President’s Day Lecture:
Advanced Nearest Neighbor Search

[Advanced Algorithms, Spring’17]
Announcements

- Evaluation on CourseWorks
- If you think homework is too easy (or too hard):
  - mark “appropriateness of workload”
### Time-Space Trade-offs (Euclidean)

<table>
<thead>
<tr>
<th>Query Time</th>
<th>Space</th>
<th>Time</th>
<th>Comment</th>
<th>Reference</th>
</tr>
</thead>
<tbody>
<tr>
<td>low</td>
<td>low</td>
<td>$n \approx n$</td>
<td>$n^\sigma \sigma = 2.09/c$</td>
<td>[Ind’01, Pan’06]</td>
</tr>
<tr>
<td>high</td>
<td>high</td>
<td>$n^{1+\rho}$</td>
<td>$n^\rho \rho = 1/c$</td>
<td>[IM’98, DIIM’04]</td>
</tr>
<tr>
<td>medium</td>
<td>medium</td>
<td>$n^{1+\rho}$</td>
<td>$n^\rho \rho = 1/c^2$</td>
<td>[AI’06]</td>
</tr>
<tr>
<td>high</td>
<td>low</td>
<td>$n^{1+\rho}$</td>
<td>$n^\rho \rho \geq 1/c^2$</td>
<td>[MNP’06, OWZ’11]</td>
</tr>
<tr>
<td>low</td>
<td>high</td>
<td>$n^{4/\epsilon^2}$</td>
<td>$O(d \log n) c = 1 + \epsilon$</td>
<td>[KOR’98, IM’98, Pan’06]</td>
</tr>
<tr>
<td>low</td>
<td>low</td>
<td>$n^{o(1/\epsilon^2)}$</td>
<td>$\omega(1)$ memory lookups</td>
<td>[AIP’06]</td>
</tr>
</tbody>
</table>

Note: $\rho = 1/c$ and $\sigma = O(1/c^2)$ in Euclidean space.
Near-linear Space for \( \{0,1\}^d \)

[Indyk’01, Panigrahy’06]

- **Setting:**
  - Close: \( r = \frac{d}{2c} \Rightarrow P_1 = 1 - \frac{1}{2c} \)
  - Far: \( cr = \frac{d}{2} \Rightarrow P_2 = \frac{1}{2} \)

- **Algorithm:**
  - Use one hash table with \( k = \frac{\log n}{\log 1/P_2} = \alpha \cdot \ln n \)
  - On query \( q \):
    - compute \( w = g(q) \in \{0,1\}^k \)
    - Repeat \( R = n^\sigma \) times:
      - \( w' \): flip each \( w_j \) with probability \( 1 - P_1 \)
      - look up bucket \( w' \) and compute distance to all points there
    - If found an approximate near neighbor, stop

Sample a few buckets in the same hash table!
Theorem: for $\sigma = \Theta \left( \frac{\log c}{c} \right)$, we have:

- $\Pr[\text{find an approx near neighbor}] \geq 0.1$
- Expected runtime: $O(n^\sigma)$

Proof:

- Let $p^*$ be the near neighbor: $||q - p^*|| \leq r$
- $w = g(q)$, $t = ||w - g(p^*)||_1$
- Claim 1: $\Pr_{g} \left[ t \leq \frac{k}{c} \right] \geq \frac{1}{2}$
- Claim 2: $\Pr_{g, w'} \left[ w' = g(p) \mid ||q - p||_1 \geq \frac{d}{2} \right] \leq \frac{1}{n}$
- Claim 3: $\Pr[w' = g(p^*) \mid \text{Claim 1}] \geq 2n^{-\sigma}$
- If $w' = g(p^*)$ at least for one $w'$, we are guaranteed to output either $p^*$ or an approx. near neighbor
## Beyond LSH

<table>
<thead>
<tr>
<th>Space</th>
<th>Time</th>
<th>Exponent</th>
<th>(c = 2)</th>
<th>Reference</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Hamming space</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>(n^{1+\rho})</td>
<td>(n^\rho)</td>
<td>(\rho = 1/c)</td>
<td>(\rho = 1/2)</td>
<td>[IM'98]</td>
</tr>
<tr>
<td></td>
<td></td>
<td>(\rho \geq 1/c)</td>
<td></td>
<td>[MNP'06, OWZ'11]</td>
</tr>
<tr>
<td>(n^{1+\rho})</td>
<td>(n^\rho)</td>
<td>(\rho \approx \frac{1}{2c - 1})</td>
<td>(\rho = 1/3)</td>
<td>[AINR'14, AR'15]</td>
</tr>
<tr>
<td><strong>Euclidean space</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>(n^{1+\rho})</td>
<td>(n^\rho)</td>
<td>(\rho \approx 1/c^2)</td>
<td>(\rho = 1/4)</td>
<td>[AI'06]</td>
</tr>
<tr>
<td></td>
<td></td>
<td>(\rho \geq 1/c^2)</td>
<td></td>
<td>[MNP'06, OWZ'11]</td>
</tr>
<tr>
<td>(n^{1+\rho})</td>
<td>(n^\rho)</td>
<td>(\rho \approx \frac{1}{2c^2 - 1})</td>
<td>(\rho = 1/7)</td>
<td>[AINR'14, AR'15]</td>
</tr>
</tbody>
</table>
New approach?

