18.409: An Algorithmist’s Toolkit

Lecture 19

Jonathan Kelner
Administrivia

- Pset hints?
Today

- Quick review of last time
- Finish Minkowski’s Theorem
- Start lattice basis reduction
Some Reminders from Last Time

- **Definition:** $L \subseteq \mathbb{R}^m$ is a lattice if:
  - $x, y \in L \implies x + y \in L$ and $-x \in L$
  - $L$ is a discrete set

- For today, all lattices will be *full-rank*, i.e., the vectors in the lattice span $\mathbb{R}^n$.

- Given linearly independent $b_1, \ldots, b_n$, can form lattice
  
  $$L(B) = L(b_1, \ldots, b_n) = \{Bx | x \in \mathbb{Z}^n\}$$

- Call the $b_i$ (and/or $B$) a basis for $L(B)$

- Lattices can have many different bases

- Change of basis is more rigid than in standard linear algebra, since we need to preserve integrality constraints

- So can’t just “find an orthonormal basis” like we do in the continuous setting

- A (the?) fundamental algorithmic question with lattices is to find a “nice” basis in which the basis elements are as small and as close to orthogonal as possible
Defini&on: For a basis B of L, the fundamental parallelepiped is

\[ P(B) = \{ Bx | x \in [0, 1)^n \} \]

Defini&on: An n x n matrix M is unimodular if all entries are integers and \( \det(M) = \pm 1 \)

Facts from last time:
- U is unimodular iff \( U^{-1} \) is unimodular
- Inverse of unimodular matrix has integer entries
- Two bases \( B_1 \) and \( B_2 \) are equivalent iff \( B_2 = UB_1 \) for some unimodular \( U \)

Corollary: Two bases are equiv. iff can get one from other by following ops:
- \( b_i \leftarrow b_i + k b_j \), for some \( k \in \mathbb{Z} \)
- \( b_i \leftrightarrow b_j \) (swap)
- \( b_i \leftarrow -b_i \)

Defini&on-Theorem: Let \( L=L(B) \) be full-rank. The determinant of \( L = \text{vol}(P(B)) = |\det(B)| \), and this is indep of choice of basis.
- Det is inversely proportional to density of lattice
(a) A basis of $\mathbb{Z}^2$

(b) Another basis of $\mathbb{Z}^2$
Blichfield’s Theorem

- Last time, we proved:
- **Theorem [Blichfield]**: Let:
  - $L$ be full-rank
  - $S \subseteq \mathbb{R}^n$ be any measurable set with $\text{vol}(S) > \text{det}(L)$.
  
  Then there exist (unequal) $z_1, z_2 \in S$ s.t. $z_1 - z_2 \in L$
Minkowski’s Theorem(s)

**Theorem:** Let $L$ be full rank, and let $K$ be any centrally-symmetric convex body s.t. $\text{vol}(K) > 2^n \det(L)$. Then $K$ contains a nonzero point of $L$.

- Why do we need the $2^n$?
- See blackboard and this picture

**Corollary:** For any full-rank $L$,

$$\lambda_1(L) \leq \sqrt[n]{\det L}^{1/n}$$

- How tight is this? And when?
- Easy generalization to other successive minima
  - Says can replace LHS with geometric mean (exercise):

$$\left( \prod_{i=1}^{n} \lambda_i(L) \right)^{1/n} \leq \sqrt[n]{\det L}^{1/n}$$
Algorithmic Questions

- A bunch of algorithmic questions we can now ask
- All lattices will have integer coords for these
  - Same as giving them rational coords. (Why?)
- **Shortest Vector Problem (SVP):** Find the shortest vector in L
  - Could also just ask for length, but this is equiv.
- **Closest Vector Problem (CVP):** Find the vector in L closest to some given point p
- Both are NP-hard, so ask approx. version
  - “Find a vector within $\gamma$ of optimum”
- Slight subtlety: could ask a few non-equivalent questions
  - E.g., a different question is just whether a vector of some length exists
• Very large gaps between best known upper and lower bounds.

• Best poly-time algorithms for approximate SVP and CVP get approx. factors essentially exponential in n
  ◦ I think the best is something like $2^{O(n \log \log n / \log n)}$

• Best exact algorithm runs in $2^{O(n)}$

• **Can’t find vector guaranteed by Minkowski!**

• SVP hard to approx. within any constant factor unless NP = RP

• CVP hard to approx. within $n^{O(1/\log \log n)}$

• Approx. within $n^{1/2}$ factor in NP ∩ co-NP
Lattice Basis Reduction

- We’ll show a poly time algorithm to approximately solve the SVP within a factor of $2^{O(n)}$
- Natural instinct may be to dismiss this as an exponential error and thus silly/obvious/useless
- Don’t do that! This algorithm is good enough to get some extremely striking results in both theory and practice.
- We’ll see some applications next time and on the problem set
Reminder of Gram-Schmidt

- We’ve already used Gram-Schmidt a few times, so this is mainly just to fix notation.
- We’re given a basis $b_1, \ldots, b_n$ for a vector space (no lattices here yet).
- We’ll 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 $k$.
- **Slightly different than before:** won’t normalize s.t. $b_i^*$ have norm 1.
**Reminder of Gram-Schmidt**

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

\[
\begin{align*}
&= \sum_{i=1}^{k-1} \text{projection of } b_k \text{ onto } b_i^* \\
&= \sum_{i=1}^{k-1} \frac{b_k \cdot b_i^*}{\|b_i^*\|^2} b_i^*
\end{align*}
\]

- Call coeffs $\mu_{ik}$ and set $\mu_{kk} = 1$, so that

\[
b_k = \sum_{i=1}^{k} \mu_{ik} b_i^*
\]
What Gram-Schmidt Gives Us

\[ b_k = \sum_{i=1}^{k} \mu_{ik} b_i^* \]

- \( b_i^* \) are orthogonal and \( \mu_{kk} = 1 \)
- Can write the above as \( B = MB^* \), where basis vects are rows of \( B \) and \( B^* \) and

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

- \( \det(M) = 1 \) [why?], so \( \text{vol}(B) = \text{vol}(B^*) \)
- Entries are not integers, so \( L(B) \neq L(B^*) \)
- **Claim:** For any \( b \in L \), \( ||b|| \geq \min_i \{ ||b_i^*|| \} \)
  - See blackboard
- So can use this to bound \( \lambda_1(L(B)) \)
Warmup: Gauss’s Algorithm

- Let’s solve the 2-d SVP exactly
- Call a basis \{u,v\} for a 2-d lattice reduced if
  - \(|u| \leq |v|\), and
    - \(|u \cdot v| \leq \frac{|u|^2}{2}\)

**Claim:** A reduced basis for a 2-d lattice comprises the first two successive minima of \(L\)

**Proof:** See blackboard
Gauss’s Algorithm

- **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
- Why can we stop if \( |u| \leq |v| \) in 2\(^{nd}\) step?
- Why does this terminate? (Don’t worry about poly time yet.)
- Kind of like a 2-d discrete version of Gram-Schmidt
- Should remind you of Euclidean GCD algorithm
- How do we make it run in poly time?
- Actually does, but proof is not totally obvious
- **Easier:** change termination criterion
  - **break** if \((1-\varepsilon)|u| \leq |v|\)
- Only gives \((1-\varepsilon)\)-approximate answer, but good enough for us
- Why does this help?
Reduced Bases

- 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 $B=MB^*$,

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

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

```latex
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**

```latex
if exists i s.t. \|\pi_s b_i\|^2 > (4/3)\|\pi_s 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
Convergence of LLL

- 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