Linear Programming in Higher Dimensions

Piotr Indyk
Linear Programming

Maximize: \[ c_1 x_1 + c_2 x_2 + \cdots + c_d x_d \]

Subject to: \[ a_{1,1} x_1 + a_{1,2} x_2 + \cdots + a_{1,d} x_d \leq b_1 \]
\[ a_{2,1} x_1 + a_{2,2} x_2 + \cdots + a_{2,d} x_d \leq b_2 \]
\[ \vdots \]
\[ a_{n,1} x_1 + a_{n,2} x_2 + \cdots + a_{n,d} x_d \leq b_n \]
Linear Programming in 2D

October 21, 2003

Lecture 16: Linear Programming in Higher Dimensions
An Infeasible Linear Program
An Unbounded LP
Incremental Algorithm

• Choose two constraints and initialize the solution
• Add new constraints one by one, keeping track of current optimum
Probability of update at round $i$

Fix first $i$ constraints:

Update only if the $i$th constraint is one of the two *defining constraints*

$$P \leq \frac{2}{i - 2}$$
Expected Run–Time Analysis

Expected time spent updating:

\[
E\left[ \sum_{i=3}^{n} T_i \right] = \sum_{i=3}^{n} E[T_i] = \sum_{i=3}^{n} P(\text{Update at round } i) O(i)
\]

\[
\leq \sum_{i=3}^{n} \frac{2}{i-2} O(i) = O(n)
\]
What about $d > 2$?

- Incrementally add new constraints
- Probability of update: $d / (i-d)$
- On update: solve $d$-dimensional
  
  \[ T(d,n) \leq O(dn) + \sum_{i=d+1}^{n} \frac{d}{i-d} T(d-1,i-1) \]

  \[ T(d,n) = O(d!n) \]
This Lecture

• $O(d!n)$ is not optimal:
  – $O(d^2n + d^{O(1)}d)$ [Clarkson]
  – A reduction from $(n,d)$ –LP to a small number of $(O(d^2),d)$ –LP’s

• Extensions:
  – $n^{O(\sqrt{d})}$ [Kalai, Matousek–Sharir–Welzl]
  – $O(d^2n + d^{O(\sqrt{d})})$ [combined]
Notation

- $H$: set of $n$ constraints
- $v(H)$: optimum subject to $H$
- A basis $B$ for $H$: minimal set of constraints such that $v(B) = v(H)$
- We have $|B| = d$
Random Sampling I

• SolveLP1(H):
  – G=∅
  – Repeat:
    • R=random subset of H, |R|=r
    • v= SolveLP( G+R )
    • V=set of constraints in H violated by v
    • If |V| ≤ t, then G=G+V
  – Until V=∅

• Correctness ?
• Running time analysis ?
Analysis

• Each time we augment $G$, we add to $G$ a new constraint from the basis $B$ of $H$
  – If $v$ did not violate any constraint in $B$, it would be optimal
  – So $V$ must contain an element from $B$, which was not in $G$ earlier

• We can augment $G$ at most $d$ times

• The number of constraints in the recursive call is $|R| + |G| \leq r + dt$

• What is the probability of augmentation?
Sampling Lemma

Lemma: The expected number of constraints \( V \) that violate \( v(G+R) \) is at most \( nd/r \).

Proof:

- Define a 0/1 random variable \( d(R,h) \), which is \( =1 \) iff \( h \) violates \( v(G+R) \)
- Need to bound

\[
E_{R} [\sum_{h} d(R,h)] = \sum_{|R|=r} \sum_{h} d(R,h) / \#R \\
= \sum_{|Q|=r+1} \sum_{h \in Q} d(Q-\{h\},h) / \#R \\
= [\#Q * (r+1) / \#R] * \Pr_{Q,h \in Q} [d(Q-\{h\},h)] \\
\leq n * d/(r+1)
\]
Analysis

• $t=2nd/r \rightarrow$ expected # iterations per augmentation is constant

• Number of constraints in the recursive call is $r+O(d^2n/r) = O(r)$ for $r=dn^{1/2}$

• Total expected time

$$T_{LP_1}(n) \leq 2d \cdot T_{LP}(dn^{1/2}) + O(d^2n)$$
Analysis ctd.

- Can use Seidel’s algorithm for LP
- This gives us $O(d^2 n + d^* d \cdot n^{1/2} \cdot d!)$
- We get better time if $\text{LP} = \text{LP2}$
- Idea: reduce the sample size
Random Sampling II

• **SolveLP2(**H**):**
  – \(G=\emptyset\)
  – Repeat:
    • \(R=\text{random subset of } H, \ |R|=r\)
    • \(v=\text{SolveLP}(R)\)
    • \(V=\text{multiset} \text{ of constraints in } H \text{ violated by } v\)
    • If \(|V| \leq t\), then \(H=H+V\)
  – Until \(V=\emptyset\)

• As before, set \(t=2^*|H|d/r\)
  → augmentation performed with prob. \(>1/2\)
Need to bound \#augmentations

- Fix a basis $B$ for $H$
- On the one hand:
  - In one iteration, the multiplicity of at least one constraint in $B$ is doubled
  - In $kd$ iterations, $|B| \geq 2^k$
- On the other hand:
  - In one iteration, $|H|$ increases by $\leq 2 |H|d/r$
  - After $kd$ iterations:
    $|B| \leq |H| \leq n (1 + 2d/r)^{kd} \leq n \exp(2kd^2/r) = n \exp(2d^2/r)^k$
- Therefore, the total number of iterations is $O(dk)$, if $k$ such that $2^k > n \exp(2d^2/r)^k$
Analysis, ctd.

\[ 2^k > n \exp(2d^2/r)^k \]

- Set \( r = 4d^2 \rightarrow 2^k > n (e^{1/2})^k \)
- We get \( k = O(\log n) \)
- The total number of iterations is \( O(d \log n) \)
Total Time

• The expected time
  \( T_{LP2}(n) = d \log n \left[ T_{LP}(4d^2) + dn \right] \)

• Plug in Seidel into LP2
  \( T_{LP2}(n) = O(d \log n \left( d^2 d! + dn \right)) \)

• Plug in LP2 into LP1
  \( T_{LP1}(n) = O(2d \left[ d \log n \left( d^2 d! + d^2 n^{1/2} \right) \right] + d^2 n) \)

• After some cleaning
  \( T_{LP1}(n) = O(d^4 d! \log n + d^2 n) = O(d^5 d! + d^2 n) \)