)$ and a target t, find one element from each list

such that the sum of the elements is $t(mod\, M)$. Wagner shows that there exists an algorithm that in time $nM^{O(\frac{1}{\log n})}$, returns a list of solutions. The number of solutions that the algorithm returns is a random variable with expected value 1. That is, we expect to find one solution. Notice that this does not imply that the algorithm finds a solution with high probability because, for example, it can find 2^n solutions with probability 2^{-n} and find 0 solutions every other time. By inspecting Wagner's algorithm, there is no reason to assume such pathological behavior, and thus the algorithm works well as a cryptanalytic tool (as was intended by the author). We make some modifications to the parameters of the algorithm and are able to obtain a proof that the algorithm indeed succeeds with high probability in finding a solution. So if $M = 2^{n^\epsilon}$ and we are given n lists each containing $2^{O(n^\epsilon/\log n)}$ numbers in the range $[0, M)$, then we can find one number from each list such that their sum is the target t in time $2^{O(n^\epsilon/\log n)}$. We will be using this algorithm as a black box to solve the random subset sum problem.

We will give a sketch of the proof for the below theorem in Section 4. For complete details, the reader is referred to [9] for a full, self-contained version of the paper describing the algorithm.

Theorem 2. *Given n numbers, $a_1, \ldots, a_n$, that are independent and uniformly distributed in the range $[0, M)$ where $M = 2^{n^\epsilon}$ for some $\epsilon < 1$, and a target t, there exists an algorithm that in time $2^{O(n^\epsilon/\log n)}$ and with probability at least $1 - 2^{-\Omega(n^\epsilon)}$ will find $x_1, \ldots, x_n \in \{0, 1\}$ such that $\sum_{i=1}^{n} a_i x_i = t(mod\, M)$.*

2 Preliminaries

2.1 Statistical Distance

Statistical distance is a measure of how far apart two probability distributions are. In this subsection, we review the definition and some of the basic properties of statistical distance. The reader may refer to [12] for reference.

Definition 1. *Let X and Y be random variables over a countable set A. The statistical distance between X and Y, denoted $\Delta(X, Y)$, is*

$$\Delta(X, Y) = \frac{1}{2} \sum_{a \in A} |Pr[X = a] - Pr[Y = a]|$$

Proposition 1. *Let $X_1, \ldots, X_k$ and $Y_1, \ldots, Y_k$ be two lists of independent random variables. Then*

$$\Delta((X_1, \ldots, X_k), (Y_1, \ldots, Y_k)) \leq \sum_{i=1}^{k} \Delta(X_i, Y_i)$$

Proposition 2. *Let X, Y be two random variables over a set A. For any predicate $f : A \to \{0, 1\}$,*

$$|Pr[f(X) = 1] - Pr[f(Y) = 1]| \leq \Delta(X, Y)$$

2.2 Leftover Hash Lemma

In this sub-section, we state the Leftover Hash Lemma [7]. We refer the reader to [12] for more information on the material in this sub-section, and to [7] for the proof of the Leftover Hash Lemma.

Definition 2. *Let $\mathcal{H}$ be a family of hash functions from X to Y. Let H be a random variable whose distribution is uniform on $\mathcal{H}$. We say that $\mathcal{H}$ is a* ***universal*** *family of hash functions if for all $x, x' \in X$ with $x \neq x'$,*

$$Pr_{H \in \mathcal{H}}[H(x) = H(x')] \leq \frac{1}{|Y|}$$

Proposition 3. ***(Leftover Hash Lemma)*** *Let $X \subset \{0,1\}^n, |X| \geq 2^l$ and $|Y| = \{0,1\}^{l-2e}$ for some $e > 0$. Let U be the uniform distribution over Y. If $\mathcal{H}$ is a universal family of hash functions from X to Y, then for all but a $2^{-\frac{e}{2}}$ fraction of the possible $h_i \in \mathcal{H}$, $\Delta(h_i(x), U) \leq 2^{-\frac{e}{2}}$ where x is chosen uniformly at random from X.*

2.3 Learning Parity

The below result was proved in [2] and we will be using it as a black box.