- A random hash function, chosen after seeing the given dataset
- Efficiently computable

Data-dependent hashing
Construction of hash function

[A.-Indyk-Nguyen-Razenshteyn’14, A.-Razenshteyn’15]

- Two components:
  - Nice geometric structure
  - Reduction to such structure

has better LSH

data-dependent
Nice geometric structure

- Points on a unit sphere, where
  - $cr \approx \sqrt{2}$, i.e., far pair is (near) orthogonal
  - this would be distance if the dataset were random on sphere
- Close pair: $r = \frac{\sqrt{2}}{c}$

- Query:
  - at angle 45’ from near-neighbor
Alg 1: Hyperplanes

[Charikar’02]

- Sample unit \( r \) uniformly, hash \( p \) into \( sgn\langle r, p \rangle \)
  - \( \Pr[h(p) = h(q)] = 1 - \alpha / \pi \)
  - where \( \alpha \) is the angle between \( p \) and \( q \)
- \( P_1 = 3/4 \)
- \( P_2 = 1/2 \)
- \( \rho \approx 0.42 \)
Alg 2: Voronoi

[A.-Indyk-Nguyen-Razenshteyn’14] based on [Karger-Motwani-Sudan’94]

- Sample $T$ i.i.d. standard $d$-dimensional Gaussians
  
  $g_1, g_2, \ldots, g_T$

- Hash $p$ into
  
  $h(p) = \arg\max_{1 \leq i \leq T} \langle p, g_i \rangle$

- $T = 2$ is simply Hyperplane LSH
Hyperplane vs Voronoi

- **Hyperplane with** $k = 6$ hyperplanes
  - Means we partition space into $2^6 = 64$ pieces
- **Voronoi with** $T = 2^k = 64$ vectors
  - $\rho = 0.18$
  - grids vs spheres
Reduction to nice structure (very HL)

Idea:
iteratively decrease the radius of minimum enclosing ball OR make more isotopic

Algorithm:
- find dense clusters
  - with smaller radius
  - large fraction of points
- recurse on dense clusters
- apply VoronoiLSH on the rest
  - recurse on each “cap”
  - eg, dense clusters might reappear

Why ok?
- no dense clusters
- like “random dataset” with radius=100cr
- even better!

radius = 99cr

*picture not to scale & dimension
Hash function

- Described by a tree (like a hash table)

radius = 100 cr

*picture not to scale & dimension*
Dense clusters

- Current dataset: radius $R$
- A dense cluster:
  - Contains $n^{1-\delta}$ points
  - Smaller radius: $(1 - \Omega(\epsilon^2))R$
- After we remove all clusters:
  - For any point on the surface, there are at most $n^{1-\delta}$ points within distance $(\sqrt{2} - \epsilon)R$
  - The other points are essentially orthogonal!
- When applying Cap Carving with parameters $(P_1, P_2)$:
  - Empirical number of far pts colliding with query: $nP_2 + n^{1-\delta}$
  - As long as $nP_2 \gg n^{1-\delta}$, the “impurity” doesn’t matter!
Tree recap

- During query:
  - Recurse in all clusters
  - Just in one bucket in VoronoiLSH
- Will look in >1 leaf!
- How much branching?
  - **Claim**: at most $(n^\delta + 1)^{o\left(1/\epsilon^2\right)}$
  - Each time we branch
    - at most $n^\delta$ clusters (+1)
    - a cluster reduces radius by $\Omega(\epsilon^2)$
    - cluster-depth at most $100/\Omega(\epsilon^2)$
- Progress in 2 ways:
  - Clusters reduce radius
  - CapCarving nodes reduce the # of far points (empirical $P_2$)
- A tree succeeds with probability $\geq n \frac{1}{2\epsilon^2 - 1} o(1)$
NNS: conclusion

1. Via sketches

2. Locality Sensitive Hashing
   - Random space partitions
   - Better space bound
     - Even near-linear!

3. Data-dependent hashing even better
   - Used in practice a lot these days