Discrete Math, Eleventh Problem Set (July 14)

REU 2003

Instructor: Laszlo Babai
Scribe: D. Jeremy Copeland

1 Lovász Lattice Reduction - analysis

Recall that a basis $(\mathbf{b}_1, \dots, \mathbf{b}_n)$ for a lattice is **Lovász-reduced** if it satisfies the relations:

1. $|\mu_{ij}| \leq \frac{1}{2}$,
2. $\|\mathbf{b}^*_{i+1}\| \geq \frac{1}{\sqrt{2}}\|\mathbf{b}^*_i\|$,

where $\mathbf{b}^*_1, \dots, \mathbf{b}^*_n$ is the Gram-Schmidt orthogonalized sequence and $\mathbf{b}_i = \mathbf{b}^*_i = \sum_{j<i} \mu_{ij}\mathbf{b}^*_j$. In order to prove that Lovász' lattice reduction algorithm terminates, we introduce the **potential function** $\mathcal{P}$ of a basis defined as:

$$\mathcal{P} = \mathrm{vol}(\mathbf{b}_1)\mathrm{vol}(\mathbf{b}_1\mathbf{b}_2)\cdots\mathrm{vol}(\mathbf{b}_1\cdots\mathbf{b}_{n-1}) = \|\mathbf{b}^*_1\|^{n-1}\cdots\|\mathbf{b}^*_{n-1}\|^1.$$

We would like to show that the Lovász algorithm terminates by observing that in each round, the potential function decreases by a factor bounded above by some constant $c < 1$. Because of this, and the fact that $\mathcal{P}^2$ is a positive integer, we know that the algorithm must terminate in a number of rounds logarithmic in the initial potential.

Exercise 1.1. Prove that the logarithm of the initial potential is bounded by a polynomial of the bit-length of the input.

Recall that the "coefficient reduction" process reduces each μ_{ij} to $|\mu_{ij}| \leq 1/2$ without affecting the orthogonalized sequence and therefore without affecting the potential function. Once "coefficient reduction" has been performed, we check the second property. If there is some i with $\|\mathbf{b}^*_{i+1}\| < \frac{1}{\sqrt{2}}\|\mathbf{b}^*_i\|$, we simply swap these two elements. This will change the orthogonalized vectors $\mathbf{b}^*_i$ and $\mathbf{b}^*_{i+1}$, but the potential is only affected by:

$$\frac{\mathcal{P}_{\text{new}}}{\mathcal{P}_{\text{old}}} = \frac{\|(\mathbf{b}^*_i)_{\text{new}}\|}{\|(\mathbf{b}^*_i)_{\text{old}}\|}$$

Next we project along the direction of $\mathcal{U}_{i-1} = \operatorname{Span}\{\mathbf{b}_1 \cdots \mathbf{b}_{i-1}\}$. After this projection, the vectors $\mathbf{b}_1 \cdots \mathbf{b}_{i-1}$ are taken to the zero vector. Denote the projection by $\mathbf{x} \mapsto \mathbf{x}'$. Now examining $V = \operatorname{Span}\{\mathbf{b}'_i, \mathbf{b}'_{i+1}\} = \operatorname{Span}\{\mathbf{b}^*_i, \mathbf{b}^*_{i+1}\}$, we find that

$$\begin{aligned}
(\mathbf{b}'_i)_{\text{old}} &= (\mathbf{b}^*_i)_{\text{old}} \\
(\mathbf{b}'_{i+1})_{\text{old}} &= (\mathbf{b}^*_{i+1})_{\text{old}} + \mu_{i+1,i}(\mathbf{b}^*_i)_{\text{old}} \\
(\mathbf{b}'_i)_{\text{new}} &= (\mathbf{b}'_{i+1})_{\text{old}} = (\mathbf{b}^*_i)_{\text{new}} \\
(\mathbf{b}'_{i+1})_{\text{new}} &= (\mathbf{b}'_i)_{\text{old}} = (\mathbf{b}^*_i)_{\text{old}}.
\end{aligned}$$

