6.876 Advanced Topics in Cryptography: Lattices September 16, 2015

Lecture 4

Lecturer: Vinod Vaikuntanathan *Scribe: Akshay Degwekar*

In this class, we will cover -

- The LLL Algorithm.
- Applications of LLL -
  - Babai's Algorithm for approximating CVP.
  - Exact SVP in $2^{O(n^2)}\text{poly}(W)$ time.
  - Solving "low density" subset sum problems.

1 The LLL Algorithm

We begin describing the algorithm of Lenstra, Lenstra and Lovász for approximating the shortest vector of a lattice.

Theorem 1. *Given a basis $\mathbf{B} \in \mathbb{Z}^{n\times n}$ there is a $\text{poly}(n, W_{\mathbf{B}})$-time algorithm for $\mathsf{SVP}_{2^{O(n)}}$, where $W_{\mathbf{B}}$ is the length of the bit representation of $\mathbf{B}$; namely, the algorithm returns a vector $v \in \mathcal{L}(\mathbf{B})$ such that $\|v\| \leq 2^{O(n)} \cdot \lambda_1(\mathbf{B})$.*

The LLL algorithm actually transforms, in polynomial time, the given basis into a "LLL-reduced" basis for the same lattice. The above theorem holds since a LLL-reduced basis has an important property — its shortest vector is a $2^{O(n)}$-approximation for the shortest vector in entire the lattice. In this lecture we'll give define a LLL-reduced basis and give some intuition for this definition; in the next lecture we'll describe and analyze the LLL algorithm itself.

1.1 LLL-reduced Basis

Our goal is to transform the given basis to one with "short" vectors. Consider the case $n = 2$, i.e., $\mathbf{B} = [\mathbf{b}_1, \mathbf{b}_2]$. Our starting point is the Gram-Schmidt orthogonalization process in which we set $\tilde{\mathbf{b}}_1 := \mathbf{b}_1$ and $\tilde{\mathbf{b}}_2 := \mathbf{b}_2 - \mu_{21}\tilde{\mathbf{b}}_1$, where $\mu_{21} = \langle \mathbf{b}_2, \tilde{\mathbf{b}}_1\rangle / \langle \tilde{\mathbf{b}}_1, \tilde{\mathbf{b}}_1\rangle$. Intuitively, $\tilde{\mathbf{b}}_2$ is the shortest vector we can hope for, since we removed all $\tilde{\mathbf{b}}_1$'s components from it (this fact is what make the Gram-Schmidt orthogonal). However, $\tilde{\mathbf{b}}_2 \notin \mathcal{L}(\mathbf{B})$, and thus cannot be in a basis of $\mathcal{L}(\mathbf{B})$. To fix this issue, we transform $\mathbf{B}$ into the following basis: $\mathbf{b}'_1 = \mathbf{b}$ and $\mathbf{b}'_2 = \mathbf{b}_2 - \lfloor \mu_{21} \rceil \mathbf{b}'_1$, where $\lfloor \cdot \rceil$ means rounding to the closest integer. $\mathbf{b}'_2$ is the shortest *lattice* vector we can hope for, as we removed all the *integer* components of $\tilde{\mathbf{b}}_1$. Note that when projecting $\mathbf{b}'_2$ to the line generated by $\mathbf{b}'_1$, then this projection is between $-\mathbf{b}'_1/2$ to $\mathbf{b}'_1/2$. The latter fact also guarantees that the resulting basis is "close" to orthogonal — the angle between $\mathbf{b}'_1$ to $\mathbf{b}'_2$ is at least 60 degrees. See Figure 1 for an example of this transformation.

So far we reduced $\mathbf{b}_2$, but left $\mathbf{b}_1$ as is. But what if $\mathbf{b}_1$ is very long to begin with? there is no guarantee that the reduced basis is short. At this point we adopt an idea from Euclid's greatest common divisor (gcd) algorithm: we reduce $\mathbf{b}_2$ with respect to $\mathbf{b}_1$ as much as we can, then swap the roles of $\mathbf{b}_1$ and $\mathbf{b}_2$ and repeat the process. The process will stop when the basis meets the following conditions, which are the definition of LLL-reduced basis in two dimensions.

