MS&E 317/CS 263: Algorithms for Modern Data Models, Spring 2014
http://msande317.stanford.edu.
Instructors: Ashish Goel and Reza Zadeh, Stanford University.

Lecture 15, 5/21/2014. Scribed by Schuyler Smith.

15.1 Performance of our Incremental Facility Location Algorithm

Recall our algorithm for incremental facility location, which accepts node v_t at time t and maintains a set F_t of facilities:

1. Initialize $F_0 = \{\}$.
2. For each v_t, compute $\delta_{v_t} = \min_{w \in F_{t-1}} d(v_t, w)$ (that is, the distance to the closest facility we've already built).
3. Open a new facility at v_t with probability $\frac{\delta_{v_t}}{f}$, where f is the cost of opening a facility. That is, $F_t = F_{t-1} \cup \{v_t\}$ with probability $\frac{\delta_{v_t}}{f}$, and $F_t = F_{t-1}$ otherwise.

Define the optimal set of facilities for this problem to be $F^* = \{c_1^*, \ldots, c_k^*\}$, define $C_i^* = \{v \mid \forall j \neq i,\ d(v, c_i^*) < d(v, c_j^*)\}$ to be the sets of nodes that are "covered" by facility c_i^* in the optimal solution, and define $a_i^* = \frac{\sum_{v \in C_i^*} d(v, c_i^*)}{|C_i^*|}$ to be the average distance to c_i^* among nodes in C_i^*. Then the total cost of the optimal solution is $f|F^*| + \sum_i a_i^* \cdot |C_i^*|$.

We will show that if we assume the nodes arrive in random order, then the expected cost of the solution produced by our algorithm is within a constant factor of this optimal cost. We don't particularly care what this factor is, just that it exists.

Proof: We'll consider each set of nodes C_i^* separately. We know the nodes in C_i^* will be processed in some random order by our algorithm (and interleaved with all other nodes, which don't matter). First, split the nodes of C_i^* into two sets: Let G_i^* be the "good" nodes, the half that are closest to c_i^*, and let B_i^* be the "bad" nodes, the other half.

First, consider G_i^*. We can further break G_i^* into two parts as described below. Hence, we have three cases: two for G_i^* and one for B_i^*.

Case 1: Consider the nodes that arrive *after* a node g in G_i^* has been chosen as a facility location by our algorithm. Note this set may be empty. Let g' be one of these nodes. Directly from our algorithm we know the expected cost of g' will be $f \cdot \frac{\delta_{g'}}{f} + \delta_{g'} \cdot \left(1 - \frac{\delta_{g'}}{f}\right) < 2\delta_{g'}$. Additionally, we know $\delta_{g'} \leq d(g, g')$ since g is a facility already picked by our algorithm and $\delta_{g'}$ is the distance to the facility closest to g'. But then

$$2\delta_{g'} \leq 2d(g, g') \leq 2\left(d(g, c_i^*) + d(g', c_i^*)\right) \leq 2\left(2a_i^* + 2a_i^*\right) = 8a_i^*$$

since from Markov's inequality we expect to find a node in B_i^* within $2a_i^*$ of c_i^* (since a_i^* is the average distance to c_i^* in all of C_i^*) and all good nodes are closer than all bad nodes, so in expectation all good nodes, including g and g', are within $2a_i^*$ of c_i^*.

Hence, the expected cost of each g' is $\leq 8a_i^*$, which is a constant factor off from the average cost a_i^* of nodes in C_i^*.

Case 2: Consider the nodes in G_i^* that arrive before any have been chosen by our algorithm as facility locations. Let $\delta_1, \ldots \delta_k$ be their δ-values upon arrival. Then the expected cost of the first node is δ_1 (since we know it wasn't chosen as a facility location). The expected cost of the second node is $\delta_2 \left(1 - \frac{\delta_1}{f}\right)$ (i.e. δ_2 if the first node isn't chosen as a facility location, and otherwise we already analyzed the second node it in the first case). Similarly, the expected cost of the third node is $\delta_3 \left(1 - \frac{\delta_2}{f}\right)\left(1 - \frac{\delta_1}{f}\right)$ (δ_3 only if the first and second nodes weren't chosen as facility locations), and so on, so the expected cost of the kth node is $\delta_k \prod_{i=1}^{k-1}\left(1 - \frac{\delta_i}{f}\right)$.

Hence, we expect the cost of all k nodes to be

$$\delta_1 + \delta_2 \left(1 - \frac{\delta_1}{f}\right) + \delta_3 \left(1 - \frac{\delta_2}{f}\right)\left(1 - \frac{\delta_1}{f}\right) + \ldots$$

which can be rewritten as

$$\delta_1 + \left(1 - \frac{\delta_1}{f}\right)\left[\delta_2 + \left(1 - \frac{\delta_2}{f}\right)\left[\delta_3 + \left(1 - \frac{\delta_3}{f}\right)[\delta_4 + \ldots]\right]\right]$$

Define $S_i = \delta_i + \left(1 - \frac{\delta_i}{f}\right)\left[\delta_{i+1} + \left(1 - \frac{\delta_{i+1}}{f}\right)[\delta_{i+2} + \ldots]\right]$, so S_1 is the cost of all k nodes. Then

$$S_i = \delta_i + \left(1 - \frac{\delta_i}{f}\right) S_{i+1} = \delta_i + S_{i+1} - S_{i+1}\frac{\delta_i}{f} = S_{i+1} + \delta_i \left(1 - \frac{S_{i+1}}{f}\right)$$

so if $S_{i+1} < f$ then $S_i > S_{i+1}$, and if $S_{i+1} > f$ then $S_i < S_{i+1}$. Hence, as i decreases, S_i will tend toward f. Hence, the expected cost from this case will be about f.