Proposition 4. *(Blum, Kalai, Wasserman) For any integers a and b such that $ab \geq n$, if we are given poly $\left(\left(\frac{1}{2\gamma}\right)^{2^a}, 2^b\right)$ labeled examples uniformly and independently distributed in $\{0,1\}^n$, each of whose labels is corrupted by noise of rate $\frac{1}{2} - \gamma$, then there exists an algorithm that in time poly $\left(\left(\frac{1}{2\gamma}\right)^{2^a}, 2^b\right)$ solves the length-n parity problem. The algorithm succeeds with error exponentially small in n.* □

For $\gamma = 2^{-n^\delta}$ where δ is any positive constant less than 1, we can set a to be any value less than $(1-\delta)\log n$ and set b to any value grater than $\frac{n}{(1-\delta)\log n}$ in order to get an algorithm with running time $2^{O(n/\log n)}$. The examples that we will be constructing in our algorithm will be corrupted by noise of rate more than $\frac{1}{2} - 2^{-n^\delta}$, that is why we will not be able to get the same running time.

3 Solving the Parity Problem With Fewer Examples

The main goal of this section will be to prove Theorem 1. But first, we will prove a lemma which will be used several times in the proof of the theorem.

Lemma 1. *Let $X \subseteq \{0,1\}^{n^{1+\epsilon}}$ such that $|X| > 2^{2n}$ and let $Y = \{0,1\}^n$. Let $\mathcal{H}$ be the family of hash functions from X to Y, where $\mathcal{H} = \{h_a | a = (a_1, \ldots, a_{n^{1+\epsilon}}), a_i \in \{0,1\}^n\}$, and where $h_a(x) = x_1 a_1 \oplus \ldots \oplus x_{n^{1+\epsilon}} a_{n^{1+\epsilon}}$. If h_a is chosen uniformly at random from $\mathcal{H}$, then with probability at least $1 - 2^{-\frac{n}{4}}$, $\Delta(h_a(x), U) \leq 2^{-\frac{n}{4}}$ where x is chosen uniformly at random from X and U is the uniform distribution over Y.*

Proof. First we will show that $\mathcal{H}$ is a universal family of hash functions from X to Y. Let $x, x' \in X$ such that $x \neq x'$ and H be a random variable whose distribution is uniform on $\mathcal{H}$. Since x has to differ from x' in some coordinate, without loss of generality, assume that $x_j = 1$ and $x'_j = 0$ for some j. Then,

$$\begin{aligned} Pr_{H\in\mathcal{H}}[H(x) = H(x')] &= Pr\left[\bigoplus_i x_i a_i = \bigoplus_i x'_i a_i\right] \\ &= Pr\left[a_j \oplus \bigoplus_{i\neq j} x_i a_i = \bigoplus_{i\neq j} x'_i a_i\right] \\ &= Pr\left[a_j = \bigoplus_{i\neq j} (x_i a_i) \oplus \bigoplus_{i\neq j} (x'_i a_i)\right] = \frac{1}{2^n} \end{aligned}$$

with the last equality being true because a_j is chosen uniformly at random and is independent of all the other a_i's.

Now we will apply the leftover hash lemma. Recall that $|X| > 2^{2n}$ and $|Y| = 2^n$. So in the application of Proposition 3, if we set $l = 2n$, and $e = \frac{n}{2}$, then we obtain the claim in the lemma. □

Now we are ready to prove the theorem.

Proof of Theorem 1

Proof. As we alluded to earlier, the way our algorithm works is by generating random-looking labeled examples and using them in the Blum, Kalai, Wasserman algorithm. We now describe the function $h(x)$ for outputting the labeled examples. We are given $n^{1+\epsilon}$ labeled examples (a_i, l_i). Define the set $X \subset \{0,1\}^{n^{1+\epsilon}}$ as $X = \{x \in \{0,1\}^{n^{1+\epsilon}} | \sum_i x_i = \lceil \frac{2n}{\epsilon \log n} \rceil\}$. In other words, the set X consists of all the bit strings of length $n^{1+\epsilon}$ that have exactly $\lceil \frac{2n}{\epsilon \log n} \rceil$ ones. The input to our function will be a random element from X.