Definition 2 (LLL-reduced basis in two dimensions)**.** *A basis $\mathbf{B} = [\mathbf{b}_1, \mathbf{b}_2]$ is LLL-reduced if*

1. $|\mu_{21}| \leq 1/2$.

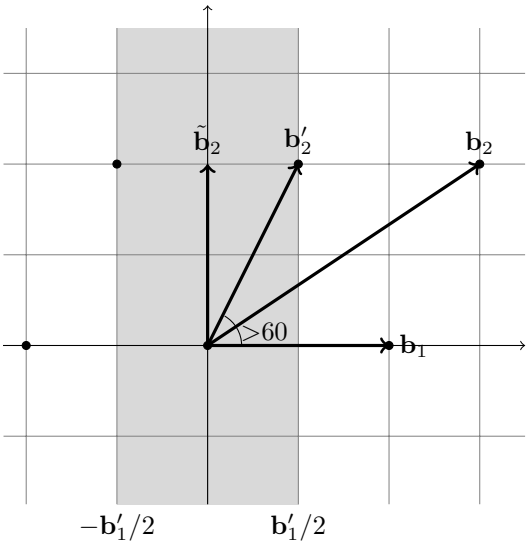

Figure 1: The LLL-reduced basis of $[\mathbf{b}_1 = (2,0), \mathbf{b}_2 = (3,2)]$ is $[\mathbf{b}'_1 = (2,0), \mathbf{b}'_2 = (1,2)]$.

2. $\|\mathbf{b}_2\| \geq \|\mathbf{b}_1\|$.

Note that the second condition can be written as $\left\|\tilde{\mathbf{b}}_2\right\|^2 \geq (1-\mu_{21}^2)\left\|\tilde{\mathbf{b}}_1\right\|^2$. Generalizing for n dimensions we get the following definition.

Definition 3 (LLL-reduced basis)**.** *Let $\delta \in [1/4, 1]$. A basis $\mathbf{B} = [\mathbf{b}_1, \ldots, \mathbf{b}_n]$ is δ-LLL-reduced if*

1. $|\mu_{ij}| \leq 1/2$ *for every* $1 \leq i < j \leq n$.
2. $\left\|\tilde{\mathbf{b}}_{i+1}\right\|^2 \geq (\delta - \mu_{i+i,i}^2)\left\|\tilde{\mathbf{b}}_i\right\|^2$ *for every* $1 \leq i \leq n-1$.

Note that the projection of a (partial) LLL-reduced basis $[\mathbf{b}_1, \ldots, \mathbf{b}_{i-1}, \mathbf{b}_i, \mathbf{b}_{i+1}]$ to $\mathsf{Span}(\tilde{\mathbf{b}}_i, \ldots, \tilde{\mathbf{b}}_n)$ is $[0, \ldots, 0, \tilde{\mathbf{b}}_i, \mathbf{b}_{i+1} + \mu_{i+1,i}\tilde{\mathbf{b}}_i]$. The last two vectors meet the definition of LLL-reduced basis in two dimensions.

2 The LLL Algorithm

2.1 LLL-reduced basis

To recap, we defined a δ-LLL reduced basis last class

Definition 4 (δ-LLL-reduced basis)**.** *Let $\delta \in (1/4, 1)$. A basis $\mathbf{B} = [\mathbf{b}_1, \ldots, \mathbf{b}_n]$ is δ-LLL-reduced if*

1. ***size-reduced.*** $|\mu_{i,j}| \leq 1/2$ *for every* $i > j$.
2. ***Lovász criterion.*** $\left\|\tilde{\mathbf{b}}_{i+1}\right\|^2 \geq (\delta - \mu_{i+i,i}^2)\left\|\tilde{\mathbf{b}}_i\right\|^2$ *for every* $1 \leq i \leq n-1$.

