6.854 Advanced Algorithms

Lecture 20: November 23, 1999 Lecturer: David Karger

Scribes: Dave Manz, Andrew Glenn, and Wes Chao

20.1 Ellipsoid

20.1.1 Defining An Ellipsoid

An ellipsoid is a n-dimensional generalization of an ellipse. It consists of a *center* z and a *positive definite matrix* $D = BB^T$, which contains the radius – or scaling fator – in each direction. The ellipsoid is defined by the point set $(x-z)^T D^{-1}(x-z) \leq 1$, where z is the center of the ellipsoid. (We note this is just a basis change of the unit sphere $x^2 \leq 1$.)

20.1.2 Outline of Ellipsoid Algorithm

Our goal is to find a feasible point for $P = Ax \leq b$. We start with an ellipse containing P with center z. If $z \in P$, we are done. Otherwise, we find a hyperplane that separates z from P, and find a smaller ellipse which contains the other half of the previous ellipse. We continue this process until we have an ellipse whose center is in P.

Consider the case of a sphere with separating hyperplane $x_1 = 0$. Let the center be $(a, 0, 0, \ldots)$. We require that:

- $d_1^{-1}(x_1 - a)^2 + \sum_{i>1} d_i^{-1} x_i^2 \leq 1$
- constraint at $(1,0,0)$: $d_1^{-1}(x-a)^2 = 1$ so $d_1 = (1-a)^2$
- constraint at $(0,1,0)$: $a^2/(1-a)^2 + d_2^{-1} = 1$ so $d_2^{-1} = 1 - a^2/(1-a)^2 \approx 1 - a^2$

The volume is about $(1-a)/(1-a^2)^{n/2}$. If we set a about $1/n$, we get $(1-1/n)$ as the ratio of the volumes. Thus, in O(n) steps we can halve the size of the ellipse.

Shrinking Lemma:

Let $E = (z, D)$ define an n-dimensional ellipsoid. Consider the separating hyperplane $ax \leq az$.

Now define the ellipsoid $E' = (z', D')$:

$$\begin{aligned} z' &= z - \frac{1}{n+1}\frac{Da^T}{\sqrt{aDa^T}} \\ D' &= \frac{n^2}{n^2-1}\left(D - \frac{2}{n+1}\frac{Da^TaD}{aDa^T}\right) \end{aligned}$$

then

$$\begin{aligned} E \cap x \mid ax \leq ez &\subseteq E' \\ vol(E') &\leq e^{1/(2n+1)} vol(E) \end{aligned}$$

For the proof, first show that it works with $D = I$ and $z = 0$. Then, for the new ellipse:

$$\begin{aligned} z' &= -\frac{1}{n+1} \\ D' &= \frac{n^2}{n^2-1}(I - \frac{2}{n+1} I_{11}) \end{aligned}$$

and the volume ratio is easy to compute directly.

For the general case, we can transform it to coordinates where $D = I$ (using new basis B), get the new ellipse, transform it back to the old coordinates, and get (z', D'). (Note that the transformation doesn't affect volume *ratios*, so the ellipsoid shrinks.) Now we need to prove two things:

- that we need not start infinitely large.
- that we can't get infinitely small.

Starting size: We may worry about how to bound polyhedra. We recall that the size of vertices is polynomially bounded, so the coordinates of vertices are exponential but no larger. Thus, we can start with a sphere with a radius exceeding this exponential bound and this should be sufficiently large. This only uses polynomial values in the D matrix.

Ending size: It is convenient to assume that the polytope is full dimensional. If it is, it has $n + 1$ affinely independent vertices, all of which have polynomial size coordinates. So, they contain a box whose volume is a polynomial size number (computable as determinant of vertex coordinates).

Putting this all together, we have:

- the starting volume is $2^{n^{O(1)}}$
- the ending volume is $2^{-n^{O(1)}}$
- each iteration reduces volume by a factor of $e^{1/(2n+1)}$

So $2n + 1$ iterations reduce by the volume by e, so $n^{O(1)}$ reduce by $e^{n^{O(1)}}$ at which point the ellipse doesn't contain P. This is a contradiction. We must have hit a point in P before.

Justifying full dimensionality, which we assumed before: Replace $Ax \leq b$ with $P' = Ax \leq b + \epsilon$ for some tiny ϵ. Any point of P is an interior of P', so P' is full dimensional. P is empty iff P' is because ϵ is too small - the proof of this comes by looking at the linear program which tries to minimize ϵ, and arguing that there is a solution which has an ϵ using a polynomial number of bits, and we can take an ϵ which is smaller than that. We can thus "round" a point of P' to P.

