The Learning with Errors Problem: Algorithms

1 Algorithms

1.1 Algebraic

This is an attack due to Arora and Ge.

The basic idea is to view an LWE sample $(\mathbf{a}, b := \mathbf{a}^T\mathbf{s} + e)$ where $e \in S \subseteq \mathbb{Z}_q$ as a polynomial equation

$$f_{\mathbf{a},b}(\mathbf{s}) = \prod_{x \in S} (b - \mathbf{a}^T \underline{\mathbf{s}} - x) \bmod q$$

where b, $\mathbf{a}$ are known and $\mathbf{s}$ is treated as the unknown variable (denoted by the underline). Clearly, if $(\mathbf{a}, b)$ is an LWE sample, then $f_{\mathbf{a},b}(\mathbf{s}) = 0 \bmod q$, else it isn't. Solving the system of polynomial equations

$$\left\{ f_{\mathbf{a}_i,b_i}(\mathbf{s}) = 0 \bmod q \right\}_{i=1}^{m}$$

of degree $|S|$ will give us the LWE secret.

This is all good except that solving systems of polynomial equations (even degree-2 equations) is NP-hard. Arora and Ge's observation is that if there are *sufficiently many* equations, one can *linearize* them and that the solution to the resulting linear system will give us the solution to the polynomial system w.h.p.

To see how to do this, note that the degree of the polynomials is $|S|$ (that is, the domain in which the error terms live) and the number of monomials is thus $\binom{n+|S|}{|S|}$. *Linearization* is the basic transformation where one substitutes each monomial by a new variable. Furthermore, if $m \gg \binom{n+|S|}{|S|}$, we have more equations than variables. To begin with, any solution to the polynomial system will be a solution to the linearized system; therefore, $\mathbf{s}$ is a solution. When m is large enough, we can also show that $\mathbf{s}$ is the *unique* solution.

Simplified Proof Intuition. For simplicity, think of S as $\{0, 1\}$ and think of $n = 1$.

Take each sample $(a, b = a \cdot s + e)$ where $e \in \{0, 1\}$ and $a, s \in \mathbb{Z}_q$. This gives us a polynomial equation

$$(b + a \cdot u) \cdot (b + a \cdot u - 1) = 0 \bmod q$$

Writing it out explicitly, we get

$$b(b-1) + (2b-1)a \cdot u + a^2 \cdot u^2 = 0 \bmod q$$

Linearizing this involves replacing u and u^2 by independent variables u_1 and u_2 giving us

$$p(a) = b(b-1) + (2b-1)a \cdot u_1 + a^2 \cdot u_2 = 0 \bmod q \tag{1}$$

It is tempting to argue that there are no (u_1, u_2) that satisfy this equation w.h.p. over $a \leftarrow \mathbb{Z}_q$. Indeed, suppose, there were a solution (u_1, u_2). Then, viewing this as a degree-2 equation over the variable a, we see that the probability that $p(a) = 0$ is at most $2/q$ by an invocation of Cauchy-Schwartz. However, that would be a mistake since a is not chosen independently of the coefficents of p. Indeed, $u_1 = s$ and $u_2 = s^2$ is a solution to this equation.

Instead, we proceed as follows. Substitute $b = as + e$ in equation 1. We get

$$\begin{aligned} p'(a) &= e(e-1) + (2e-1)(s+u_1) \cdot a + (u_2 + 2su_1 + s^2) \cdot a_2 \\ &= (2e-1)(s+u_1) \cdot a + (u_2 + 2su_1 + s^2) \cdot a_2 = 0 \bmod q \end{aligned}$$

since $e(e-1)$ is 0 by definition. (This, by the way, is easily seen to be a linearization of the polynomial $(e + a \cdot (s+u)) \cdot (e - 1 + a \cdot (s+u))$.)

Now, we can think of this as a polynomial in a with coefficients chosen independent of a, for any fixed u_1, u_2. We argue that there are no solutions with $u_1 \neq -s$. Fix a (u_1, u_2) where $u_1 \neq -s$. Then, $p'(a)$ is a non-zero polynomial in a which is 0 w.p. at most $2/q$ over the choice of a. A Chernoff and union bound now finish off the job for us.

For the full proof, see the paper of Arora and Ge.

When χ is the discrete Gaussian distribution. Let's now see what this does to $\mathsf{LWE}(n, m, q, \chi)$ where χ is a Gaussian with standard deviation s. The probability that the error parameter is less than $k \cdot s$ is $e^{-O(k^2)}$.

- We get a reasonable chance that all equations have error bounded by $k \cdot s$ if $m \cdot e^{-O(k^2)} \ll 1$.
- On the other hand, we need $m > \binom{n}{k \cdot s}$ for linearization to work.

Put together, we get an attack when $m \sim n^{\tilde{O}(s^2)}$. This is non-trivial when $s = \tilde{O}(\sqrt{n})$ which, by some (not so?) strange coincidence, defines the boundary of when the worst-case to average-case reductions (i.e., security proofs) for LWE stop working (as we will see in later lectures).

1.2 Geometric

This is an attack that follows using the LLL algorithm and (building on LLL) the BKZ algorithm that find approximately short vectors in integer lattices.

We will here use facts about integer lattices; we refer the reader to Regev's lecture notes for background on lattices and lattice algorithms.