Case 3: Now, we'll consider nodes in B_i^*. Here we'll use our random-order assumption for the first time. Consider the interleaving of bad and good nodes as they arrive: since it's random and we have an equal number of bad and good nodes, on average we'll get streaks of two bad or two good nodes in a row.

Hence, we expect to see about one bad node before we see any good nodes (we start with a good node with probability $\frac{1}{2}$, or with a bad streak of length about 2 with probability $\frac{1}{2}$). The expected cost of this bad node will be at most f.

Similarly, after each good node we encounter in the stream, we expect to see about one bad node before the next good node. Suppose we get a bad node b_k after good node g_j. Then the expected cost of this node will be $\delta_{b_k} \leq d(b_k, g_j) + \delta_{g_j} \leq d(b_k, c_i^*) + d(g_j, c_i^*) + \delta_{g_j}$, since at worst we can go through g_j to get to a nearby facility already picked by our algorithm.

But we already know this is $\leq d(b_k, c_i^*) + 2a_i^* + \delta_{g_j}$, where $d(b_k, c_i^*)$ is just the optimal cost for b_k, $2a_i^*$ is acceptable, and δ_{g_j} is the expected cost of g_j, which we've already shown is acceptable.

Hence, the expected cost of our algorithm for the good nodes is at most $f + (|G_i^*| - k)8a_i^*$, and the expected cost for the bad nodes is at most this again plus $|B_i^*|2a_i^*$ plus the optimal cost of the nodes in B_i^*, plus f. This is all clearly within a constant factor of $f + |C_i^*|a_i^*$, the optimal cost for the nodes in C_i^*, so we're done. ■

15.2 Implementing Locality Sensitive Hashing

Recall that we want our locality-sensitive hash function h to have parameters c, R, p_1 and p_2 such that $p_1 \leq \text{Prob}\left[h(x) = h(y) \mid d(x, y) \leq R\right]$ and $p_2 \geq \text{Prob}\left[h(x) = h(y) \mid d(x, y) > cR\right]$.

The correct hash family to use to implement h will depend on the underlying metric space. We'll present a family for l_2. Recall that with the l_2-norm, $d(x, y) = ||x - y|| = \sqrt{\sum_{i=1}^{D}(x_i - y_i)^2}$ where D is the dimension.

We can break a D-dimensional space into a linear set of Δ-width buckets in the following way:

1. Choose a (random) line R, and an offset $\delta \in [0, \Delta)$.
2. For each point x, project x onto R, so $x \to \pi(x) = x \cdot R$ (where $\cdot$ is the inner/dot product).
3. Compute the integer $n = \left\lfloor \frac{||\pi(x)|| + \delta}{\Delta} \right\rfloor$.
4. Define $h(x) = h'(n)$, where h' is any suitable hash function on integers.

Hence, we partition space into many cylinders, each of height Δ and infinite radius. The direction of R determines the orientation of the cylinders, and all points in the same cylinder will hash to the same value.

We can choose $R = (r_1, \ldots r_D)$ randomly by choosing $r_i \in N(0; 1)$.

Note that then $\pi(x) - \pi(y) = x \cdot R - y \cdot R = (x - y) \cdot R$, which will be distributed as $||x - y|| \times N(0; 1)$, so $E\left[(\pi(x) - \pi(y))^2\right] = ||x - y||^2$, so $E\left[|\pi(x) - \pi(y)|\right] = ||x - y||$.

Now, note that if $|\pi(x) - \pi(y)| > \Delta$ then $h(x) \neq h(y)$, otherwise $h(x) = h(y)$ (i.e. x and y are placed in the same cylindrical bucket) with probability $1 - \frac{|\pi(x) - \pi(y)|}{\Delta}$. Hence, $\text{Prob}\left[h(x) = h(y)\right] = 1 - \frac{|\pi(x) - \pi(y)|}{\Delta} = 1 - \frac{||x - y||}{\Delta} = 1 - \frac{d(x, y)}{\Delta}$.

For example, let $\Delta = 2cR$. Then if $d(x, y) < R$ then $h(x) = h(y)$ with probability at least $1 - \frac{R}{\Delta} = 1 - \frac{1}{2c}$, and if $d(x, y) > cR$ then $h(x) = h(y)$ with probability at most $1 - \frac{cR}{\Delta} = \frac{1}{2}$.

15.3 PageRank

See the lecture slides.