Lecture 24: Sketching and Nearest Neighbor Search

David Woodruff
Carnegie Mellon University

Slides mostly from Alex Andoni

1

Sketching

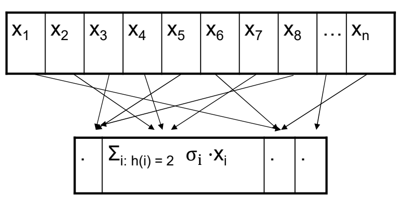
- Random linear projection M: $R^n \rightarrow R^k$ that preserves properties of any $v \in R^n$ with high probability ,where $k \ll n$

$$\begin{pmatrix} & M & \end{pmatrix}\begin{pmatrix} \\ v \\ \\ \end{pmatrix} = \begin{pmatrix} Mv \end{pmatrix} \longrightarrow \text{answer}$$

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



CountSketch

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

$$= \sum_j \sum_{i,i'} x_i x_{i'} E[\delta(h(i)=j)\delta(h(i')=j)\sigma(i)\sigma(i')]$$

$$= \sum_j \sum_{i,i'} x_i x_{i'}\ E[\delta(h(i)=j)\delta(h(i')=j)]E[\sigma(i)\sigma(i')]$$

$$= \frac{\sum_j \sum_i x_i^2}{k} = |x|^2$$

Estimating the Norm from CountSketch

- In recitation, you will show $\mathrm{Var}[|Sx|^2] = O(|x|^4/k)$
- By Chebyshev's inequality,

$$\Pr\left[\left||Sx|^2 - |x|^2\right| > \epsilon|x|^2\right] \le \frac{\mathrm{Var}[|Sx|^2]}{\epsilon^2|x|^4} \le \frac{1}{10} \text{ if } k = \Theta\left(\frac{1}{\epsilon^2}\right)$$

- If S has $k = \Theta(\frac{1}{\epsilon^2})$ rows, can estimate $|x|^2$ from Sx up to a $(1+\epsilon)$-factor with probability at least 9/10

Measuring similarity between objects

000000
001100
000100
000100
110100
111111

objects ⇒ high-dimensional vectors

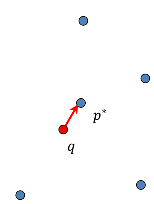
similarity ⇒ distance b/w vectors

$\{0,1\}^d$

Hamming distance

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…

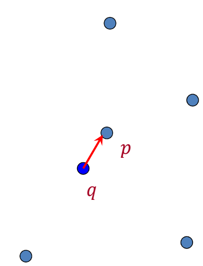


n: number of points
d: dimension

7

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



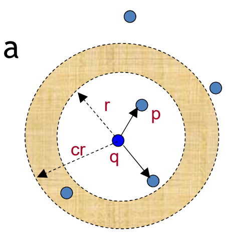
Approximate NNS

c-approximate

- r-near neighbor problem: given a new point q, report a point $p \in D$ s.t. d(p,q) $\leq$ $\boxed{cr}$

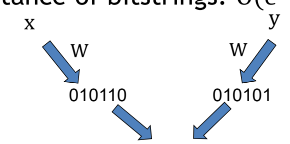
  if there exists a point at distance $\leq r$

- Randomized: a point p returned with 90% probability



Sketching

- $W: \Re^d \to$ 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



Is $d(W(x), W(y)) \leq t$?

Yes: close, $d(x,y) \leq r$

No: far, $d(x,y) > (1+\epsilon)r$

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 $p \in D$ 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 R^d satisfying:

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

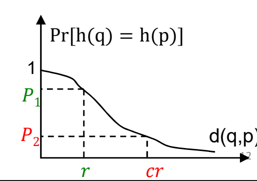
$P_1 =$ $\Pr[h(q) = h(p)]$ is "not-so-small"

for far pair (when d(q,p') $> cr$)

$P_2 =$ $\Pr[h(q) = h(p')]$ is "small"

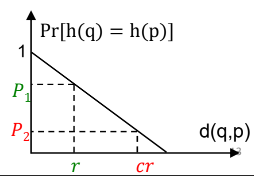
Use several hash tables

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



LSH for Hamming space

- Hash function g is usually a concatenation of "primitive" functions:
  - $g(p) = \langle h_1(p), h_2(p), \dots, h_k(p) \rangle$
- 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
  - $\Pr[h(p) = h(q)] = 1 - \frac{Ham(p,q)}{d}$
  - $P_1 = 1 - \frac{r}{d} \approx e^{-r/d}$
  - $P_2 = 1 - \frac{cr}{d} \approx e^{-cr/d}$
  - $\rho = \frac{\log 1/P_1}{\log 1/P_2} = \frac{r/d}{cr/d} = \frac{1}{c}$



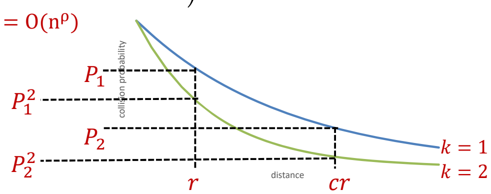
Full Algorithm

- **Data structure** is just $L = n^\rho$ hash tables:
  - Each hash table uses a fresh random function $g_i(p) = \langle h_{i,1}(p), \dots, h_{i,k}(p) \rangle$
  - 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: $O(nL\log n) = O(n^{1+\rho}\log n)$ bits, plus space to store original points
  - Expected Query time: $O(L \cdot (k + d)) = O(n^\rho \cdot d)$
  - 50% probability of success

14

Choice of parameters k, L ?

- L hash tables with $g(p) = \langle h_1(p), \dots, h_k(p) \rangle$
- Pr[collision of far pair] $= P_2^k$ $\boxed{\text{set } k \text{ s.t. } = 1/n}$
- 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 ?
  - $O\left(\frac{1}{P_1^k}\left(timeToHash + nP_2^k d\right)\right)$
- Hence $L = O(n^\rho)$



15

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 $g_1(q), g_2(q), \dots, g_L(q)$
- 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

$$(1 - P_1^k)^L = \left(1 - \frac{1}{n^\rho}\right)^{n^\rho} \leq 1/e$$

16

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 $P_2^k = 1/n$
    - 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: $O(Lk) + O(Ld) = O(n^\rho d)$ in expectation

17

Find pairs of similar images

18