$h(x)\{$
  $a' = \bigoplus_i x_i a_i$
  $l' = \bigoplus_i x_i l_i$
  Output (a', l')
$\}$

So an element $x \in X$ essentially chooses which examples and labels to xor together.

Now we will show that with high probability, the function h produces labeled examples such that the a' are distributed statistically close to uniform, and that the l' are the correct values of $a' \cdot c (\text{mod } 2)$ corrupted by noise of rate at most $\frac{1}{2} - \frac{1}{2}\left(\frac{\eta}{2}\right)^{\frac{2n}{\epsilon \log n}}$.

To prove that the distribution of a' is statistically close to uniform on the set $\{0,1\}^n$, we need to think of the $n^{1+\epsilon}$ a_i given to us as a random element

of a family of hash functions $\mathcal{H}$ defined in Lemma 1. And this hash function is mapping elements from X to $\{0,1\}^n$. Notice that,

$$|X| = \binom{n^{1+\epsilon}}{\lceil \frac{2n}{\epsilon \log n} \rceil} \geq \left(\frac{n^{1+\epsilon}}{\lceil \frac{2n}{\epsilon \log n} \rceil} \right)^{\lceil \frac{2n}{\epsilon \log n} \rceil} > 2^{2n}$$

so now we can apply Lemma 1 and conclude that with high probability the distribution of a' generated by h is statistically close to uniform.

Now we need to find the probability that l' is the correct value of $c \cdot a' (\text{mod } 2)$. For that, we first need to show that with high probability enough of the $n^{1+\epsilon}$ examples we are given are labeled correctly.

Lemma 2. *If we are given $n^{1+\epsilon}$ pairs (a_i, l_i) where l_i is the correct label of a_i with probability $\frac{1}{2} + \eta$, then with probability at least $1 - e^{-(n^{1+\epsilon})\frac{\eta^2}{2}}$, there will be more than $\frac{n^{1+\epsilon}}{2}(1+\eta)$ pairs (a_i, l_i) where l_i is the correct label of a_i.* □

Notice that l' will equal $c \cdot a' (\text{mod } 2)$ if $l' = \bigoplus x_i l_i$ is the xor of an even number of incorrectly labeled examples. Using the next lemma, we will be able to obtain a lower bound for the probability that our random x chooses an even number of mislabeled examples.

Lemma 3. *If a bucket contains m balls, $(\frac{1}{2} + p)m$ of which are colored white, and the rest colored black, and we select k balls at random without replacement, then the probability that we selected an even number of black balls is at least $\frac{1}{2} + \frac{1}{2}\left(\frac{2mp-k+1}{m-k+1}\right)^k$.* □

Lemma 2 tells us that out of $n^{1+\epsilon}$ examples, at least $\frac{n^{1+\epsilon}}{2}(1+\eta)$ are correct. Combining this with the statement of Lemma 3, the probability when we pick $\lceil \frac{2n}{\epsilon \log n} \rceil$ random examples, an even number of them will be mislabeled is at least

$$\begin{aligned}
\frac{1}{2} + \frac{1}{2}\left(\frac{n^{1+\epsilon}\eta - \lceil \frac{2n}{\epsilon \log n} \rceil + 1}{n^{1+\epsilon} - \lceil \frac{2n}{\epsilon \log n} \rceil + 1} \right)^{\lceil \frac{2n}{\epsilon \log n} \rceil} &> \frac{1}{2} + \frac{1}{2}\left(\eta - \frac{\lceil \frac{2n}{\epsilon \log n} \rceil}{n^{1+\epsilon}} \right)^{\lceil \frac{2n}{\epsilon \log n} \rceil} \\
&> \frac{1}{2} + \frac{1}{2}\left(\eta - \frac{2}{\epsilon n^{\epsilon}} \right)^{\frac{2n}{\epsilon \log n}} \\
&> \frac{1}{2} + \frac{1}{2}\left(\frac{\eta}{2} \right)^{\frac{2n}{\epsilon \log n}}
\end{aligned}$$

with the last inequality being true since $\eta > \frac{4}{\epsilon n^{\epsilon}}$.

