18.409: An Algorithmist’s Toolkit
Lecture 20

Jonathan Kelner
Today

- Finish lattice basis reduction
- Lenstra’s algorithm
Lattice Basis Reduction

- Takes any basis for a lattice and produces a “good” one
- Gives a poly time algorithm to approximately solve the SVP within a factor of $2^{O(n)}$
- Has lots of applications
Reminder: What Gram-Schmidt Gives Us

- We’re given a basis $b_1, \ldots, b_n$ for a vector space (no lattices here yet),
- Construct an orthogonal basis $b_1^*, \ldots, b_n^*$ such that $\text{span}(b_1, \ldots, b_k) = \text{span}(b_1^*, \ldots, b_k^*)$ for all $k$
- 
  $$b_k = \sum_{i=1}^{k} \mu_{ik} b_i^*$$
- $b_i^*$ are orthogonal and $\mu_{kk}=1$
- Can write the above as $B = MB^*$, where basis vects are rows of $B$ and $B^*$ and

\[
M = \begin{bmatrix}
\mu_{11} & 0 & 0 & \ldots & 0 \\
\mu_{21} & \mu_{22} & 0 & \ldots & 0 \\
\vdots & \vdots & \ddots & \ddots & \vdots \\
\mu_{n1} & \mu_{n2} & \mu_{n3} & \ldots & \mu_{nn}
\end{bmatrix} = \begin{bmatrix}
1 & 0 & 0 & \ldots & 0 \\
\mu_{21} & 1 & 0 & \ldots & 0 \\
\vdots & \vdots & \ddots & \ddots & \vdots \\
\mu_{n1} & \mu_{n2} & \mu_{n3} & \ldots & 1
\end{bmatrix}
\]

- $\det(M)=1$, so $\text{vol}(B)=\text{vol}(B^*)$
- Entries are not integers, so $L(B) \neq L(B^*)$
- **Claim:** For any $b \in L$, $||b|| \geq \min_i \{|b_i^*|\}$
Reminder: Gauss’s Algorithm

- Call a basis \{u,v\} for a 2-d lattice *reduced* if
  - \(|u| \leq |v|\), and
  - \(|u \cdot v| \leq \frac{|u|^2}{2}\)

- Showed it gives first two successive minima
- Gauss’s Algorithm finds a 2-d reduced basis:
  - **While** \{u,v\}, \(|u| \leq |v|\), is not reduced:
    - \(v \leftarrow v - mu\), where \(m \in \mathbb{Z}\) is chosen to give vect of min length
    - If \(|u| \leq |v|\), I claim basis is reduced, so **break**
    - Otherwise, swap(u,v) and repeat

- Not totally obvious that takes poly time (although it actually does)
- **Fix:** change termination criterion
  - **break** if \((1-\varepsilon)|u| \leq |v|\)
- Only gives \((1-\varepsilon)\)-approximate answer, but good enough for us
Can make concrete the statement that we’re trying to do discrete G-S

One way to do this is to say that the result of our procedure is “almost orthogonalized” so that doing Gram-Schmidt to it doesn’t change much

Call a basis meeting two conds below, “reduced”

**Condition 1:** If result of G-S is $B=MB^*$,

$$M = \begin{bmatrix}
1 & 0 & 0 & \ldots & 0 \\
\mu_{21} & 1 & 0 & \ldots & 0 \\
\vdots & \vdots & \ddots & \ddots & \ddots \\
\mu_{n1} & \mu_{n2} & \mu_{n3} & \ldots & 1
\end{bmatrix}$$

then we require all (nondiagonal) $|\mu_{ik}| \leq 1/2$

- Note that our 2-d def. of reduced guaranteed this

This was satisfied by first iteration of algorithm

Need to deal with cases when we’d like to “swap”

Let $S_i = \text{orthog comp of span}(b_1, \ldots, b_{i-1})$

**Condition 2:** $||\pi_{S_i} b_i||^2 \leq (4/3) ||\pi_{S_i} b_{i+1}||^2$

4/3 is somewhat arbitrary—could be anything in (1,4)

- If it were 1, how does this relate to 2-d def of reduced?
The LLL Algorithm

• Repeat the following two steps until you have a reduced basis

**Step 1: Gauss Reduction**

```python
for i = 1 to n
    for k = i-1 to 1
        m ← round(μ_{ik})
        b_i ← b_i − mb_k
    end
end
```

**Step 2: Swapping**

- if exists i s.t. $||\pi_{S_i}b_i||^2 > (4/3)||\pi_{S_i}b_{i+1}||^2$
  - then swap $b_i$ and $b_{i+1}$

