COMS E6998-9: Algorithmic Techniques for Massive Data — Oct 15, 2015

# Lecture 12 – More LSH, Data-Dependent Hashing

Instructor: *Alex Andoni* — Scribes: *Mingxian Zhong*

## 1 Time-space Trade-offs

Below we present a table of LSH algorithms using different space and time.

| Type | Space | Time | Comment | Ref |
|---|---|---|---|---|
| Low space and High query time | $\approx n$ | $n^{\sigma}$ | $\sigma = 2.09/c$ | [Ind'01,Pan'06] |
| | $\approx n$ | $n^{\sigma}$ | $\sigma = O(1/c^2)$ | [AI'06] |
| Medium space and Medium query time | $n^{1+\rho}$ | $n^{\rho}$ | $\rho = 1/c$ | [IM'98,DIIM'04] |
| | $n^{1+\rho}$ | $n^{\rho}$ | $\rho = 1/c^2$ | [AI'06] |
| | $n^{1+\rho}$ | $n^{\rho}$ | $\rho \geq 1/c^2$ | [MNP'06,OWZ'11] |
| | $n^{1+o(1+1/c^2)}$ | $\omega(1)$ memory lookup | | [PTW'08,PTW'10] |
| High space and Low query time | $n^{4/\epsilon^2}$ | $O(dlogn)$(1 mem lookup) | $c = 1 + \epsilon$ | [KOR'98,IM'08,Pan'06] |
| | $n^{o(1/\epsilon^2)}$ | $\omega(1)$ memory lookup | | [AIP'06] |

## 2 Near-linear Space for $\{0,1\}^d$

[Indyk'01,PanIgrahy'06]

- General idea: Sample a few bucket in the same hash table.
- Setting:
  - Close: $r = \dfrac{d}{2c}$ [Note that from last lecture $P_1 = 1 - \dfrac{r}{d} = 1 - \dfrac{1}{2c}$]
  - Far: $cr = \dfrac{d}{2}$ [Note that from last lecture $P_2 = 1 - \dfrac{cr}{d} = \dfrac{1}{2}$]
- Algorithm:
  - Use on hash table with $k = \dfrac{\log n}{\log 1/P_2} = \alpha \ln n$
    [Note that since $P_2 = 1/2$ here $\alpha$ is a constant]
  - On query q:
    - Compute $w = g(p) \in \{0,1\}^k$
    - Define $w'$ such that starting from $w$, flip each $w_j$ with probability $1 - P_1$
    - Lookup bucket $g(w')$ and compute distance to all points there
    - Repeat $R = n^{\sigma}$ times, stop if found an approximate near neighbor

**Theorem.** *For* $\sigma = \Theta(\frac{\log c}{c})$*, we have*

- *Pr[find an approximate near neighbor]*$\geq 0.1$
- *Expected runtime:* $O(n^\sigma)$

*Proof.* Let $p^*$ be the near neighbor, then we know that $\|q - p^*\| \leq r$. Define $w = g(q), t = \|w - g(p^*)\|_1$.

**Claim 1.** $Pr[t \leq \frac{k}{c}] \geq \frac{1}{2}$

*Proof.* Note that $E[t] = \frac{r}{d}k = \frac{k}{2c}$.

Hence by Markov Inequality, $Pr[t \leq \frac{k}{c}] \geq 1 - \frac{k}{2c}/\frac{k}{c} = \frac{1}{2}$ □

**Claim 2.** $Pr[w' = g(p)|\|q - p\|_1 \geq \frac{d}{2}] \leq \frac{1}{n}$

*Proof.*

$$\begin{aligned} Pr[\text{Collision}] &\leq (P_1P_2 + (1-P_1)(1-P_2))^k \\ &= (P_2(P_1 + 1 - 1) + (1-P_1)(1-P_2))^k \\ &= (P_2 + (1-P_1)(1-2P_2))^k \\ &\leq P_2^k = 1/n \end{aligned}$$

□

**Claim 3.** $Pr[w' = g(p^*)|\textit{Claim 1}] \geq n^{-\sigma}$

*Proof.*

$$\begin{aligned} Pr[w' = g(p^*)|\text{Claim 1}] &= (1-P_1)^t P_1^{k-t} \\ &\geq (1 - (1 - \frac{1}{2c}))^{k/c}(1 - \frac{1}{2c})^{k(1-1/c)} \\ &\geq (\frac{1}{2c})\frac{k}{c}e^{-\frac{1}{2c}k} \\ &\geq n^{-\frac{\Theta(1)\lg c}{c}} n^{-\frac{\alpha}{2c}} \\ &\geq n^{-\sigma} \end{aligned}$$

□

Since if $w' = g(p^*)$ for at least one $w'$, we are guaranteed to output either $p^*$ or an approximate near neighbor, we are done by Claim 3. □

# 3 Beyond LSH

Below we give a contrast of LSH algorithms and other algorithm.