Therefore,

$$\frac{\mathcal{P}_{\text{new}}}{\mathcal{P}_{\text{old}}} = \frac{\|(\mathbf{b}'_i)_{\text{new}}\|}{\|(\mathbf{b}'_i)_{\text{old}}\|}$$

Now, since $\mu_{ij} \leq 1/2$, the projection of $\mathbf{b}'_{i+1}$ onto $\mathbf{b}^*_i$ is prevented from being very large. The calculation follows.

$$\begin{aligned}
\|(\mathbf{b}'_i)_{\text{new}}\|^2 &= \|(\mathbf{b}'_{i+1})_{\text{old}}\|^2 \\
&= \|(\mathbf{b}^*_{i+1})_{\text{old}}\|^2 + \mu^2_{i+1,i}\|(\mathbf{b}^*_i)_{\text{old}}\|^2 \\
&\leq \left(\frac{1}{2} + \frac{1}{4}\right)\|(\mathbf{b}^*_i)_{\text{old}}\|^2 \\
&\leq \frac{3}{4}\|(\mathbf{b}'_i)_{\text{old}}\|^2,
\end{aligned}$$

so $\mathcal{P}_{\text{new}}/\mathcal{P}_{\text{old}} \leq \sqrt{3}/2$.

We see now that we could replace the second Lovász condition by:

(2_δ) $\|\mathbf{b}^*_{i+1}\| \geq \delta\|\mathbf{b}^*_i\|$,

where δ is allowed to be any number such that $\sqrt{\delta^2 + 1/4} < 1$, so $\delta < \sqrt{3}/2$. The running time and quality of the output would be expected to be greater for larger values of δ.

Remark 1.2. These are the values of δ for which we know the algorithm terminates (and specifically in polynomial time). However, it is possible for the algorithm to terminate for special inputs and larger values of δ. (For example, an orthogonal basis and any δ would produce an algorithm that terminates.)

Remark 1.3. Odlyzko and TeRiele used this algorithm to approximate 70 zeroes of the Riemann Zeta function. In their experiments, the lattice reduction algorithm terminated rather quickly even in the case $\delta = \sqrt{3}/2$, even though in this case we cannot prove termination, let alone termination in polynomial time. Moreover, this large value of δ produced outputs of far higher quality than predicted by the analysis. This is a case in which the search for a polynomial-time algorithm required an insight into the nature of the problem, and that insight lead to a solution that turned out to be "unreasonably effective" in practice.

2 Linear programming

The objective of **linear programming** is to maximize a linear objective function $\mathbf{c}^T\mathbf{x} = c_1x_1 + \cdots + c_nx_n$ under a system of constraints of the form $A\mathbf{x} \leq \mathbf{b}$, where A is some fixed $k \times n$ matrix, and $\mathbf{b}$ is a fixed $k \times 1$ vector and the meaning of "$\leq$" is the simultaneous system of inequalities in $\mathbf{x}$:

$$\begin{array}{rcl} a_{11}x_1 + \cdots + a_{1n}x_n & \leq & b_1 \\ & \vdots & \\ a_{k1}x_1 + \cdots + a_{kn}x_n & \leq & b_k \end{array}$$

The objective is then to maximize, within this domain, the value of the objective function.

For example, $\mathbf{c}^T\mathbf{x}$ could be a profit function, and the constraints of A, $\mathbf{b}$ would be production constraints.

Throughout we will use words such as "up" to denote the direction of the vector $\mathbf{c}$. In this terminology, we need to reach the apex (top vertex) of the polytope defined by the constraints (if it has vertices).

The **simplex algorithm** (Dantzig, 1950) solves this problem by hopping from vertex to neighboring vertex, always ascending in the polytope. The algorithm terminates when the apex is reached. This algorithm is provably exponential in the worst case, yet in practice is extremely efficient.

This situation presented one of the foremost challenges to the notion of "polynomial time algorithms."