• Pretty intuitive algorithm, but not obvious that it converges in poly number of steps or gives good answer
• First need to show that reduced basis gives short vector
  ◦ See blackboard
Need to show two things:

1) For all non-diag, step 1 makes $|\mu_{ij}| \leq 1/2$

2) Total number of iterations of whole {Step 1-Step 2} block is polynomial

See blackboard
Applications of LLL

• LLL has many applications. Here are a few (many taken from Regev’s notes):
  ◦ Solve integer programs in bounded dimension
    • Will do this today
  ◦ Factor polynomials over the integers or rationals
    • Note: harder than over reals. Have to distinguish $x^2-1$ from $x^2+1$, etc.
  ◦ Given an approximation of an algebraic number, find its minimal polynomial.
    • E.g., given 0.645751, output $x^2 + 4x - 3$
  ◦ Find integer relations among a set of numbers
    • E.g. turns out that $\arctan(1) - 4 \arctan(1/5) + \arctan(1/239) = 0$. How would you find this just given these numbers as decimals?
  ◦ Approximate SVP, CVP, etc.
  ◦ Break a whole bunch of cryptosystems
    • RSA for low public exponents
    • Knapsack cryptosystems
  ◦ Real life algorithms for some NP-hard problems, e.g. subset sum
INTEGER PROGRAMMING IN BOUNDED DIMENSION
Linear, Convex, and Integer Programming

- **Linear Programming (feasibility):**
  - **Given:** An m x n matrix $A$ and a vector $b \in \mathbb{R}^n$
  - **Goal:** Find a point $x \in \mathbb{R}^n$ s.t. $Ax \leq b$, or determine that none exists
  - Should really make all the $\mathbb{R}^n$s be $\mathbb{Q}^n$s
  - Other versions, such as optimization are equivalent

- **Convex Programming (feasibility):**
  - **Given:** A separation oracle for a convex body $K$, and a promise that:
    - $K$ is contained in a ball of singly exponential radius $R$
    - If $K$ is nonempty, it contains a ball of radius $r$ that is at least $1/(\text{singly exponential})$
  - **Goal:** Find a point $x \in \mathbb{R}^n$ that belongs to $K$, or determine that none exists

- **Integer Programming:**
  - Same thing as above, but required to produce a point in $\mathbb{Z}^n$, not just $\mathbb{R}^n$

- Linear and convex programming are in P, but integer programming is NP-Hard
What We’ll Show

- **Theorem (Lenstra):** If our polytope/convex body is in \( \mathbb{R}^n \) for any constant \( n \), there exists a polynomial-time algorithm for integer programming.
- For LP, running time will grow exponentially in \( n \), but polynomially in \( m \) and number of bits in the inputs.
- For convex programming, running time is polynomial in \( \log(R/r) \).
- Equivalently, could ask for maximum of \( c \cdot x \) over all \( x \in K \cap \mathbb{Z}^n \)
  - Why is this equivalent?
- See next slide for a picture of why these problems are hard.
How Can We Get Around This?
Change bases so that $K$ is well-rounded, i.e. contains a ball of radius 1 and is contained in a ball of radius $c(n)$ for some $c$ that depends only on $n$

- Needs an algorithmic version of Fritz John’s theorem, although can get away with weaker bounds
- This was on the problem set

This sends $\mathbb{Z}^n$ to some lattice $L$

- Made polytope nice, but might have made lattice really bad
- Might have made the image of our standard basis really bad

Find a reduced basis of $L$

- Now basis can’t be too bad, and polytope is sandwiched between to reasonable-sized balls

Chop space up in some intelligent way and search for lattice points in $K$

**Preprocessing:** Assume that $K$ is full-dimensional and bounded

- How might we get these?

For details of Lenstra’s algorithm, see blackboard

We’ll have some lemmas that we need. These will be listed on the next slide and proved on the blackboard.
Lemmas for Lenstra’s Algorithm

- **Lemma 1**: Let $b_1, \ldots, b_n$ be any basis for $L$ with $||b_1||^2 \leq \ldots \leq ||b_n||^2$. Then

  $$\forall x \in \mathbb{R}^n, \exists y \in L : ||x - y||^2 \leq \frac{1}{4} \left(||b_1||^2 + \cdots + ||b_n||^2\right) \leq \frac{1}{4} n||b_n||^2$$

- **Lemma 2**: For a reduced basis $b_1, \ldots, b_n$ ordered as above:

  - $\prod_{i=1}^{n} ||b_i|| \leq 2^{n(n-1)/4} \det(L)$

  - Let $H = \text{span}(b_1, \ldots, b_{n-1})$. Then

    $$2^{-n(n-1)/4} ||b_n|| \leq \text{dist}(H, b_n) \leq ||b_n||$$