Lecture 11:
Nearest Neighbor Search

COLUMBIA | ENGINEERING
The Fu Foundation School of Engineering and Applied Science

Plan

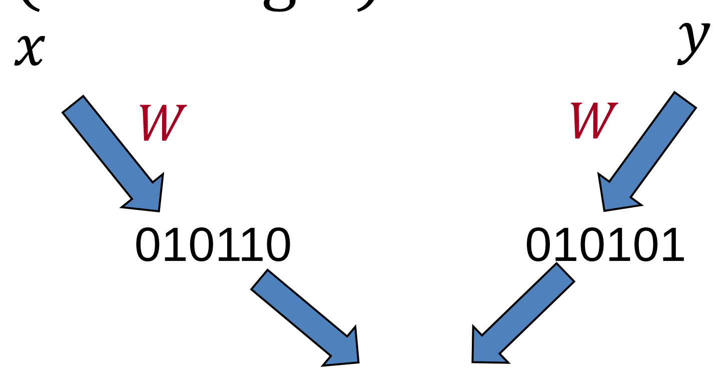
- Distinguished Lecture
  - Quantum Computing
  - Oct 19, 11:30am, Davis Aud in CEPSR

- Nearest Neighbor Search

- Scriber?

Sketching

- $W: \Re^d \rightarrow$ 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.
- ℓ_2, ℓ_1 norm: $O(\epsilon^{-2} \cdot \log n)$ bits



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]

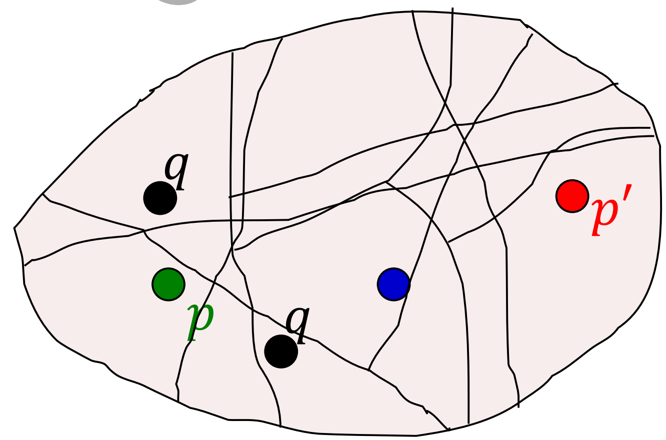
Random hash function h on R^d satisfying:

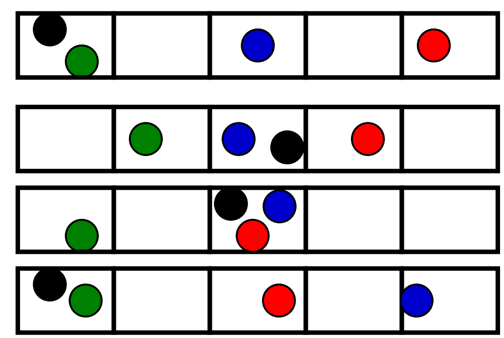
for *close pair* (when $\|q-p\| \le r$)

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

for *far pair* (when $\|q-p'\| > cr$)

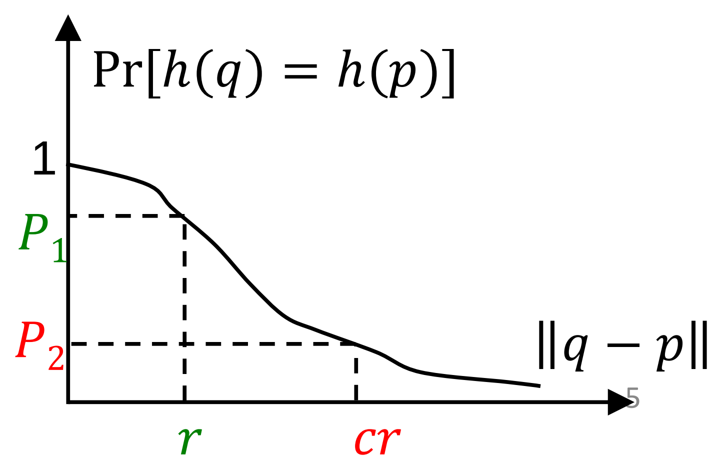
$P_2 =$ $\Pr[h(q)=h(p')]$ is "small"