The **ellipsoid method** (Khachiyan, 1979) was the first polynomial-time algorithm for the linear programming problem. Basically the algorithm envelopes the polytope in an ellipsoid, and shrinks the volume of the ellipsoid by a constant factor, at each step making certain to still retain the topmost vertex (though the topmost vertex is unknown). While this algorithm is polynomial time, it is not a practical method for linear programming.

Remark 2.1. The ellipsoid method also suffers from the fact that it may become unstable if the polytope is lower dimensional, so in this case, it is important to first reduce to a hyperplane on which the problem is "of maximal rank" then solve there. The ellipsoid method itself pinpoints the approximate direction of a vector perpendicular to such a hyperplane; then a simultaneous diophantine approximation of the coordinates of this approximate normal vector results in the exact direction. This "rounding" was Lovász' original objective in developing the basis reduction algorithm.

Remark 2.2. An advantage that the ellipsoid method has is that, in cases where the constraints are implicitly defined, one only needs a **separation oracle**, or algorithm for determining some constraint that a point outside the polytope fails.

A third method to approach this problem is the **interior point method** of Karmarkar, which starts from the "center" of the polytope; chooses a ball in the interior of the polytope around the starting point, and moving up inside the ball. Then a projective transformation is used to make it appear that the new point is in the center, and we repeat the previous step. (Note: any interior point in a ball can be transformed into the center of the ball without changing the shape of the ball). The result is that the point will approach the apex from inside along a curved trajectory which is a straight line in the hyperbolic space into which the interior of the polytope can be converted.

This, again, is a polynomial time algorithm. In contrast with the ellipsoid method, it is also practical; for very large numbers of variables, the interior point method beats the simplex algorithm on benchmark inputs.

3 Primality testing

The objective of primality testing is to take as input a positive integer, and return as output a verdict whether the number is prime or composite. Only as recently as 2002 was it shown that this age-old problem can be solved in polynomial time; the algorithm was discovered by Manindra Agarwal, Neeraj Kayal, and Nitin Saxena (professor and two graduate students) at the Indian Institute of Technology, Kanpur.

Euclid used the sieve of Eratosthenes to perform primality testing. In this method, one constructs a list of the integers from 2 to N. From this list, we remove all of the multiples of 2 on the first step. Since 3 is the smalles remaining number, it must be prime. We next remove all multiples of 3. Now 5 is the smallest number remaining, so it is prime. Remove all its multiples.

Remark 3.1. In order to discover all primes less than N, it is necessary to continue this process for all primes less than $\sqrt{N}$ (why?). Therefore, if X is a number with n digits, $X \approx 10^n$, then it is necessary to sieve until $10^{n/2}$ - an exponentially long process.

If we know the primes up to $10^{n/2}$, then we need only $\pi(10^{n/2}) = \#\{\text{primes } p < 10^{N/2}\}$ trial divisions. How much of a savings is this? To estimate this, we need the celebrated

Theorem 3.2 (Prime Number Theorem). *$\pi(x) \sim x/\ln x$.*

Remark 3.3. The error in this approximation is $\pi(x) = x/\ln x + O(\frac{x}{(\ln x)^2})$.

Remark 3.4. A better approximation is $\pi(x) = \text{li}(x) + O(x^{1-\epsilon})$ for some constant $\epsilon > 0$, where

$$\text{li}(x) = \int_{t=2}^{x} \frac{\mathrm{d}t}{\ln t}.$$

This expression as a good approximation of $\pi(x)$ was conjectured by Gauss. li is called the **logarithmic integral**.

Remark 3.5. It is believed - and in fact is equivalent to the Riemann Hypothesis - that the error in the logarithmic integral expression is of the order of $\sqrt{x}$: $\pi(x) = \mathrm{li}(x) + O(\sqrt{x})$.

Remark 3.6. With regard to the sieve of Eratosthenes, notice that

$$\pi(10^{n/2}) \sim \frac{10^{n/2}}{\frac{n}{2}\ln 10},$$

which for large enough n, is larger than $9.9^{n/2}$. Thus we still have exponential complexity.