To get some intuition for this definition, we look at the 2d variant we saw last class where we said that $\|\mathbf{b}_2\| \geq \|\mathbf{b}_1\|$ equivalently $\left\|\tilde{\mathbf{b}}_2\right\|^2 \geq (1-\mu_{2,1}^2)\left\|\tilde{\mathbf{b}}_1\right\|^2$. We are slightly changing this by adding a δ as a parameter. Geometrically, the criterion looks at the projection of vectors $\mathbf{b}_1, \mathbf{b}_2 \cdots \mathbf{b}_n$ on to the Gram-Schmidt vectors. The first few vectors $\mathbf{b}_1 \cdots \mathbf{b}_{i-1}$ project to 0, $\mathbf{b}_i$ becomes $\tilde{\mathbf{b}}_i$ and $\mathbf{b}_{i+1}$ projects to

$\tilde{\mathbf{b}}_{i+1} + \mu_{i+1,i}\tilde{\mathbf{b}}_i$. So the Lovász criterion compares the norms of these two projected vectors and says that the second one is 'longer' than the first one, like the 2d case.

It is interesting to note that this condition is extremely local and it can be extended to looking at k vectors at a time to give a 2^k time, $2^{n/k}$ approximation to SVP. We want to argue that finding a LLL-reduced basis is enough to get an approximate shortest vector.

Lemma 5. *If B is a δ-LLL reduced basis, then $\|b_1\| \leq \left(\frac{2}{\sqrt{4\delta-1}}\right)^{n-1} \lambda_1$*

Proof. Since B is δ-LLL reduced, we know that -

$$\left\|\tilde{\mathbf{b}}_{i+1} + \mu_{i+1,i}\tilde{\mathbf{b}}_i\right\|^2 \geq \delta \left\|\tilde{\mathbf{b}}_i\right\|$$

Using the fact that the Gram-Schmidt basis is ortogonal, we rearrange the equation to get

$$\left\|\tilde{\mathbf{b}}_{i+1}\right\|^2 \geq (\delta - \mu_{i+1,i}^2) \left\|\tilde{\mathbf{b}}_i\right\|^2$$

Here we substitute the fact that $|\mu_{i+1,i}| \leq \frac{1}{2}$ to get that

$$\left\|\tilde{\mathbf{b}}_{i+1}\right\|^2 \geq (\frac{4\delta-1}{4}) \left\|\tilde{\mathbf{b}}_i\right\|^2$$

From this we infer that $\left\|\tilde{\mathbf{b}}_i\right\|^2 \geq (\frac{4\delta-1}{4})^{i-1} \left\|\tilde{\mathbf{b}}_1\right\|^2$. This used in conjunction with $\lambda_1 \geq \min_i \left\{\left\|\tilde{\mathbf{b}}_i\right\|\right\}$ gives us the required result. □

So, in any LLL-reduced basis, the first vector would be a good approximation to the shortest vector. It is interesting to note that this gives us a bit more - we can use the fact that the i-th largest element of the set - $\left\{\left\|\tilde{\mathbf{b}}_j\right\|\right\}_j$ gives us a lower bound on λ_i to get a comparable approximation on λ_i for all i.

2.2 Finding an LLL-reduced basis

Lemma 5 reduces the problem of getting a $2^{O(n)}$ approximation to SVP to finding an LLL-reduced basis. In this section we describe the LLL algorithm to find a reduced basis and analyze it.

Input Basis $\mathbf{b}_1, \mathbf{b}_2 \cdots \mathbf{b}_n$
while
1 Compute $\tilde{\mathbf{b}}_1, \tilde{\mathbf{b}}_2 \cdots \tilde{\mathbf{b}}_n$ [a]
2 **for** $i = 2$ to n // Reduction Step
for $j = i - 1$ to 1
$\mathbf{b}_i \leftarrow \mathbf{b}_i - c_{i,j}\mathbf{b}_j$ where $c_{i,j} = \lfloor \mu_{i,j} \rceil$
3 **if** $\exists$ i such that $\left\|\tilde{\mathbf{b}}_{i+1}\right\| < \sqrt{\delta - \mu_{i+1,i}^2} \left\|\tilde{\mathbf{b}}_i\right\|$ [b] // Swap step
Swap $\mathbf{b}_i$ and $\mathbf{b}_{i+1}$
else
Output $\mathbf{b}_1, \mathbf{b}_2 \cdots \mathbf{b}_n$

[a] This step does have some issues with runtime - the exact Gram-Schmidt vectors might keep getting bigger in terms of bit length. We don't consider those issues here.

