CPSC 420+500: Advanced Algorithm Design and Analysis
January 18 & 20, 2017

Scribe: Laura Barton

Lower Bound on Element Uniqueness

Lemma Any linear decision tree that computes some function F has height:

$$\lceil \log_3 \big(\sum_{outputs\ t} \#connected\ components\ of\ F_t \big) \rceil$$

i.e., $\log_3$ of the number of connected components for each of the possible outputs of F.

Theorem Any linear decision tree that computes Element Uniqueness has height $\Omega(n \log n)$. The NO components for Element Uniqueness = 1, since all hyperplanes are joined at the origin.

Proof. Let

$$x = (1, 2, ..., n)$$

$$y = (2, 1, 3, ..., n)$$

Are x and y in the same connected component? *No.* If you have $(1, 2)$ and $(2, 1)$, at some point a path between them would have to cross a NO region.

Let v be a vector of n unique numbers and $v \neq v'$ be any permutation of v. There myst be indices i and j such that v_i is smaller than v_j and v'_i is bigger than v'_j, i.e., $v_i < v_j, v'_i > v'_j$. Any continuous path from v to v' must contain a point z with $z_i = z_j$ (by the intermediate value theorem*). z is a NO input, so v and v' are not in the same connected component.

***Intermediate Value Theorem** Let p be a path from points x to y where

$$p : [0, 1] \in \mathbb{R}^n, p(0) = x, p(1) = y$$

Let $q(t) = p(t)_j - p(t)_i$ (the difference between i^{th} and j^{th} coordinate in point $p(t)$). At some point, $q(t) = 0$ because $q(0) > 0, q(1) < 0$ so it had to have crossed from positive to negative.

Since there are $n!$ different permutations of a list of n numbers, and none of those permutations are in the same component, there are at least $n!$ different connected components, i.e., $\#F_{YES} \geq n!$. Therefore:

$$\#F_{YES} + F_{NO} = n! + 1$$

Plugging this into our formula for the height of the decision tree:

$$\lceil \log_3 (\sum \#F_t) \rceil = \lceil \log_3 (n! + 1) \rceil$$

$$\lceil \log (n!) \rceil = n \log n \in \Omega(n \log n)$$

∎

Practice Question: reduce Element Uniqueness to Convex Hull.

However, Linear Decision Trees aren't powerful enough to calculate the Convex Hull. Algebraic Decision Trees of the d^{th} order use internal node tests that are d^{th} order polynomials (i.e., Linear Decision Trees are Algebraic Decision Trees of the 1^{st} order.)

Jarvis March is $\in O(nh)$ and Graham's Scan is $\in O(n \log n)$ (where h is the number of points on the hull); is there a more efficient algorithm?

Chan's Algorithm

($\sim$1996) $\in O(n \log h)$ Given n points in set P and a guess h:

1. $O(n)$ Divide points into $\lceil n/h \rceil$ groups of size h
2. $O(h \log h)$ *per group*, $\in O(n \log h)$ *total* Use Graham's Scan to find the convex hull of each group
3. $O(n)$ Find the lowest point p_0 in P
4. $O(h(n/h) \log h) = O(n \log h)$. Do giftwrapping (Jarvis March) for h steps.

   Note that we don't need to scan all the points in P, since we have h hulls that are now sorted within themselves; we can find the rightmost tangent from p_i for each sub-hull using binary search. n/h binary searches each taking $O(\log h)$ time take a total of $O(n/h \log h)$ time. Since we do this gift-wrapping step h times, the total time is $O(hn/h \log h) = O(n \log h)$. Let p_{i+1} = point on the rightmost out of those tangents.

5. If $p_{i+1} = p_0$ in $\leq h$ steps, then output the hull. Otherwise output that h is too small.

Where does our guess h come from? Generate the guesses for the "true" h, $h*$, by squaring each time:

$$h = 4, 16, 256...$$

$$t^{th} try = 2^{2^t}$$

The time complexity of all tries until $h \geq h*$

$$\sum_{h=2^{2^t}} O(n \log h) \ until \ h \geq h*$$

$$\sum_{t=1}^{\lceil \log \log h* \rceil} O(n2^t) = n[\sum_{t=1}^{\lceil \log \log h* \rceil} O(2^t) = O(n \log h*)$$

$$\approx 2^{lglgh*}$$