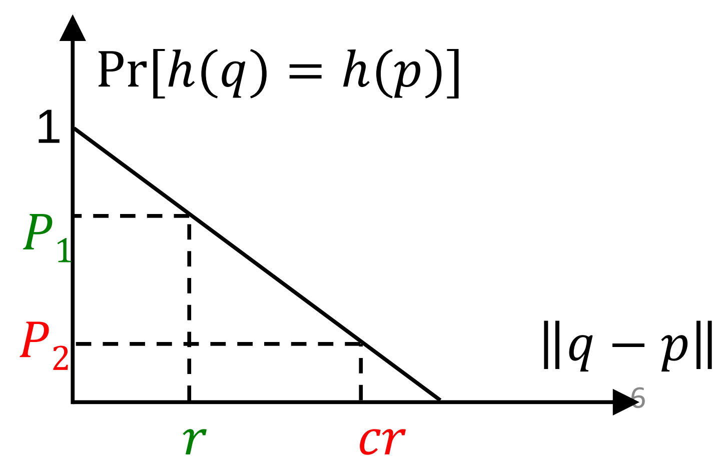
Use several hash tables

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:
  - $g(p) = \langle h_1(p), h_2(p), \ldots, h_k(p) \rangle$
- 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
  - $\Pr[h(p) = h(q)] = 1 - \frac{Ham(p,q)}{d}$
  - $P_1 = 1 - \frac{r}{d} \approx e^{-r/d}$
  - $P_2 = 1 - \frac{cr}{d} \approx e^{-cr/d}$
  - $\rho = \frac{\log 1/P_1}{\log 1/P_2} = \frac{r/d}{cr/d} = \frac{1}{c}$

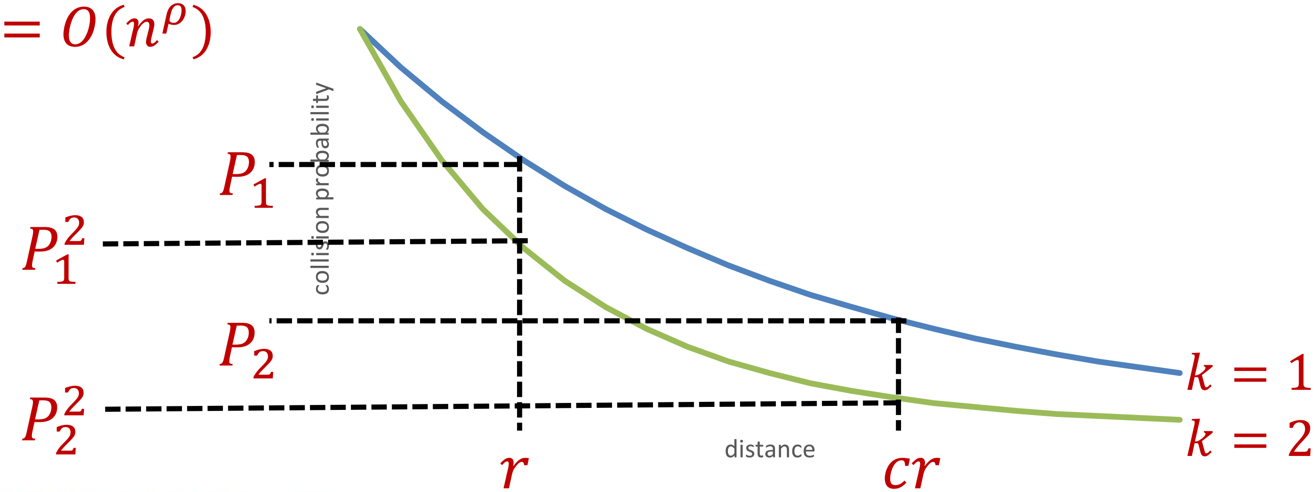


Full Algorithm

- **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: $O(nL) = O(n^{1+\rho})$, 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 set k 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 ?
  - $O\left(\frac{1}{P_1^k}\left(timeToHash + nP_2^k\right)\right)$
- Hence $L = O(n^\rho)$