So far we have shown that our function produces examples a' which are statistically close to uniform and labels l' which are correct with probability at least $\frac{1}{2} + \frac{1}{2}\left(\frac{\eta}{2}\right)^{\frac{2n}{\epsilon \log n}}$. All that is left to show is that the correctness of l' is almost independent of a'. We do this by showing that the distribution of the a' that are

labeled correctly and the distribution of the a' that are labeled incorrectly are both statistically close to the uniform distribution over the set $\{0,1\}^n$.

Consider the sets X_{even} and X_{odd} where X_{even} consists of all the $x \in X$ that choose an even number of mislabeled examples, and X_{odd} consists of all the $x \in X$ that choose an odd number of mislabeled examples. While we of course do not know the contents of X_{even} or X_{odd}, we can lower bound their sizes.

X_{even} contains all the x that choose exactly zero mislabeled examples. Since Lemma 2 says that there are at least $\frac{n^{1+\epsilon}}{2}(1+\eta)$ correctly labeled examples,

$$|X_{even}| > \binom{\frac{n^{1+\epsilon}}{2}}{\lceil \frac{2n}{\epsilon \log n} \rceil} > 2^{2n}$$

and similarly, X_{odd} contains all the x that choose exactly one mislabeled example. So,

$$|X_{odd}| > \binom{\frac{n^{1+\epsilon}}{2}}{\lceil \frac{2n}{\epsilon \log n} \rceil - 1} > 2^{2n}$$

Now we can use Lemma 1 to conclude that the distribution of the a' generated by function $h(x)$ is statistically close to the uniform distribution over the set $\{0,1\}^n$ when x is chosen uniformly from the set X_{even} (and similarly from X_{odd}).

So we have shown that with high probability, our function h outputs labeled examples with distribution statistically close to the distribution of labeled examples chosen uniformly from $\{0,1\}^n$ whose labels are corrupted by noise of rate at most $\frac{1}{2} - \frac{1}{2}\left(\frac{\eta}{2}\right)^{\frac{2n}{\epsilon \log n}}$. Now we can apply the parity function learning algorithm from Proposition 4. If our noise rate is $\eta = 2^{-(\log n)^{\delta}}$ for some constant $\delta < 1$, then we use the algorithm with the following parameters: $a = \kappa(\log\log n)$, $b = \frac{n}{\kappa \log\log n}$, $\gamma = \frac{1}{2}\left(\frac{\eta}{2}\right)^{\frac{2n}{\epsilon \log n}}$, where $\delta + \kappa < 1$. The algorithm will run in time,

$$\begin{aligned}
poly\left(\left(\frac{1}{2\gamma}\right)^{2^a}, 2^b\right) &= poly\left(\left(\frac{1}{\left(\frac{\eta}{2}\right)^{\frac{2n}{\epsilon \log n}}}\right)^{(\log n)^{\kappa}}, 2^{\frac{n}{\kappa \log\log n}}\right) \\
&= poly\left(2^{O\left((\log n)^{\delta}(\frac{2n}{\epsilon \log n})(\log n)^{\kappa}\right)}, 2^{\frac{n}{\kappa \log\log n}}\right) \\
&= poly\left(2^{O\left(n(\log n)^{\delta+\kappa-1}\right)}, 2^{\frac{n}{\kappa \log\log n}}\right) \\
&= 2^{O(\frac{n}{\log\log n})}
\end{aligned}$$

if given access to $2^{O(n/\log\log n)}$ labeled examples. Since our function h generates labeled examples whose distribution is statistically close ($2^{-\Omega(n/4)}$) to the correct distribution, if we generate a sub-exponential number of such pairs, Proposition 1 tells us that the statistical distance is still exponentially small. And Proposition 2 says that if we run the Blum, Kalai, Wasserman algorithm with the labeled examples produced by our function, the success probability of the algorithm only goes down by at most an exponentially small amount. And this concludes the proof of Theorem 1. $\square$

4 The Subset Sum Problem

In this section, we will sketch the proof for Theorem 2. All the details of the proof may be found in [9].

First, we state a theorem about an algorithm that will be used as a black box analogous to the way the Blum, Kalai, Wasserman algorithm was used as a black box in the previous section.