In Hamming Space

| Type | Space | Time | Comment | $c = 2$ | Reference |
|---|---|---|---|---|---|
| LSH | $n^{1+\rho}$ | $n^{\rho}$ | $\rho = 1/c$ | $\rho = 1/2$ | [IM'98] |
| | | | $\rho \geq 1/c$ | | [MNP'06,OWZ'11] |
| Non-LSH | $n^{1+\rho}$ | $n^{\rho}$ | $\rho \approx \frac{1}{2c-1}$ | $\rho = 1/3$ | [AINR'14,AR'15] |

In Euclidean Space

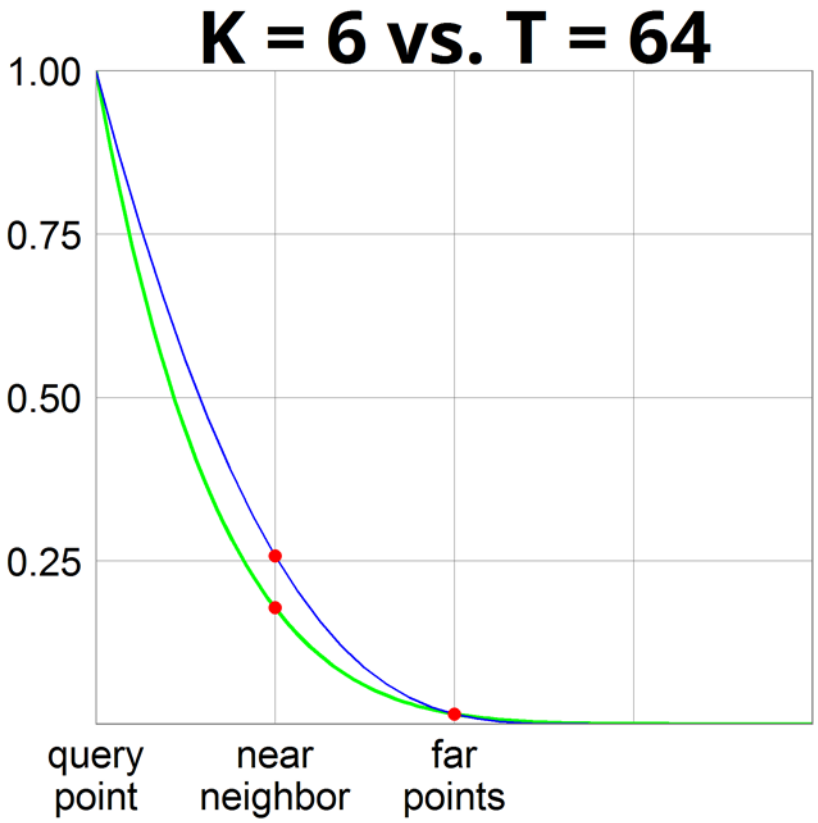
| Type | Space | Time | Comment | $c = 2$ | Reference |
|---|---|---|---|---|---|
| LSH | $n^{1+\rho}$ | $n^{\rho}$ | $\rho \approx 1/c^2$ | $\rho = 1/4$ | [AI'06] |
| | | | $\rho \geq 1/c^2$ | | [MNP'06,OWZ'11] |
| Non-LSH | $n^{1+\rho}$ | $n^{\rho}$ | $\rho \approx \frac{1}{2c^2-1}$ | $\rho = 1/7$ | [AINR'14,AR'15] |

# 4 Data-dependent hashing

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

- General idea: Using a random has function, which is chosen after seeing the given dataset
- Feature: Efficiently computable
- Components:
  - Nice geometric structure (has better data partition)
  - Reduction to such structure (depends on the data)
- 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
- Alg 1: Hyperplanes[Charikar'02]
  - We sample unit r uniformly, hash $p$ into $sgn < r, p >$,
    $Pr[h(p) = h(q)] = 1 - \alpha/\pi$, 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 $g_2, g_2, .., g_T$.
  - Hash $p$ into $h(p) = argmax_{1 \leq i \leq T} < p, g_i >$
  - Note that it is simply Hyperplane LSH when $T = 2$

- Hyperplane VS Voronoi
  - Hyperplane with $k = 6$ hyperplanes , which means we partition space into $2^6 = 64$ pieces
  - Voronoi with $T = 2^k = 64$ vectors. $\rho = 0.18$




  - In Hyperplane algorithm we partition into grids while in Voronoi we partition into sphere

## 5 Nearest Neighbor Search: Conclusion

- Approach 1: Via sketches
- Approach 2: Locality Sensitive Hashing
  - Use Random Space Partitions
  - Algorithm with Better Space Bound
  - Use Data-dependent hashing