Lecture 12:
More LSH
Data-dependent hashing
Announcements & Plan

• **PS3:**
  – Released tonight, due next Fri 7pm
• **Class projects:** think of teams!
• **I’m away until Wed**
  – Office hours on Thu after class
• **Kevin will teach on Tue**
• **Evaluation on courseworks next week**

• **LSH:** better space
• **Data-dependent hashing**
  – Scribe?
## Time-Space Trade-offs (Euclidean)

<table>
<thead>
<tr>
<th></th>
<th>Space</th>
<th>Time</th>
<th>Comment</th>
<th>Reference</th>
</tr>
</thead>
<tbody>
<tr>
<td>high</td>
<td>$\approx n$</td>
<td>$n^\sigma$</td>
<td>$\sigma = 2.09/c$</td>
<td>[Ind’01, Pan’06]</td>
</tr>
<tr>
<td>low</td>
<td>$n^{1+\rho}$</td>
<td>$n^\rho$</td>
<td>$\rho = 1/c$</td>
<td>[IM’98, DIIM’04]</td>
</tr>
<tr>
<td>medium</td>
<td>$n^{1+o(1/c^2)}$</td>
<td>$\omega(1)$ memory lookups</td>
<td></td>
<td>[PTW’08, PTW’10]</td>
</tr>
<tr>
<td>medium</td>
<td>$n^4/\varepsilon^2$</td>
<td>$O(d \log n)$</td>
<td>$c = 1 + \varepsilon$</td>
<td>[KOR’98, IM’98, Pan’06]</td>
</tr>
<tr>
<td>low</td>
<td>$n^{o(1/\varepsilon^2)}$</td>
<td>$\omega(1)$ memory lookups</td>
<td></td>
<td>[AIP’06]</td>
</tr>
</tbody>
</table>

### Table Notes:

- $\sigma = O(1/c^2)$
- $\rho = 1/c$
- $\rho = 1/c^2$
- $\rho \geq 1/c^2$
- 1 mem lookup

### References:

- [AI’06]
- [KOR’98, IM’98, Pan’06]
- [Ind’01, Pan’06]
- [IM’98, DIIM’04]
- [PTW’08, PTW’10]
Near-linear Space for $\{0,1\}^d$

[Indyk’01, Panigrahy’06]

Sample a few buckets in the same hash table!

• 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 $g(w')$ and compute distance to all points there
    • If found an approximate near neighbor, stop
Near-linear Space

- **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: $||q - p^*|| \leq r$
  - $w = g(q), \quad t = ||w - g(p^*)||_1$
  - 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><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>$\rho \approx \frac{1}{2c - 1}$</td>
<td>$\rho = 1/3$</td>
<td>[AINR’14, AR’15]</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></td>
<td></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
Construction of hash function

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

• Warning: hot off the press!

• Two components:
  – Nice geometric structure
  – Reduction to such structure

has better LSH
data-dependent
Nice geometric structure

- Like a random dataset on a sphere
  - s.t. random points at distance $\approx cr$
- Query:
  - At angle 45’ from near-neighbor

[Pictures curtesy of Ilya Razenshteyn]
Alg 1: Hyperplanes

[Charikar’02]

- Sample unit $r$ uniformly, hash $\rho$ into $\text{sgn}\langle r, \rho \rangle$
  - $\Pr[h(\rho) = h(q)] = 1 - \frac{\alpha}{\pi}$,
  - where $\alpha$ is the angle between $\rho$ and $q$

- $P_1 = \frac{3}{4}$
- $P_2 = \frac{1}{2}$
- $\rho \approx 0.42$
Alg 2: Voronoi

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

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

- 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
NNS: conclusion

1. Via sketches

2. Locality Sensitive Hashing
   - Random space partitions
   - Better space bound
     • Even near-linear!
   - Data-dependent hashing even better
     • Used in practice a lot these days
• The following was not presented in the lecture
Reduction to nice structure (HL)

- **Idea:** iteratively decrease the radius of minimum enclosing ball

- **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)
Dense clusters

- Current dataset: radius $R$
- A dense cluster:
  - Contains $n^{1-\delta}$ points
  - Smaller radius: $(1 - \Omega(\varepsilon^2))R$
- After we remove all clusters:
  - For any point on the surface, there are at most $n^{1-\delta}$ points within distance $(\sqrt{2} - \varepsilon)R$
  - 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 \((n^\delta + 1)^{O(1/e^2)}\)
  - Each time we branch
    - at most \(n^\delta\) clusters (+1)
    - a cluster reduces radius by \(\Omega(\varepsilon^2)\)
    - cluster-depth at most \(100/\Omega(\varepsilon^2)\)
- Progress in 2 ways:
  - Clusters reduce radius
  - CapCarving nodes reduce the # of far points (empirically)
- A tree succeeds with probability \(\geq n^{1/(2c^2-1)-o(1)}\)
Fast preprocessing

• How to find the dense clusters fast?

• Step 1: reduce to $O(n^2)$ time.
  – Enough to consider centers that are data points

• Step 2: reduce to near-linear time.
  – Down-sample!
  – Ok because we want clusters of size $n^{1-\delta}$
  – After downsampling by a factor of $\sqrt{n}$, a cluster is still somewhat heavy.
• In the analysis,
  – Instead of working with “probability of collision with far point” \( P_2 \), work with “empirical estimate” (the actual number)
  – A little delicate: interplay with “probability of collision with close point”, \( P_1 \)
    • The empirical \( P_2 \) important only for the bucket where the query falls into
    • Need to condition on collision with close point in the above empirical estimate
  – In dense clusters, points may appear *inside* the balls
    • whereas VoronoiLSH works for points on the sphere
    • need to partition balls into thin shells (introduces more branching)