Theorem 3. *Given b independent lists each consisting of $kM^{\frac{2}{\log b}}$ independent uniformly distributed integers in the range $[-\frac{M}{2}, \frac{M}{2})$, there exists an algorithm that with probability at least $1 - be^{-\frac{k}{4b^2}}$ returns one element from each of the b lists such that the sum of the b elements is 0. The running time of this algorithm is $O(b \cdot kM^{\frac{2}{\log b}} \cdot \log(kM^{\frac{2}{\log b}}))$ arithmetic operations.*

Proof. (sketch) The algorithm will be building a tree whose nodes are lists of numbers. The initial b lists will make up level 0 of the tree. Define intervals

$$I_i = \left[-\frac{M^{1-\frac{i}{\log b}}}{2}, \frac{M^{1-\frac{i}{\log b}}}{2} \right)$$

for integers $0 \leq i \leq \log b$. Notice that all the numbers in the lists at level 0 are in the range I_0. Level 1 of the tree will be formed by pairing up the lists on level 0 in any arbitrary way, and for each pair of lists, L_1 and L_2 (there are $\frac{b}{2}$ such pairs), create a new list L_3 whose elements are in the range I_1 and of the form $a_1 + a_2$ where $a_1 \in L_1$ and $a_2 \in L_2$. So level 1 will have half the number of lists as level 0, but the numbers in the lists will be in a smaller range. We construct level 2 in the same way; that is, we pair up the lists in level 1 and for each pair of lists L_1 and L_2, create a new list L_3 whose elements are in the range I_2 and of the form $a_1 + a_2$ where $a_1 \in L_1$ and $a_2 \in L_2$. Notice that if we continue in this way, then level $\log b$ will have one list of numbers in the interval $I_{\log b}$ where each number is the sum of b numbers, one from each of the original b lists at level 0. And since the only possible integer in $I_{\log b}$ is 0, if the list at level $\log b$ is not empty, then we are done. So what we need to prove is that the list at level $\log b$ is not empty with high probability.

Claim: *With probability at least $1 - be^{-\frac{k}{4b^2}}$, for $0 \leq i \leq \log b$, level i consists of $\frac{b}{2^i}$ lists each containing $\frac{k}{4^i} \cdot M^{\frac{2}{\log b}}$ independent, uniformly distributed numbers in the range I_i.*

Notice that proving this claim immediately proves the theorem. To prove the claim, we essentially have to show that when we combine two lists in order to obtain a list containing numbers in a smaller range, the new list still consists of many random, independently distributed numbers. The key to the proof is the following technical lemma:

Lemma 4. *Let L_1 and L_2 be lists of numbers in the range $\left[-\frac{R}{2}, \frac{R}{2}\right)$ and let p and c be positive reals such that $e^{-\frac{c}{12}} < p < \frac{1}{8}$. Let L_3 be a list of numbers $a_1 + a_2$ such that $a_1 \in L_1, a_2 \in L_2$, and $a_1 + a_2 \in \left[-\frac{Rp}{2}, \frac{Rp}{2}\right)$. If L_1 and L_2 each contain*

at least $\frac{c}{p^2}$ *independent, uniformly distributed numbers in* $\big[-\frac{R}{2}, \frac{R}{2}\big)$*, then with probability greater than* $1-e^{-\frac{c}{4}}$*,* L_3 *contains at least* $\frac{c}{4p^2}$ *independent, uniformly distributed numbers in the range* $\big[-\frac{Rp}{2}, \frac{Rp}{2}\big)$*.* □

Using this lemma, it's not too hard to finish the proof of the theorem by induction. □

Corollary 1. *Given* b *independent lists each consisting of* $kM^{\frac{2}{\log b}}$ *independent uniformly distributed integers in the range* $[0, M)$*, and a target* t*, there exists an algorithm that with probability at least* $1-be^{-\frac{k}{4b^2}}$ *returns one element from each of the* b *lists such that the sum of the* b *elements is* $t(mod\, M)$*. The running time of this algorithm is* $O(b \cdot kM^{\frac{2}{\log b}} \cdot \log(kM^{\frac{2}{\log b}}))$*.*