Analysis: correctness

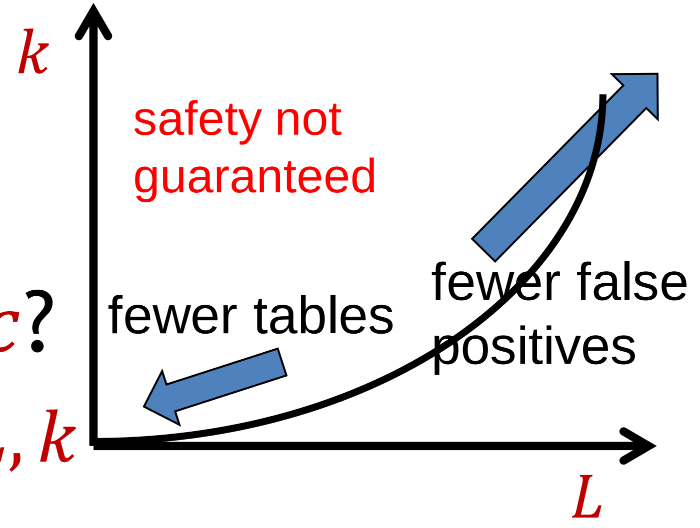
- Let p^* be an r-near neighbor
  - If does not exists, algorithm can output anything
- Algorithm fails when:
  - near neighbor p^* is not in the searched buckets $g_1(q), g_2(q), \ldots, g_L(q)$
- Probability of failure:
  - Probability q, p^* do not collide in a hash table: $\leq 1 - P_1^k$
  - Probability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

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

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

LSH in practice



- If want exact NNS, what is c?
  - Can choose any parameters L, k
  - Correct as long as $\left(1 - P_1^k\right)^L \leq 0.1$
  - Performance:
    - trade-off between # tables and false positives
    - will depend on dataset "quality"

LSH Algorithms

	Space	Time	Exponent	$c = 2$	Reference
Hamming space	$n^{1+\rho}$	n^{ρ}	$\rho = 1/c$	$\rho = 1/2$	[IM'98]
			$\rho \geq 1/c$		[MNP'06, OWZ'11]
Euclidean space	$n^{1+\rho}$	n^{ρ}	$\rho = 1/c$	$\rho = 1/2$	[IM'98, DIIM'04]
			$\rho \approx 1/c^2$	$\rho = 1/4$	[AI'06]
			$\rho \geq 1/c^2$		[MNP'06, OWZ'11]

Table does not include:

- $O(nd)$ additive space
- $O(d \cdot \log n)$ factor in query time

LSH Zoo (ℓ_1)

- Hamming distance [IM'98]
  - h: pick a random coordinate(s)
- ℓ_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 π on the universe

    $h(A) = \min_{a \in A} \pi(a)$

    min-wise hashing [Bro'97]

Claim: Pr[collision]=*J(A,B)*

To be or not to be

To sketch or not to sketch

be not or sketch to

…11101…

…01111…

…21102…

…01122…

{be,not,or,to}

{not,or,to, sketch}

be

to

π=be,to,sketch,or,not

LSH for Euclidean distance

[Datar-Immorlica-Indyk-Mirrokni'04]

- LSH function $h(p)$:
  - pick a random line ℓ, and quantize
  - project point into ℓ
  - $h(p) = \left\lfloor \frac{p \cdot \ell}{w} + b \right\rfloor$
    - ℓ is a random Gaussian vector
    - b random in $[0,1]$
    - w is a parameter (e.g., 4)
- **Claim:** $\rho = 1/c$