18.409 An Algorithmist's Toolkit 11/17/2009

Lecture 19

Lecturer: Jonathan Kelner *Scribe: Steven Sam*

1 Review of last lecture

Recall that $L \subset \mathbf{R}^n$ is a *lattice* if L is a discrete set and is closed under addition and negation. In this lecture, we assume that all lattices are *full-rank*, i.e., the vectors in the lattice span $\mathbf{R}^n$.

Given linearly independent $b_1, \ldots, b_n \in \mathbf{R}^n$, one can form a lattice

$$L(B) = L(b_1, \ldots, b_n) = \{Bx \mid x \in \mathbf{Z}^n\},$$

where B is the set of the b_i's. We call the b_i's (and also the set B) a basis for $L(B)$. Recall that a lattice can have many different bases. As opposed to linear algebra over a field, change of bases for lattices is more rigid, since the integrality constraints must be preserved. Because of this, one cannot usually find an orthonormal basis, and instead one of the most fundamental problems becomes finding a nice basis, which consists of short and almost orthogonal vectors.

Recall that for a basis B of L, the *fundamental parallelepiped* is

$$P(B) = \{Bx \mid x \in [0,1)^n\}.$$

Furthermore, if L is full rank, then the *determinant* of L is defined as $\mathrm{Vol}(P(B)) = |\det(B)|$, and this is independent of the choice of a basis. The determinant is inversely proportional to the density of a lattice.

We say that an $n \times n$ matrix M is *unimodular* if all entries of M are integers, and $|\det(M)| = 1$. Last time we saw that a matrix U is unimodular if and only if U^{-1} is unimodular. This implies that an inverse of a unimodular matrix has integer entries. Unimodular matrices are interesting for us, because two lattice bases B_1 and B_2 are equivalent if and only if $B_1 = UB_2$, for some unimodular matrix U. Moreover two bases are equivalent if and only if one can be obtained from the other by the following operations:

- $b_i \leftarrow b_i + k \cdot b_j$, for $i \neq j$ and $k \in \mathbf{Z}$,
- swapping vectors: $b_i \leftrightarrow b_j$,
- $b_i \leftarrow -b_i$.

Last time, we proved the following theorem, which we will need for the proof of Minkowski's theorem.

Theorem 1 (Blichfield) *For any full rank lattice L and measurable set $S \subseteq \mathbf{R}^n$ with $\mathrm{Vol}(S) > \det(L)$, there exist distinct $z_1, z_2 \in S$ such that $z_1 - z_2 \in L$.*

2 Minkowski's theorem

Theorem 2 (Minkowski's Theorem) *If L is a full rank lattice, and S any centrally-symmetric convex set of volume greater than $2^n \cdot \det(L)$, then K contains a nonzero point of L.*

Proof Consider the set $\hat{S} = \frac{1}{2}S$. Then $\mathrm{Vol}(\hat{S}) = 2^{-n}\mathrm{Vol}(S) > \det(L)$ by assumption. So we can apply Blichfield's theorem to conclude that there exist distinct points $z_1, z_2 \in \hat{S}$ such that $z_1 - z_2 \in L$. In particular, $2z_1, 2z_2 \in K$. Since K is centrally-symmetric, we also have $-2z_2 \in K$. Hence the point $z_1 - z_2 = \frac{1}{2}(2z_1 + (-2z_2))$ is in K since it is convex. ■

This theorem is very useful in many settings. For one, many nice number theory theorems follow from it. It also guarantees that the length $\lambda_1(L)$ of the shortest vector in the lattice is not too big.

Corollary 3 *For any full-rank lattice L,*

$$\lambda_1(L) \leq \sqrt{n} \cdot (\det L)^{1/n}.$$

Proof We first bound the volume of the ball $B(0,r)$, for some radius r. This ball contains the hypercube $\left[-\frac{r}{\sqrt{n}}, \frac{r}{\sqrt{n}}\right]^n$. Hence, its volume is greater than $\left(\frac{2r}{\sqrt{n}}\right)^n$.

For $r = \sqrt{n} \cdot \det(L)^{1/n}$, the volume of $B(0,r)$ is greater than $2^n \cdot \det(L)$, so the ball contains a nonzero lattice vector, and therefore, the length of the shortest vector is at most $\sqrt{n} \cdot \det(L)^{1/n}$. ■

The above corollary easily generalizes to other minima. For instance, we will see in a problem set that

$$\left(\prod_{i=1}^{n} \lambda_i(L)\right)^{1/n} \leq \sqrt{n} \cdot (\det L)^{1/n},$$

where $\lambda_i(L)$ is the length of the ith shortest vector.