20.2 Separation vs Optimization

Notice in the ellipsoid algorithm, we were only using one constraint at a time. Thus, it doesn't matter how many constraints there are, and we don't need to see all of them at once. All that is necessary is that each one can be represented in polynomial size. Thus, the algorithm works even if you have a huge number of constraints, so long as you have *a separation oracle* which, given point not in P, finds a separating hyperplane. Of course, we can convert feasibility to optimality, so we can optimize with this separation oracle too. This works on a polytope by polytope basis. If can separate a particular polytope, we can optimize over that polytope.

This is very useful in many applications. Consider a network design problem, where you are trying to build a network with some redundancy such that for any pair of nodes there exists at least three separate paths from one to the other. This problem is NP-complete, but if we solve it fractionally it can be formulated as a linear program where we stipulate that "across every cut, the number of edges is greater than or equal to 3." This specifices 2^n constraints on m edges, one per cut. Such a problem can be solved by an ellipsoid algorithm where the separation oracle is a minimum cut algorithm and the equation for the separating hyperplane is $\sum$(edges along that cut)$\geq$ 3.

20.3 Interior Point

The ellipsoid algorithm has problems in practice ($O(n^6)$, for one). Thus, people have developed a different approach that has been extremely successful, known as interior point methods.

What goes wrong with the simplex algorithm? It follows the edges of a polytope, where complex stuctures are, so it runs into walls, etc. Interior point algorithms stay away from the walls, where the structure is simpler. Karmarkar developed the first such algorithm in 1984. The one we will be discussing was published by Ye in 1991.

20.3.1 The Potential Function

Potential function:

- Idea: use a (nonlinear) potential function that is large when the current point is not optimal, then minimize the potential function via gradient descent.

In Ye's algorithm, we use the standard primal $Ax = b, x \geq 0$ and its dual $yA + s = c, s \geq 0$. The duality gap is sx. By complementary slackness, at the optimal point xs=0 (so if xs is large, we are in the wrong place). We will use a *logarithmic barrier function:*

$$G(x, s) = q \ln xs - \sum \ln x_j - \sum \ln s_j$$

and try to minimize it. In the above formula for $G(x, s)$, the first term tends to move towards the optimum, while $\sum \ln x_j$ and $\sum \ln s_j$ tend to keep $x, s \geq 0$. These barriers prevent us from ever hitting the optimum, but as discussed above, just getting close will be enough because there exist algorithms to round the point to a vertex point which is no worse.

We choose q so that the first term dominates, guaranteeing a good G is a good xs. The idea is that a small $G(x,s)$ should mean xs is small, and if xs is large then $G(x,s)$ is large as well. To do this, we let:

- $G = \ln (xs)^q / \prod x_j s_j$
- $xs > x_j s_j$, so $(xs)^n > \prod x_j s_j$. So taking $q > n$ makes the top term dominate, and $G > \ln xs$.

Since the first constraint wants a big q, and the second wants a small q, we compromise at $n + \sqrt{n}$, giving us $O(L\sqrt{n})$ iterations.

20.3.2 Reducing the Potential Function

To minimize the potential function, we will use gradient descent. To do this, we take the current point (x,s), and take a linear approximation to the potential function around this point. By moving to where the linear approximation is smaller $(-\nabla_x G)$, we can deduce the potential also went down. Of course, we can only use this direction for as far as the linear approximation holds.

When minimizing our potential function, we need to show that our solution still stays feasible, and that at each step a reasonable reduction in the potential is achieved. We have that the gradient $g = \nabla_x G$. Observe that if the potential function is not minimized, we will have a reasonably large gradient.

Problem 1: What if $g = \nabla_x G$ is not a feasible direction?

Solution 1: If $g = \nabla_x G$ is very perpendicular to the feasible space of the primal then we move in the dual direction instead, so a small step will improve the potential a lot. Thus, we project G onto the nullspace(A) to get d. Then, $A(x+d) = Ax = b$, and for a sufficiently small step, $x \geq 0$.

Problem 2: The potential reduction is proportional to the length of d. What if d too small?

Solution 2: In this case, we move s (actually y) by $g - d$ which will be large if d is small. This leads us to the following claim:

Claim 1 *:* We either take a big step in the primal or a big step in the dual.

Proof: Consider if d (perpendicular to A) has $Ad = 0$, then it is a good primal move. Conversely, the part spanned by A has $g - d = wA$, so we can choose $y' = y + w$ and get $s' = c - Ay' = c - Ay - (g - d) = s - (g - d)$.

We note $dG/dx_j = s_j/(xs) - 1/x_j$ and $dG/ds_j = x_j/(xs) - 1/s_j = (x_j/s_j)dG/dx_j \approx dG/dx_j$