Linear Programming II

David Woodruff

Outline

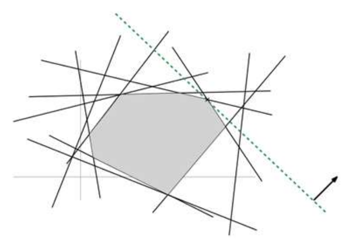
- Another linear programming example – l1 regression
- Seidel's 2-dimensional linear programming algorithm
- Ellipsoid algorithm, and continued discussion of simplex algorithm

L1 Regression

- Input: n x d matrix A with n larger than d, and n x 1 vector b
- Find x with Ax = b
- Unlikely an x exists, so instead compute $\min_x \sum_{i=1,\ldots,n} |A_i \cdot x - b_i|$
- Solve with linear programming? How to handle the absolute values?
- Create variables s_i for i = 1, …, n with $s_i \geq 0$
  - Also have variables $x_1, \ldots, x_d$
- Add constraints $A_i \cdot x - b_i \leq s_i$ and $-(A_i \cdot x - b_i) \leq s_i$ for i = 1, …, n
- What should the objective function be?
- min $\sum_{i=1,\ldots,n} s_i$

Seidel's 2-Dimensional Algorithm

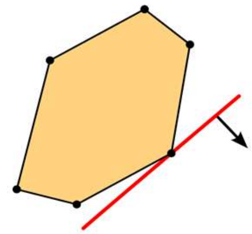
- Variables x_1, x_2
- Constraints $a_1 \cdot x \leq b_1, \ldots, a_m \cdot x \leq b_m$
- Maximize $c \cdot x$
- Start by making sure the program has bounded objective function value



What if the LP is unbounded?

- Add constraints $-M \leq x_1 \leq M$ and $-M \leq x_2 \leq M$ for a large value M
- How large should M be?
- Maximum, if it were bounded, occurs at the intersection of two constraints $ax_1 + bx_2 = c$ and $ex_1 + fx_2 = d$
  $\begin{bmatrix} a & b \\ e & f \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} = \begin{bmatrix} c \\ d \end{bmatrix}$
- If a, b, e, f, c, d are specified with L bits, can show $|x_1|, |x_2|$ specified with O(L) bits
- Can evaluate the objective function on each of the 4 corners of the box to find two constraints c_1, c_2 which give the maximum

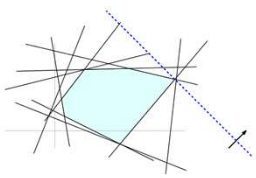
What Convexity Tells Us

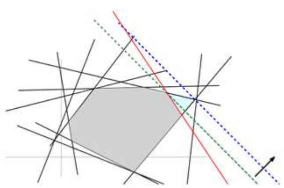


- Maximizing a linear function over the feasible region finds a tangent point
- What's a super naïve $O(m^3)$ time algorithm?
- Find the intersection of each pair of constraints, compute its objective function value, and make sure this point is feasible for all constraints
- What's a less naïve $O(m^2)$ time algorithm?

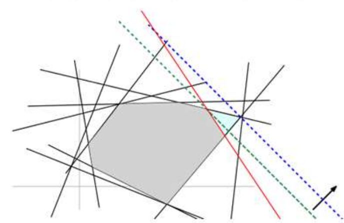
An $O(m^2)$ Time Algorithm

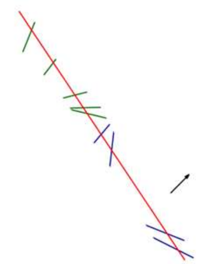
- Order the constraints $a_1 \cdot x \leq b_1, \ldots, a_m \cdot x \leq b_m, c_1, c_2$
- Recursively find optimum point x^* of $a_2 \cdot x \leq b_2, \ldots, a_m \cdot x \leq b_m, c_1, c_2$
- If $a_1 x^* \leq b_1$, then x^* is overall optimum
- Otherwise, new optimum intersects the line $a_1 x^* = b_1$
- Need to solve a 1-dimensional problem





1-Dimensional Problem





- Takes O(m) time to solve
- Note: new optimum might not be determined by one of the two constraints determining the old optimum

An $O(m^2)$ Time Algorithm

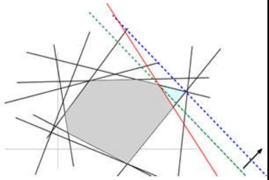
- Recursively find optimum point x^* of $a_2 \cdot x \leq b_2, \ldots, a_m \cdot x \leq b_m, c_1, c_2$
- If $a_1 x^* \leq b_1$, then x^* is still optimal
- Otherwise, new optimum intersects the line $a_1 \cdot x = b_1$
- Solve a 1-dimensional problem in O(m) time
- T(m) = T(m-1) + O(m) = $O(m^2)$ time
- Can we get O(m) time?