[b] We do not recompute the $\tilde{\mathbf{b}}_i$'s again because the Reduction step does not affect $\tilde{\mathbf{b}}_i$'s.

Figure 2: The LLL algorithm

The LLL-algorithm is an iterative algorithm where in each iteration, we first replace the vectors $\mathbf{b}_i$'s by 'approximately' orthogonal vectors in the lattice obtained from the vectors $\tilde{\mathbf{b}}_i$'s. After this, we check if the Lovász criterion is violated for any pair of vectors $\mathbf{b}_i, \mathbf{b}_{i+1}$. In case of a violation, we swap the two vectors and continue.

Correctness To prove correctness, we need to show that, if the algorithm terminates, then the output basis is LLL-reduced.

In order to show that $|\mu_{i,j}| \leq \frac{1}{2}$, consider the last iteration. It will not do any swaps and hence the vectors $\tilde{\mathbf{b}}_i$'s will remain unchanged in the iteration. On the other hand, because we subtract $c_{i,j}$'s, the values of $\mu_{i,j}$'s changes. Namely $\mu_{i,j}^{new} = \mu_{i,j}^{old} - \lfloor \mu_{i,j}^{old} \rceil$ and hence $\left|\mu_{i,j}^{new}\right| \leq 1/2$. Note that this relies on the fact that we are decrementing j in step 2 of the algorithm. The Lovász criterion is satisfied by termination – if it was not satisfied for some i, then we would swap $\mathbf{b}_i$ and $\mathbf{b}_{i+1}$ and iterate again.

Termination The termination is a potential argument. We define a non-negative potential function $\phi(\mathbf{B})$ for any basis and then show that it was not too large to begin with and that each iteration reduces this function by a constant.

Let $\phi(\mathbf{B}) = \prod_i \phi_i(\mathbf{B})$ where

$$\phi_i(\mathbf{B}) = |\det(\mathcal{L}(\mathbf{b}_1, \mathbf{b}_2 \cdots \mathbf{b}_i))|$$

Where the determinant for non full rank matrices is defined as $\det(\mathbf{B}) = \det(\sqrt{\mathbf{B}^\top \mathbf{B}})$. Another way to write it would be

$$\phi_i(\mathbf{B}) = \left\|\tilde{\mathbf{b}}_1\right\| \cdot \left\|\tilde{\mathbf{b}}_2\right\| \cdots \left\|\tilde{\mathbf{b}}_i\right\|$$

Observation 6. *$\phi(\mathbf{B})$ is not too large to begin with*

$$\phi(\mathbf{B}_{init}) \leq \prod_{i=1}^{n} \|\mathbf{b}_1\| \cdot \|\mathbf{b}_2\| \cdots \|\mathbf{b}_i\| \leq \max_i(\|\mathbf{b}_i\|)^{O(n^2)}$$

So, $\log(\phi(\mathbf{B}_{init})) = \text{poly}(n, W)$ where W is the bit length of the vectors. We also know that $\phi(\mathbf{B}) \geq 1$ because of the fact that we are dealing with integer lattices and each potential ϕ_i can be interpreted as the i-dimensional volume enclosed by the vectors $\mathbf{b}_1, \mathbf{b}_2 \cdots \mathbf{b}_i$.

Observation 7. *The reduction step does not change the potential function*

This is evident by looking at $\phi_i(\mathbf{B}) = \left\|\tilde{\mathbf{b}}_1\right\| \left\|\tilde{\mathbf{b}}_2\right\| \cdots \left\|\tilde{\mathbf{b}}_i\right\|$ and observing that the reduction step leaves $\tilde{\mathbf{b}}_i$'s invariant.

Observation 8. *The swap step reduces ϕ by a constant factor.*

Proof. Let us say that $\mathbf{b}_i$ and $\mathbf{b}_{i+1}$ were swapped. This only affects the value of $\phi_i(\mathbf{B})$ because changing order of vectors does not affect the determinant.