3 Algorithmic questions

One could ask, for instance, if the bound given above for $\lambda_1(L)$ is tight, and when it holds. Here we will focus on the algorithmic aspect of lattices. There are several interesting questions that one can ask for lattices. We assume that all lattices have integer coordinates. This is the same as giving them rational coordinates, since we can always multiply all coordinates of all vectors by the least common multiple of their denominators.

- **Shortest Vector Problem (SVP)**: Find the shortest vector in L. Finding just the length of the shortest vector is equivalent.
- **Closest Vector Problem (CVP)**: Find the vector in L closest to some given point p.

Both of the above problems are NP-hard, so one usually focuses on the approximate version of them: "Find a vector within γ of the optimum". Some similar questions, like "Does a vector of a given length exist?" turn out to be non-equivalent.

For the approximation versions of SVP and CVP, the gaps between the best known upper and lower bounds are very large. For instance, the best polynomial time algorithms for these problems get approximation factors which are essentially exponential in n. The best known factor is roughly $2^{O(n \log\log n/\log n)}$. The best exact algorithm runs in $2^{O(n)}$ time. It turns out that one cannot find the vector guaranteed by Minkowski's Theorem. SVP is hard to approximate within any constant factor unless NP = RP. CVP is hard to approximate within $n^{O(1/\log\log n)}$. Approximation within the factor $\sqrt{n}$ is in NP $\cap$ co-NP.

4 Lattice basis reduction

We will show a polynomial time algorithm to approximately solve the SVP within a factor of $2^{O(n)}$. Because of an exponential error this might seem to be a very weak and useless result. Nevertheless, this algorithm is good enough to give extremely striking results both in theory and practice. For instance, it can be used to show that an integer program with a constant number of variables can be solved in polynomial time.

4.1 Review of the Gram–Schmidt algorithm

Since our approach resembles the Gram–Schmidt algorithm, we first review this method for orthogonalizing a basis for inner product spaces.

We are given a basis $b_1, \ldots, b_n$ for a vector space, and we want to construct an orthogonal basis $b_1^\star, \ldots, b_n^\star$ such that $\text{span}(b_1, \ldots, b_k) = \text{span}(b_1^\star, \ldots, b_k^\star)$, for all $k \in \{1, \ldots, k\}$. In the Gram–Schmidt algorithm, the vectors $b_i^\star$ are usually normalized, but we will not do it here.

The process works as follows:

- Let $b_1^\star := b_1$.
- For $k = 2$ to n: $b_k^\star := b_k - [\text{projection of } b_k \text{ onto span}(b_1, \ldots, b_{k-1})]$.

The projection is computed in the following way:

$$\begin{aligned}\text{projection of } b_k \text{ onto span}(b_1, \ldots, b_{k-1}) &= \text{projection of } b_k \text{ onto span}(b_1^\star, \ldots, b_{k-1}^\star) \\ &= \sum_{i=1}^{k-1} \text{projection of } b_k \text{ onto } b_i^\star = \sum_{i=1}^{k-1} \frac{b_k \cdot b_i^\star}{\|b_i^\star\|^2} b_i^\star\end{aligned}$$

We set coefficients μ_{ki} so that $\mu_{kk} = 1$, and

$$b_k = \sum_{i=1}^{k} \mu_{ki} b_i^\star.$$

Therefore, we can write the above as $B = MB^\star$, where the basis vectors are rows of B and $B^\star$, and

$$M = \begin{bmatrix} \mu_{11} & 0 & 0 & \cdots & 0 \\ \mu_{21} & \mu_{22} & 0 & \cdots & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ \mu_{n1} & \mu_{n2} & \mu_{n3} & \cdots & \mu_{nn} \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & \cdots & 0 \\ \mu_{21} & 1 & 0 & \cdots & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ \mu_{n1} & \mu_{n2} & \mu_{n3} & \cdots & 1 \end{bmatrix}.$$

Note that $\det(M) = 1$, so for lattices, we have $\text{Vol}(B) = \text{Vol}(B^\star)$, but since entries of M are not necessarily integers, $L(B) = L(B^\star)$ does not have to hold. However, $B^\star$ can be used to bound the length $\lambda_1(L(B))$ of the shortest vector in $L(B)$.

Lemma 4 *For any nonzero $b \in L(B)$, $\|b\| \geq \min_i \|b_i^\star\|$.*

Proof Every nonzero $b \in L(B)$ can be expressed as $b = \sum_{i=1}^{k} \lambda_i b_i$, where $\lambda_k \neq 0$ and for each λ_i is an integer. We have

$$b = \sum_{i=1}^{k} \lambda_i b_i = \sum_{i=1}^{k} \lambda_i \sum_{j=1}^{i} \mu_{ij} b_j^\star = \lambda_k b_k^\star + \sum_{j=1}^{k-1} \sum_{i=1}^{k} \lambda_i \mu_{ij} b_j^\star,$$

and therefore,

$$\|b\|^2 \geq \|\lambda_k b_k^\star\| \geq \|b_k^\star\|^2,$$

which finishes the proof. ■

4.2 Gauss's Algorithm

We start by presenting an algorithm for solving the 2-dimensional SVP exactly.