Seidel's O(m) Time Algorithm

- Order constraints randomly: $a_{i_1} \cdot x \leq b_{i_1}, \ldots, a_{i_m} \cdot x \leq b_{i_m}, c_1, c_2$
  - Leave c_1, c_2 at the end
- Recursively find the optimum x^* of $a_{i_2} \cdot x \leq b_{i_2}, \ldots, a_{i_m} \cdot x \leq b_{i_m}, c_1, c_2$
- Case 1: If $a_{i_1} \cdot x^* \leq b_{i_1}$, then x^* is overall optimum
  - O(1) time
- Case 2: If $a_{i_1} \cdot x^* > b_{i_1}$, then we need to intersect the line $a_{i_1} \cdot x = b_{i_1}$ with each other line $a_{i_j} \cdot x = b_{i_j}$ and solve a 1-dimensional problem in O(m) time

Backwards Analysis

- Let x^* be the optimum point of $a_{i_2} \cdot x \leq b_{i_2}, \ldots, a_{i_m} \cdot x \leq b_{i_m}, c_1, c_2$
- What is the chance that $a_{i_1} \cdot x^* > b_{i_1}$?
- Suppose the optimum x' of $a_{i_1} \cdot x \leq b_{i_1}, \ldots, a_{i_m} \cdot x \leq b_{i_m}, c_1, c_2$ is the intersection of two constraints $a_{i_j} \cdot x = b_{i_j}$ and $a_{i_{j'}} \cdot x = b_{i_{j'}}$
- If we've seen these two constraints, then the new constraint $a_{i_1} \cdot x \leq b_{i_1}$ can't change the optimum. Otherwise, optimum would change
- Expected time for processing the last constraint is at most (1-2/m) $\cdot$ O(1) + (2/m) $\cdot$ O(m) = O(1)



Backwards Analysis

- We process the randomly ordered constraints in reverse order:

  $a_{i_1} \cdot x \leq b_{i_1}, \ldots, a_{i_m} \cdot x \leq b_{i_m}, c_1, c_2$
- When processing the last constraint of:

  $a_{i_j} \cdot x \leq b_{i_j}, \ldots, a_{i_m} \cdot x \leq b_{i_m}, c_1, c_2$

  the expected amount of time is

  (1-2/(m-j+1)) $\cdot$ O(1) + (2/(m-j+1)) $\cdot$ O(m-j+1) = O(1)
- The expected total time to process m constraints is $\sum_j O(1) = O(m)$, as desired!
- Formally, let T(m) be the expected time to process all m constraints

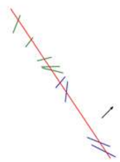
  T(m) $\leq$ (1-2/m) O(1) + (2/m) $\cdot$ $O(m)$ + T(m-1)

  = O(1) + T(m-1)

  = O(m). Also add initial constant time for finding c_1, c_2

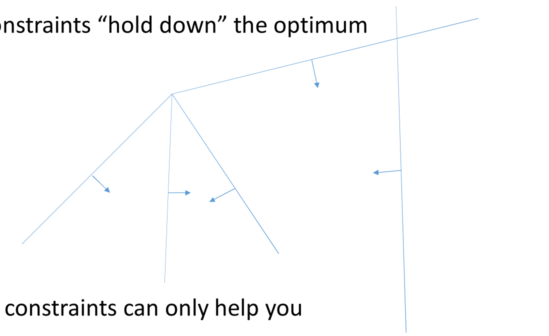
What if the LP is Infeasible?