We shall present two randomized ("Monte Carlo") algorithms. These algorithms have two possible outputs:

1. "Composite" with a proof of compositeness.
2. "Prime."

In case "Composite" is returned, we know that the number is composite, because a proof of compositeness is produced. However, in the case the output is "Prime," this is only a guess. It is possible, although unlikely, that on composite input, the algorithm returns "prime." The probability of this happening is less than a predetermined $\epsilon > 0$, and it is not unreasonable to ask ϵ to be less than $1/2^{1500}$, which error is less than 1/(number of particles in the known universe).

Remark 3.7. It is often said that a number X deemed to be prime through this process is "very likely prime." This statement, however, does not make sense. X is not chosen at random, it is given explicitly. Therefore either X is prime, or it is not, it cannot have a probability of being prime. Our verdict, however, is probabilistic; with certain probability we say "X is prime" and otherwise we say "X is composite." If X is prime, we shall always say so, so in this case the chance that we make an error is zero. If X is composite, then we may accidentally declare that X is prime, but the probability of this happening is $< \epsilon$. So overall, the probability that our verdict is in error is $< \epsilon$.

Our first tool is Fermat's little Theorem.

Theorem 3.8 (Fermat's little theorem). *If p is a prime, and $gcd(p, a) = 1$, then $a^{p-1} \equiv 1 \mod p$.*

Definition 3.9. A **Fermat witness** of compositeness of x is a number a such that $1 \leq a \leq x - 1$ such that

1. $\gcd(a, x) \neq 1$, or
2. $\gcd(a, x) = 1$ and $a^{x-1} \not\equiv 1 \mod x$.

Remark 3.10. Notice that in the second case, we do not find any factor of x though we do have a proof that it is composite using the contrapositve of Fermat's little theorem.

Unfortunately, this won't quite work as a basis for a primality test, because of the following:

Definition 3.11. x is a **Carmichael number** if x is odd, not prime, but for all a with $\gcd(a, x) = 1$, $a^{x-1} \equiv 1 \mod x$.

Carmichael numbers are rare, but it has been proved recently that there are infinitely many of them (Alford, Granville, Pomerance, 1994).

Exercise 3.12. Show that if $x = pq$ is the product of two primes, then x is not a Carmichael number.

Exercise 3.13. Prove that $x = pqr$ is a Carmichael number if and only if $(p - 1) \mid qr - 1$, $(q - 1) \mid rp - 1$, and $(r - 1) \mid pq - 1$.

Exercise 3.14. Show that the following numbers are Carmichael: 561 ($3 \cdot 11 \cdot 17$), 1105 ($5 \cdot 13 \cdot 17$), 1729 ($7 \cdot 13 \cdot 19$), 2465 ($5 \cdot 17 \cdot 19$), 2821 ($7 \cdot 13 \cdot 31$), 6601 ($7 \cdot 23 \cdot 41$), 8911 ($7 \cdot 19 \cdot 67$), 10585 ($5 \cdot 29 \cdot 73$). (Note: these are the eight smallest Carmichael numbers.)

Exercise 3.15. Prove: if $n = 3pq$ is a Carmichael number where p, q are primes then $n = 561$.

Exercise 3.16. Prove: if $n = (6k+1)(12k+1)(18k+1)$ and each of the three factors is prime then n is a Carmichael number (Korselt, 1899). (Note that for $k = 1$ we obtain 1729; the next case is $k = 6$.)

Exercise 3.17. Prove that all Carmichael numbers are square free. (A number is *square free* if it is not divisible by the square of any prime.) *Hint.* Use the fact that the multiplicative group of mod p^2 residue classes relatively prime to p is cyclic, i. e., that there exists a number a, relatively prime to p, such that $a^j \not\equiv 1 \pmod{p^2}$ for any j, $1 \le j < p^2 - p$.

The next exercise shows how to find a witness of compositeness if the input number is not Carmichael.

Exercise 3.18. Prove that if x is composite and not Carmichael, then the probability that a random a is a Fermat witness is at least $1/2$.