We call a basis u, v for a 2-dimensional lattice *reduced* if $\|u\| \leq \|v\|$, and $2|u \cdot v| \leq \|u\|^2$. One can show that the following claim holds.

Proposition 5 *A reduced basis for a 2-dimensional lattice contains the first two successive minima of L.*

Sketch of Proof Rotate the plane, so that $u = (u_1, 0)$, and $v = (v_1, v_2)$. We claim that the vector v is a vector with the smallest possible nonnegative second coordinate. The property $2|u \cdot v| \leq \|u\|^2$ implies that $v_1 \in [-\frac{u_1}{2}, \frac{u_1}{2}]$, which in turn implies that v is the shortest vector whose second coordinate is v_2. Because $|v_2| \geq \sqrt{3}/2|u|$, every vector whose second coordinate is greater than $|v_2|$ (that is, at least $2|v_2|$) has length at least $\sqrt{3}|u|$ and cannot be shorter than u. Therefore, u is the shortest vector. Also, since $|v_2| \geq \sqrt{3}/2|v|$, one can show that every vector with second coordinate greater than $|v_2|$ has length at least $\sqrt{3}|v|$. This implies that v is the shortest vector not generated by u. ■

A formal description of Gauss's algorithm follows.

While $\{u, v\}$, where $\|u\| \leq \|v\|$, is not reduced:

- Set $v := v - mu$, where $m \in \mathbf{Z}$ is chosen to minimize the length of $v - mu$.
- If $\|u\| \leq \|v\|$, **break**.
- If $\|v\| \leq \|u\|$, then swap u and v, and repeat.

In the second step, if $\|u\| \leq \|v\|$ even after the reduction, the basis cannot be further reduced and one can prove that $2|u \cdot v| \leq \|u\|^2$.

The algorithm is like a 2-dimensional discrete version of Gram–Schmidt, and is similar to the Euclidean GCD algorithm. Can we make it run in polynomial time? It turns out that it actually does run in polynomial time, but the proof of this fact is not obvious, and therefore, we do not present it here. Instead of this, we replace the termination criterion with

$$\text{If } (1 - \varepsilon)\|u\| \leq \|v\|, \textbf{ break}.$$

It is easy to prove that the modified algorithm gives an $(1 - \varepsilon)$ approximate answer. Now in each reduction, we decrease the length of one of the vectors by at least a constant factor. Therefore the modified algorithm runs in weakly polynomial $O(\log(\|u\| + \|v\|)/\varepsilon)$ time.

The proof that Gauss's algorithm runs in polynomial time uses the fact that for a sufficiently small ε, after the modified algorithm stops, only one more reduction suffices to get a reduced basis.

4.3 Reduced bases

We want to extend the notion of reduced bases to higher dimensions. In order to find a short vector in the lattice, we would like to perform a discrete version of the Gram–Schmidt. So we need to formalize the notion of being orthogonal in lattice problems. One way to do this is to say that the result of our procedure is "almost orthogonalized" so that doing Gram–Schmidt does not change much. In this section, we use the notation from Section 4.1.

Definition 6 (Reduced bases) *Let $\{b_1, \ldots, b_n\}$ be a basis for a lattice L and let M be its Gram–Schmidt matrix defined above. Then $\{b_1, \ldots, b_n\}$ is a reduced basis if it meets the following two conditions:*

1. *All the non-diagonal entries of M satisfy $|\mu_{ik}| \leq 1/2$.*
2. *For each i, $\|\pi_{S_i} b_i\|^2 \leq \frac{4}{3}\|\pi_{S_i} b_{i+1}\|^2$, where S_i is the subspace orthogonal to $\operatorname{span}(b_1, \ldots, b_{i-1})$.*

Remark The constant 4/3 here is somewhat arbitrary. In fact, any number strictly between 1 and 4 will do.

Remark Condition 2 is equivalent to $\|b^\star_{i+1} + \mu_{i+1,i} b^\star_i\|^2 \geq \frac{3}{4}\|b^\star_i\|^2$ and one may think it as requiring that the projections of any two successive basis vectors b_i and b_{i+1} onto S_i satisfy a gapped norm ordering condition, analogous to what we did in Gauss's algorithm for 2-dimensional case.