Proof. First, we subtract the target t from every element in the first list. Then we transform every number (in every list) in the interval $[0, M)$ into an equivalent number modulo M in the interval $[-\frac{M}{2}, \frac{M}{2})$. We do this by simply subtracting M from every number greater or equal to $\frac{M}{2}$. Now we apply Theorem 3. Since exactly one number from every list got used, it means that $-t$ got used once. And since the sum is 0, we found an element from each list such that their sum is $t(mod\, M)$. □

Trying to solve the random subset sum problem by using the algorithm implicit in the above corollary is very similar to the position we were in trying to decode random linear codes by using the Blum, Kalai, Wasserman algorithm. In both cases, we had algorithms that solved almost the same problem but required a lot more samples than we had. So just as in the case of random linear codes, the idea for the algorithm for random subset sum will be to combine the small number of elements that we are given in order to create a slightly subexponential number of elements that are almost indistinguishable from uniform, independently distributed ones.

Now we state a lemma which is analogous and will be used analogously to Lemma 1. The proof is very similar to the one for Lemma 1, and something very similar was actually previously shown by Impagliazzo and Naor in [6].

Lemma 5. *Let* $X = \{0,1\}^n$*, and* $Y = \{0, 1, \ldots, M\}$ *for some* $M \leq 2^{n/2}$*. Let* $\mathcal{H}$ *be a family of hash functions from* X *to* Y*, where* $\mathcal{H} = \{h_a | a = (a_1, \ldots, a_n),\ a_i \in Y\}$*, and where* $h_a(x) = \sum x_i a_i (mod\ M)$*. If* h_a *is chosen uniformly at random from* $\mathcal{H}$*, then with probability at least* $1 - 2^{-\frac{n}{4}}$*,* $\Delta(h_a(x), U) \leq 2^{-\frac{n}{4}}$ *where* x *is chosen uniformly at random from* X *and* U *is the uniform distribution over* Y*.* □

Now we can finally sketch the idea to prove Theorem 2. The idea for our algorithm will be to first take the n given numbers modulo $M = 2^{n^\epsilon}$ and break them up into $\frac{1}{2}n^{1-\epsilon}$ groups each containing $2n^\epsilon$ numbers. Then for each group, we generate a list of $n^2 M^{\frac{2}{\log \frac{1}{2} n^{1-\epsilon}}}$ numbers by taking sums of random subsets of the elements in the group. By Lemma 5 and Proposition 1, we can say that with

high probability the lists are not too far from being uniformly distributed. Now that we have lists that are big enough, we can apply Corollary 1 which states that we can find one number from each list such that the sum of the numbers is the target. And since every number in the list is some subset of the numbers in the group from which the list was generated, we have found a subset of the original n numbers which sums to the target.

Proof of Theorem 2

Proof. We will show that the below *SubsetSum* algorithm satisfies the claim in the theorem.

$SubsetSum(a_1, \ldots, a_n, t, M)$ /** Here $M = 2^{n^\epsilon}$
(1) Break up the n numbers into $\frac{1}{2}n^{1-\epsilon}$ groups each containing $2n^\epsilon$ numbers.
(2) For group $i = 1$ to $\frac{1}{2}n^{1-\epsilon}$ do
(3) List $L_i = GenerateListFromGroup(\{a_j | a_j \in group\ i\}, M)$
(4) Apply the algorithm from Corollary 1 to $L_1, \ldots, L_{.5n^{1-\epsilon}}, t, M$.

$GenerateListFromGroup(\{a_1, \ldots, a_m\}, M)$
(5) Initialize list L to be an empty list.
(6) For $i = 1$ to $n^2 2^{\frac{2n^\epsilon}{\log .5n^{1-\epsilon}}}$ do /** Notice that $n^2 2^{\frac{2n^\epsilon}{\log .5n^{1-\epsilon}}} = n^2 M^{\frac{2}{\log \frac{1}{2}n^{1-\epsilon}}}$
(7) Generate m random $x_j \in \{0, 1\}$
(8) Add the number $\sum_{j=1}^{m} a_j x_j (mod\, M)$ to list L.
(9) Return L.