- Let j be the largest index for which $a_{i_j} \cdot x \le b_{i_j}, \ldots, a_{i_m} \cdot x \le b_{i_m}, c_1, c_2$ is infeasible. That is, $a_{i_{j+1}} \cdot x \le b_{i_{j+1}}, \ldots, a_{i_m} \cdot x \le b_{i_m}, c_1, c_2$ is feasible
- Since $a_{i_{j+1}} \cdot x \le b_{i_{j+1}}, \ldots, a_{i_m} \cdot x \le b_{i_m}, c_1, c_2$ is randomly ordered, we spend an expected O(m) time to process such constraints
- When processing $a_{i_j} \cdot x \le b_{i_j}$ we will find the constraints are infeasible in O(m) time when solving the 1-dimensional problem



What If More than 2 lines Intersect at a Point?

- 2 of the constraints "hold down" the optimum



- Additional constraints can only help you

Higher Dimensions?

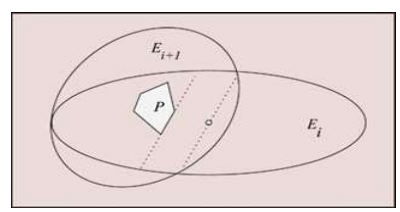
- The probability that our optimum changes is now at most d/m instead of 2/m
- When we find a violated constraint, we need to find a new optimum
- New optimum inside this hyperplane
  - Project each constraint into this hyperplane
  - Solve a (d-1)-dimensional linear program on m-1 constraints to find optimum
  - $T(d, m) \le T(d, m-1) + O(d) + \frac{d}{m}[O(dm) + T(d-1, m-1)]$
  - T(d,m) = O(d! m)

Ellipsoid Algorithm

Solves feasibility problem

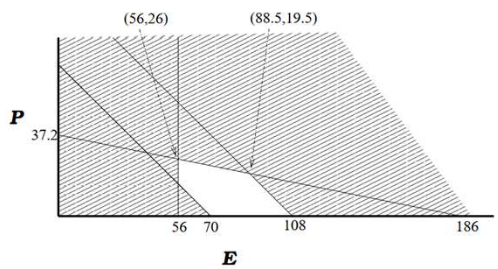
Replace objective function with constraint, do binary search

Replace "minimize $x_1 + x_2$" with $x_1 + x_2 \le \lambda$



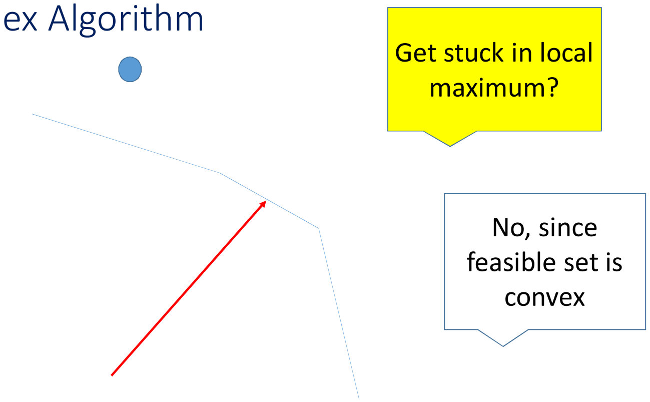
Can handle exponential number of constraints if there's a separation oracle

Simplex Algorithm



Start at vertex of the feasible region (polyhedron in high dimensions)
Look at cost of objective function at each neighbor
Move to neighbor of maximum cost
Always make progress, but could take exponential time (in high dimensions)

Simplex Algorithm



Other Annoyances I

- How to start at a vertex of the feasible region?
- $Ax \leq b$
  $x \geq 0$
- What if it's not even feasible?
- Introduce "slack" variable s. Consider:
- $\min s$
  subject to $Ax \leq b + s \cdot 1^m$
  $x \geq 0, s \geq 0, s \leq \max_i -b_i$
- Feasible. Can run simplex starting at x = 0^n and s = $\max_i -b_i$
- If original LP is feasible, minimum achieved when s = 0, and x that is output is a vertex in the feasible region of original LP

Other Annoyances II

- What if the feasible region is unbounded?
  - Ok, as long as objective function is bounded
- What if objective function is unbounded?
  - Output ∞, how to detect this?
- Many ways
  - see one based on duality in the next lecture
  - include constraints $-M \leq x_i \leq M$ for all i, for a very large value M
  - can efficiently find M to ensure if solution is finite, still find the optimum