The attacks use the fact that LWE is, at its core, a problem of finding short vectors in integer lattices. Consider the m-dimensional lattices

$$\mathcal{L} := \{\mathbf{s}^T \mathbf{A} : \mathbf{s} \in \mathbb{Z}_q^n\} \oplus \mathbb{Z}_q^n$$

and

$$\mathcal{L}_{\mathbf{y}} := \{\mathbf{s}^T \mathbf{A} : \mathbf{s} \in \mathbb{Z}_q^n\} \oplus \mathbb{Z}_q^n \oplus \{\mathbf{0}, \mathbf{y}\}$$

where $\oplus$ denotes the Minkowski sum of sets and $(\mathbf{A}, \mathbf{y} = \mathbf{s}^T \mathbf{A} + \mathbf{e}^T)$ is the presumed LWE instance.

Lets look at the case where χ is a B-bounded distribution. We argue that:

- $\mathcal{L}_{\mathbf{y}}$ has a short vector, in fact a vector of ℓ_2 norm $\tilde{O}(B)$ (where $\tilde{O}$ hides $\mathsf{poly}(m)$ factors) since $\mathbf{e} \in \mathcal{L}_y$.
- $\mathcal{L}$ does not have any short vectors. The shortest vector of $\mathcal{L}$ has ℓ_2-norm at least $q^{(m-n)/m} = q \cdot q^{-n/m}$ by a probabilistic argument. This also tells us that the second (linearly independent) shortest vector in $\mathcal{L}_{\mathbf{y}}$ has length $q \cdot q^{-n/m}$.

The LLL/BKZ algorithm finds a vector of length at most $\tilde{O}(2^{m/\log m} \cdot B)$ in polynomial time. As long as this is smaller than $q \cdot q^{-n/m}$, LLL will find $\mathbf{e}$. That is, if $q/B \gg q^{n/m} \cdot 2^{m/\log m}$, LLL/BKZ is bad news for us. Optimizing for m, we get $m \sim \sqrt{n \log q}$ and thus, the attack succeeds if $q/B \gg 2^{\sqrt{n \log q}}$. Setting B to be $\mathsf{poly}(m)$, we get that the attack works if $q \gg 2^n$.

1.3 Combinatorial

This is an attack originally due to Blum, Kalai and Wasserman.

The basic idea is to find small-weight linear combinations $\mathbf{x}_{i,j}$ of the columns of $\mathbf{A}$ that sums up to a fixed vector, say the unit vectors $\mathbf{u}_i$, that is $\mathbf{A}\mathbf{x}_{i,j} = \mathbf{u}_i \bmod q$. Once we find such vectors, we compute

$$\mathbf{b}^T \mathbf{x}_{i,j} = (\mathbf{s}^T \mathbf{A} + \mathbf{e}^T)\mathbf{x}_{i,j} = s_i + \mathbf{e}^T \mathbf{x}_{i,j} \bmod q$$

which, with many copies and averaging, gives us s_i as long as $|\mathbf{e}^T \mathbf{x}_{i,j}| \ll q$. Iterating for all $i \in [n]$ gives us $\mathbf{s}$.

In another variant, we find small-norm $\mathbf{x}_{i,j}$ such that $\mathbf{A}\mathbf{x}_{i,j} = 2^j \mathbf{e}_i \bmod q$. Upon multiplying with $\mathbf{b}$ as before, we get

$$\mathbf{b}^T \mathbf{x}_{i,j} = (\mathbf{s}^T \mathbf{A} + \mathbf{e}^T)\mathbf{x}_{i,j} = s_i 2^j + \mathbf{e}^T \mathbf{x}_{i,j} \bmod q$$

As long as $|\mathbf{e}^T \mathbf{x}_{i,j}| \ll q$, this allows us to "decode" s_i with many fewer copies, essentially $O(\log q)$ of them, using the following decoding algorithm: use $s_i 2^{\lfloor \log(q/2) \rceil} + \mathbf{e}^T \mathbf{x}_{i,j}$ to learn the least significant bit of s_i; this is possible as long as the additive error is sufficiently small; subtract the l.s.b., divide by 2, rinse and repeat.

Back to BKW: The idea of the algorithm is to split the n rows of $\mathbf{A}$ into α groups of size $\beta := n/\alpha$ each.

- For each column $\mathbf{a}_i$ of $\mathbf{A}$ we put it into one of q^β buckets depending on what $\mathbf{a}_i[1 \dots \beta]$ is.
- Notice that the difference of any two vectors, one in bucket $\mathbf{w} \in \mathbb{Z}_q^\beta$ and the other in $\mathbf{w} + \mathbf{v} \in \mathbb{Z}_q^\beta$, starts with $\mathbf{v}$ in the first β positions.
- This gives us many vectors whose first β locations match the target vector. The goal of the rest of the algorithm is to continue along this way while generating vectors whose $\beta \cdot i$ locations match the target, for $i \in [1 \dots \alpha]$.

The result is a linear combination with Hamming weight 2^α of the columns of $\mathbf{A}$ which sum to any given target vector. The process needs q^β vectors to begin with, by a balls-and-bins argument. Assuming the error magnitude is B, we need $2^\alpha \cdot B \ll q$ for correctness. That is,

$$\alpha \ll \log(q/B)$$

This means the sample and time complexity is roughly

$$q^\beta \gg q^{n/\log(q/B)}$$

When, say, $B = n$ and $q = n^2$, this gives us a $2^{O(n)}$-time algorithm (as opposed to the $B^n = n^{O(n)}$ that comes out of enumeration).

Although this presentation of BKW makes it look inferior to lattice reduction, there are versions of BKW that perform concretely better, see e.g., Kirchner-Fouque and Lyubashevsky.