The old value of ϕ_i is, $\phi_i^{old} = \left\|\tilde{\mathbf{b}}_1\right\| \cdot \left\|\tilde{\mathbf{b}}_2\right\| \cdots \left\|\tilde{\mathbf{b}}_i\right\|$ while the new value is $\phi_i^{new} = \left\|\tilde{\mathbf{b}}_1\right\| \cdot \left\|\tilde{\mathbf{b}}_2\right\| \cdots \left\|\tilde{\mathbf{b}}_{i-1}\right\| \cdot \left\|\tilde{\mathbf{b}}_{i+1}\right\|$. We see that the component of $\tilde{\mathbf{b}}_{i+1}$ orthogonal to the span of $\mathbf{b}_1, \mathbf{b}_2 \cdots \mathbf{b}_{i-1}$ is $\tilde{\mathbf{b}}_{i+1} + \mu_{i+1,i}\tilde{\mathbf{b}}_i$. So,

$$\frac{\phi^{new}}{\phi^{old}} = \frac{\phi_i^{new}}{\phi_i^{old}} = \frac{\left\|\tilde{\mathbf{b}}_{i+1} + \mu_{i+1,i}\tilde{\mathbf{b}}_i\right\|}{\left\|\tilde{\mathbf{b}}_i\right\|} < \sqrt{\delta}$$

So as long as $\delta < 1$ is a fixed constant, we know that the potential function decreases by a constant factor. □

Combining the observations we see that the algorithm has to terminate in time $\text{poly}(n, W)$.

3 Applications

The LLL algorithm has many applications - the most famous one being factoring polynomials over rationals. It also gives us an algorithm for finding the exact shortest vector - while the algorithm takes exponential time, it was the first algorithm to solve exact SVP for a fixed dimension.

3.1 Computing Shortest Vectors in $2^{O(n^2)}$ Time

Corollary 9 (Exact SVP)**.** *There is an exact SVP algorithm running in time* $2^{O(n^2)}$

Proof. To show this we consider an LLL-reduced basis $\mathbf{B}$. Consider a shortest vector $v = \sum_i c_i \mathbf{b}_i$. We want to say that $c_i \leq 2^{O(n)}$ and hence by iterating over c_i's and returning the smallest vector would give us a $2^{O(n^2)}$ algorithm.

Consider $v = \sum_i c_i \mathbf{b}_i = v = \sum_i \tilde{c}_i \tilde{\mathbf{b}}_i$. We look at the last i such that c_i is non-zero. Then $c_i = \tilde{c}_i$ because it is the only coefficient contributing in the $\tilde{\mathbf{b}}_i$ direction. Since $\|\mathbf{b}_1\| \geq \|v\| \geq c_i \left\|\tilde{\mathbf{b}}_i\right\|$, we get that $c_i \leq (\frac{4}{4\delta-1})^{(i-1)/2}$ because $\left\|\tilde{\mathbf{b}}_i\right\|^2 \geq (\frac{4\delta-1}{4})^{i-1} \left\|\tilde{\mathbf{b}}_1\right\|^2$. □

3.2 Babai's Algorithms for Approximate CVP

Corollary 10 (Babai's Algorithm for Approximate CVP)**.** *Let* $\mathbf{B}$ *be a reduced LLL-basis and* $\boldsymbol{y}$ *be the input vector. There are two variants of the algorithm -*

1. *Babai's rounding algorithm - We find the representation of* $\boldsymbol{y}$ *in the real span of* $\mathbf{B}$ *and output the rounded coefficients. Succinctly, output* $\mathbf{B} \left\lfloor \mathbf{B}^{-1} y \right\rceil$
2. *Babai's nearest plane algorithm - The algorithm works iteratively. We consider the hyperplane* $H = \mathsf{Span}(\mathbf{b}_1 \cdots \mathbf{b}_{n-1})$ *and its translations -* $\mathbf{b}_n + H$*,* $2\mathbf{b}_n + H$ *and so on and find the one closest to* $\boldsymbol{y}$*, then project* $\boldsymbol{y}$ *on* H *and recurse.*

References

[1] Divesh Aggarwal and Daniel Dadush and Oded Regev and Noah Stephens-Davidowitz. *Solving the Shortest Vector Problem in* 2^n *Time Using Discrete Gaussian Sampling: Extended Abstract.* STOC 2015.

[2] Lenstra, A.K. and Lenstra, H.W.jun. and László Lovász. *Factoring polynomials with rational coefficients.* Math. Ann. 1982