Topic 12: Sketching and Nearest Neighbor Search

David Woodruff

Sketching

- Random linear projection $M: \mathbb{R}^n \rightarrow \mathbb{R}^k$ that preserves properties of any $v \in \mathbb{R}^n$ with high probability, where $k \ll n$

- Matrix $M$ doesn’t depend on $v$, e.g., $M$ is a random matrix (typically, we require the entries of $M$ be $O(\log n)$ bits)

Estimating the Norm of a Vector

- For a vector $x \in \mathbb{R}^n$, its (squared) Euclidean norm is $|x|^2 = \sum x_i^2$
- Want to output a number $Z$ for which $(1 - \epsilon)|x|^2 \leq Z \leq (1 + \epsilon)|x|^2$
- Choose a 2-universal independent hash function $h: [n] \rightarrow [k]$
- Choose a 4-universal independent hash function $\sigma: [n] \rightarrow \{-1, 1\}$

CountSketch

- CountSketch is a linear map $S: \mathbb{R}^n \rightarrow \mathbb{R}^k$
- A row $i$ of $S$ is a hash bucket, and $(Sx)_i$ is the value in the bucket
- Output $|Sx|^2$
- $E[|Sx|^2] = E[\sum_i \delta(h(i) = j)\sigma(i)x_i^2]$
  
  = $\sum_i x_i^2 E[\delta(h(i) = j)\sigma(i)\sigma(i')]$
  
  = $\sum_i x_i^2 E[\delta(h(i) = j)\delta(h(i') = j)]E[\sigma(i)\sigma(i')]$
  
  = $\frac{x_i^2}{k} = |k|^2$
Estimating the Norm from CountSketch

- In recitation, you will show \( \text{Var}[|Sx|^2] = O(|x|^4/k) \)
- By Chebyshev’s inequality,
  \[
  \Pr[|Sx|^2 - |x|^2 > \epsilon |x|^2] \leq \frac{\text{Var}[|Sx|^2]}{\epsilon^2 |x|^4} \leq \frac{1}{10} \text{ if } k = \Theta\left(\frac{1}{\epsilon^2}\right)
  \]
- If \( S \) has \( k = \Theta\left(\frac{1}{\epsilon^2}\right) \) rows, can estimate \( |x|^2 \) from \( Sx \) up to a \( (1 + \epsilon) \)-factor with probability at least 9/10

Problem: Nearest Neighbor Search (NNS)

- Preprocess: a set \( P \) of points
- Query: given a query point \( q \), report a point \( p^* \in P \) with the smallest distance to \( q \)
- Useful for clustering problems, and many other problems on large sets of multi-feature objects
- Applications:
  - speech/image/video/music recognition, signal processing, bioinformatics, etc...

Measuring similarity between objects

00000
001100
000100
000100
110100
111111

objects \( \Rightarrow \) high-dimensional vectors
similarity \( \Rightarrow \) distance b/w vectors

\( \{0,1\}^d \)

Hamming distance

Preamble: How to check for an exact match ?

<table>
<thead>
<tr>
<th>Preprocess:</th>
<th>Sort the points</th>
</tr>
</thead>
<tbody>
<tr>
<td>Query:</td>
<td>Perform binary search</td>
</tr>
<tr>
<td>Query time</td>
<td>( O(d \log n) )</td>
</tr>
<tr>
<td>Space</td>
<td>( O(nd) )</td>
</tr>
</tbody>
</table>
Nearest Neighbor Search (NNS)

- **Preprocess**: a set $P$ of points
- **Query**: given a query point $q$, report a point $p \in P$ with the smallest distance to $q$

Approximate NNS

- **r-near neighbor problem**: given a new point $q$, report a point $p \in D$ s.t. $d(p,q) \leq r$
- **Randomized**: a point $p$ returned with 90% probability

Sketching

- $W: \mathbb{R}^d \rightarrow$ short bit-strings
  - given $W(x)$ and $W(y)$, can distinguish between:
    - **Close**: $d(x,y) \leq r$
    - **Far**: $d(x,y) > cr$
  - With high success probability: only $\delta = 1/n^3$ failure prob.
- Hamming distance of bitstrings: $O(\epsilon^{-2} \cdot \log n)$ bits

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 $A[\sigma] = \text{point } p \in D \text{ such that } d(W(p), \sigma) < t$
    - 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

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

satisfying:

for close pair (when $d(q,p) \leq r$)

$$P_1 = \Pr[h(q) = h(p)] \text{ is "not-so-small"}$$

for far pair (when $d(q,p) > c \cdot r$)

$$P_2 = \Pr[h(q) = h(p)] \text{ is "small"}$$

Use several hash tables

$n^\rho$, where $\rho = \frac{\log 1/P_1}{\log 1/P_2}$

Full Algorithm

- Data structure is just $L = n^\rho$ hash tables:
  - Each hash table uses a fresh random function $g_j(p) = \{h_{1j}(p), \ldots, h_{kj}(p)\}$
  - Hash all dataset points into the table

- Query:
  - Check for collisions in each of the hash tables
  - until we encounter a point within distance $c \cdot r$

- Guarantees:
  - Space: $O(nL\log n) = O(n^{1+\rho} \log n)$ bits, plus space to store original points
  - Expected Query time: $O(L \cdot (k + d)) = O(n^\rho \cdot d)$
  - 50% probability of success

LSH for Hamming space

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

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

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

- $P_2 = 1 - \frac{c \cdot r}{d} \approx e^{-c/d}$

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

Choice of parameters $k, L$?

- $L$ hash tables with $g(p) = \{h_1(p), \ldots, h_k(p)\}$

- $\Pr[\text{collision of far pair}] = P_2^k = 1/n^k$
  - $\Pr[\text{collision of close pair}] = P_1 = (P_1^k)^k = 1/n^\rho k$
  - Success probability for a hash table: $P_1^k$
  - $L = O(1/P_1^k)$ tables should suffice

- Runtime as a function of $P_1, P_2$?
  - $O(\frac{1}{P_1^k} \text{timeToHash} + nP_2^kd)$
  - Hence $L = O(n^\rho)$
Analysis: correctness

- Let \( p^* \) be an \( r \)-near neighbor
  - If does not exist, algorithm can output anything
- Algorithm fails when:
  - near neighbor \( p^* \) is not in the searched buckets \( b_1(q), b_2(q), \ldots, b_k(q) \)
- 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
    \[
    (1 - P_1^k)^L = \left(1 - \frac{1}{n^r}\right)^n \leq \frac{1}{e}
    \]

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 \( P_1^k = 1/n \)
    - 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: \( O(Lk) + O(Ld) = O(n^p d) \) in expectation

Find pairs of similar images

- Naively: about \( n^2 \) comparisons
- Can we do better?

Notes on NNS

<table>
<thead>
<tr>
<th>Space</th>
<th>Time</th>
<th>Exponent</th>
<th>( c = 2 )</th>
<th>Reference</th>
</tr>
</thead>
<tbody>
<tr>
<td>( m^p )</td>
<td>( n^p )</td>
<td>( p = 1/c )</td>
<td>( p = 1/2 )</td>
<td>[IM98, DIMP04]</td>
</tr>
<tr>
<td>( m^p )</td>
<td>( 1/c^2 )</td>
<td>( p = 1/4 )</td>
<td>[AF06]</td>
<td></td>
</tr>
</tbody>
</table>

NO: any map must satisfy \( \rho \geq 1/c^2 \)
[Motwani-Naor-Panigrahy’06, O’Donell-Wu-Zhou’11]