If we assume for a second that the lists $L_1, \ldots L_{.5n^{1-\epsilon}}$ in line (4) consist of independent, uniformly distributed numbers, then we are applying the algorithm from Corollary 1 with parameters $b = \frac{1}{2}n^{1-\epsilon}$ and $k = n^2$. So with probability

$$1 - be^{-\frac{k}{4b^2}} = 1 - \frac{1}{2}n^{1-\epsilon}e^{-\frac{n^2}{4(.5n^{1-\epsilon})^2}} = 1 - e^{-\Omega(n^{2\epsilon})}$$

the algorithm will return one element from each list such that the sum of the elements is $t(mod\, M)$. And this gives us the solution to the subset sum instance. The running time of the algorithm is

$$\begin{aligned} O(b \cdot kM^{\frac{2}{\log b}} \cdot \log(kM^{\frac{2}{\log b}})) &= O((b \cdot kM^{\frac{2}{\log b}})^2) \\ &= O\Big(\Big(\frac{1}{2}n^{1-\epsilon}n^2 M^{\frac{2}{\log .5n^{1-\epsilon}}}\Big)^2\Big) \\ &= O\Big(\Big(\frac{1}{2}n^{1-\epsilon}n^2 2^{\frac{2n^\epsilon}{\log .5n^{1-\epsilon}}}\Big)^2\Big) \\ &= 2^{O\left(\frac{n^\epsilon}{(1-\epsilon)\log n}\right)} = 2^{O\left(\frac{n^\epsilon}{\log n}\right)} \end{aligned}$$

The problem is that each list in line (4) is not generated uniformly and independently from the interval $[0, M)$. But it is not too hard to finish off the proof by showing that the distribution of each list is statistically close to the uniform distribution, and thus our algorithm still has only a negligible probability of failure. $\square$

Acknowledgements. I am very grateful to Daniele Micciancio for his many comments and corrections of an earlier version of this work. I am also very grateful to Russell Impagliazzo for some extremely useful conversations about some issues related to the technical parts of the paper. I would also like to thank the anonymous members of the program committee for many useful suggestions.

References

1. Miklos Ajtai. "Generating hard instances of lattice problems" Proc. 28th Annual Symposium on Theory of Computing, pp. 99-108, 1996
2. Avrim Blum, Adam Kalai, and Hal Wasserman. "Noise-tolerant learning, the parity problem, and the statistical query model" JACM., vol. 50, iss. 4 2003 pp. 506-519
3. Abraham Flaxman and Bartosz Przydatek. "Solving medium-density subset sum problems in expected polynomial time" Proc. of the 22nd Symposium on Theoretical Aspects of Computer Science (STACS) (2005) pp. 305-314
4. Alan Frieze. "On the Lagarias-Odlyzko algorithm for the subset sum problem" SIAM J. Comput., vol. 15 1986 pp. 536-539
5. Oded Goldreich and Leonid A. Levin. "Hard-core predicates for any one-way function." Proc. 21st Annual Symposium on Theory of Computing, pp. 25-32, 1989
6. Russell Impagliazzo and Moni Naor. "Efficient cryptographic schemes provably as secure as subset sum" Journal of Cryptology Volume 9, Number 4, 1996, pp. 199-216
7. Russell Impagliazzo and David Zuckerman. "How to recycle random bits" Proc. 30th Annual Symposium on Foundations of Computer Science, 1989, pp. 248-253
8. Jeffrey C. Lagarias and Andrew M. Odlyzko. "Solving low density subset sum problems", J. of the ACM, Volume 32, 1985 pp. 229-246
9. Vadim Lyubashevsky "On random high density subset sums", Technical Report TR05-007, Electronic Coloquium on Computational Complexity (ECCC), 2005
10. Daniele Micciancio and Oded Regev. "Worst-case to average-case reductions based on gaussian measure" Proc. 45th Annual Symposium on Foundations of Computer Science, pp. 372-381, 2004.
11. Richard Schroeppel and Adi Shamir. "A $T = O(2^{(n/2)})$, $S = O(2^{(n/4)})$ algorithm for certain NP-complete problems." SIAM J. Comput. vol. 10 1981 pp. 456-464
12. Victor Shoup. *A Computational Introduction to Number Theory and Algebra*. Cambridge University Press 2005
13. David Wagner. "A generalized birthday problem" CRYPTO 2002, LNCS, Springer-Verlag, pp. 288-303