Lecture 11: Nearest Neighbor Search
Plan

• Distinguished Lecture
  – Quantum Computing
  – Oct 19, 11:30am, Davis Aud in CEPSR

• Nearest Neighbor Search

• Scriber?
Sketching

• $W: \mathbb{R}^d \rightarrow$ short bit-strings
  – given $W(x)$ and $W(y)$, can distinguish between:
    • Close: $||x - y|| \leq r$
    • Far: $||x - y|| > cr$
  – With high success probability: only $\delta = 1/n^3$ failure prob.

• $\ell_2$, $\ell_1$ norm: $O(\epsilon^{-2} \cdot \log n)$ bits

\[ x \quad W \quad 010110 \quad W \quad y \quad \text{Is } ||W(x) - W(y)|| \leq t? \]

Yes: close, $||x - y|| \leq r$
No: far, $||x - y|| > (1 + \epsilon)r$
NNS: approaches

- Sketch $W$: uses $k = O(\epsilon^{-2} \cdot \log n)$ bits
- 1: Linear scan++
  - Precompute $W(p)$ for $p \in D$
  - Given $q$, compute $W(q)$
  - For each $p \in D$, estimate distance using $W(q), W(p)$
- 2: Exhaustive storage++
  - For each possible $\sigma \in \{0,1\}^k$
    - compute $A[\sigma] =$ point $p \in D$ s.t. $||W(p) - \sigma||_1 < t$
  - On query $q$, output $A[W(q)]$
  - Space: $2^k = n^{O(1/\epsilon^2)}$

Near-linear space and sub-linear query time?
Locality Sensitive Hashing

[Indyk-Motwani’98]

Random hash function $h$ on $\mathbb{R}^d$ satisfying:

- for close pair (when $\|q - p\| \leq r$)
  
  \[ P_1 = \Pr[h(q) = h(p)] \text{ is “not-so-small”} \]

- for far pair (when $\|q - p’\| > cr$)
  
  \[ P_2 = \Pr[h(q) = h(p’)] \text{ is “small”} \]

Use several hash tables

\[ n^\rho, \text{ where } \rho = \frac{\log 1/P_1}{\log 1/P_2} \]
LSH for Hamming space

- Hash function $g$ is usually a concatenation of “primitive” functions:
  
  - $g(p) = \langle h_1(p), h_2(p), \ldots, h_k(p) \rangle$

- **Fact 1:** $\rho_g = \rho_h$

- **Example:** Hamming space $\{0,1\}^d$
  
  - $h(p) = p_j$, i.e., choose $j^{th}$ bit for a random $j$
  
  - $g(p)$ chooses $k$ bits at random

  - $\Pr[h(p) = h(q)] = 1 - \frac{\text{Ham}(p,q)}{d}$

  - $P_1 = 1 - \frac{r}{d} \approx e^{-r/d}$

  - $P_2 = 1 - \frac{cr}{d} \approx e^{-cr/d}$

  - $\rho = \frac{\log 1/P_1}{\log 1/P_2} = \frac{r/d}{cr/d} = \frac{1}{c}$
Full Algorithm

• **Data structure** is just $L = n^\rho$ hash tables:
  – Each hash table uses a fresh random function $g_i(p) = \langle h_{i,1}(p), ..., h_{i,k}(p) \rangle$
  – Hash all dataset points into the table

• **Query:**
  – Check for collisions in each of the hash tables
  – until we encounter a point within distance $cr$

• **Guarantees:**
  – Space: $O(nL) = O(n^{1+\rho})$, plus space to store points
  – Query time: $O(L \cdot (k + d)) = O(n^\rho \cdot d)$ (in expectation)
  – 50% probability of success.
Choice of parameters $k, L$?

- $L$ hash tables with $g(p) = \langle h_1(p), \ldots, h_k(p) \rangle$
  - Pr[collision of far pair] = $P_2^k = 1/n$
  - Pr[collision of close pair] = $P_1^k = (P_2^\rho)^k = 1/n^\rho$
    - Success probability for a hash table: $P_1^k$
    - $L = O(1/P_1^k)$ tables should suffice
- Runtime as a function of $P_1, P_2$?
  - $O\left(\frac{1}{P_1^k}(\text{timeToHash} + nP_2^k)\right)$
- Hence $L = O(n^\rho)$
Analysis: correctness

• Let $p^*$ be an $r$-near neighbor
  – If does not exists, algorithm can output anything
• Algorithm fails when:
  – near neighbor $p^*$ is not in the searched buckets $g_1(q), g_2(q), ..., g_L(q)$
• Probability of failure:
  – Probability $q, p^*$ do not collide in a hash table: $\leq 1 - P_1^k$
  – Probability they do not collide in $L$ hash tables at most

\[
(1 - P_1^k)^L = \left(1 - \frac{1}{n^\rho}\right)^{n^\rho} \leq 1/e
\]
Analysis: Runtime

- Runtime dominated by:
  - Hash function evaluation: $O(L \cdot k)$ time
  - Distance computations to points in buckets

- Distance computations:
  - Care only about far points, at distance > $cr$
  - In one hash table, we have
    - Probability a far point collides is at most $P^k_2 = 1/n$
    - Expected number of far points in a bucket: $n \cdot \frac{1}{n} = 1$
  - Over $L$ hash tables, expected number of far points is $L$

- Total: $O(Lk) + O(Ld) = O(n^\rho (\log n + d))$ in expectation
If want exact NNS, what is $c$?
- Can choose any parameters $L, k$
- Correct as long as $(1 - P_1^k)^L \leq 0.1$
- Performance:
  - trade-off between # tables and false positives
  - will depend on dataset “quality”
## LSH Algorithms

<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</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</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, DIIM’04]</td>
</tr>
<tr>
<td></td>
<td></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>
</tbody>
</table>

Table does not include:
- $O(nd)$ additive space
- $O(d \cdot \log n)$ factor in query time
• Hamming distance [IM’98]
  – \( h \): pick a random coordinate(s)
• \( \ell_1 \) (Manhattan) distance [AI’06]
  – \( h \): cell in a randomly shifted grid
• Jaccard distance between sets:
  – \( J(A, B) = \frac{A \cap B}{A \cup B} \)
  – \( h \): pick a random permutation \( \pi \) on the universe
    \[
    h(A) = \min_{a \in A} \pi(a)
    \]
    min-wise hashing [Bro’97]
Claim: \( \Pr[\text{collision}] = J(A, B) \)
LSH for Euclidean distance

[Datar-Immorlica-Indyk-Mirrokni’04]

• LSH function $h(p)$:
  – pick a random line $\ell$, and quantize
  – project point into $\ell$

  $h(p) = \left\lfloor \frac{p \cdot \ell}{w} + b \right\rfloor$
  • $\ell$ is a random Gaussian vector
  • $b$ random in $[0,1]$
  • $w$ is a parameter (e.g., 4)

• Claim: $\rho = 1/c$