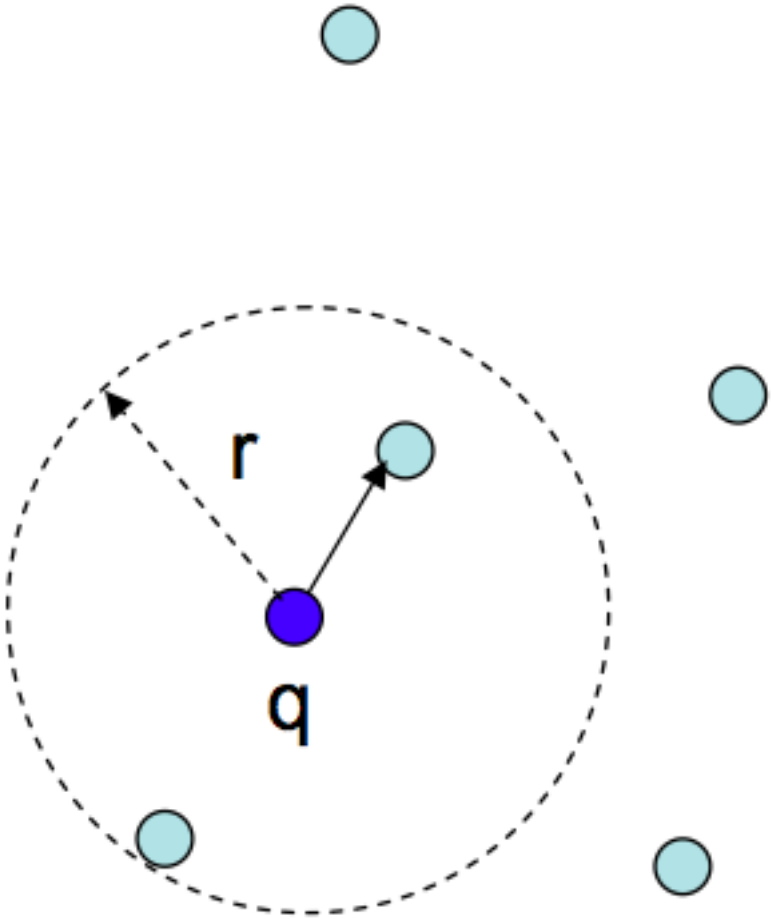
Locality Sensitive Hash in High Dimensional Nearest Neighbor Search

Yan Zheng
Database seminar, Spring 2012
School of Computing, University of Utah

Nearest Neighbor

- Input: D as a set of points in R^d
- Nearest Neighbor: for any query q, returns a point $p \in D$ minimizing $||p-q||$
- r-Near Neighbor: for any query q, returns a point $p \in D$ $||p-q|| \leq r$ (if it exists)

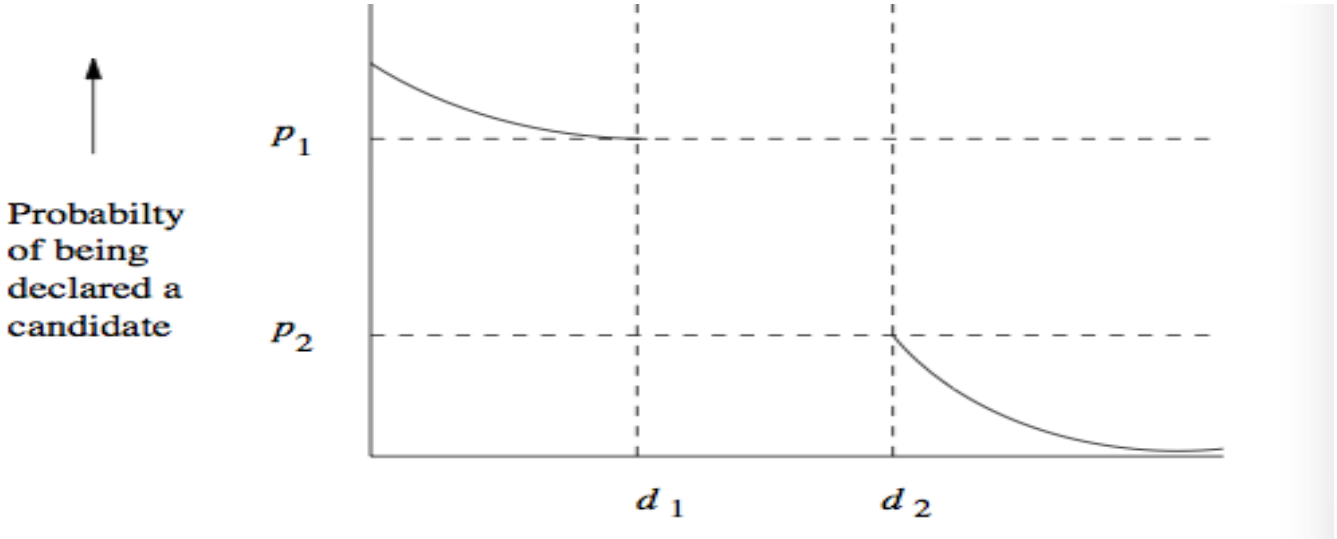


Locality-Sensitive Hashing

A family H is called (d_1, d_2, P_1, P_2)-sensitive if for any two points $p, q \in R^d \quad R^d \rightarrow U$

- If $\|p-q\| \leq d_1 \quad then \quad P_r[h(q)=h(p)] \geq P_1$
- If $\|p-q\| \geq d_2 \quad then \quad P_r[h(q)=h(p)] \leq P_2$

$P_1 > P_2$ and P_1 is high and P_2 is small

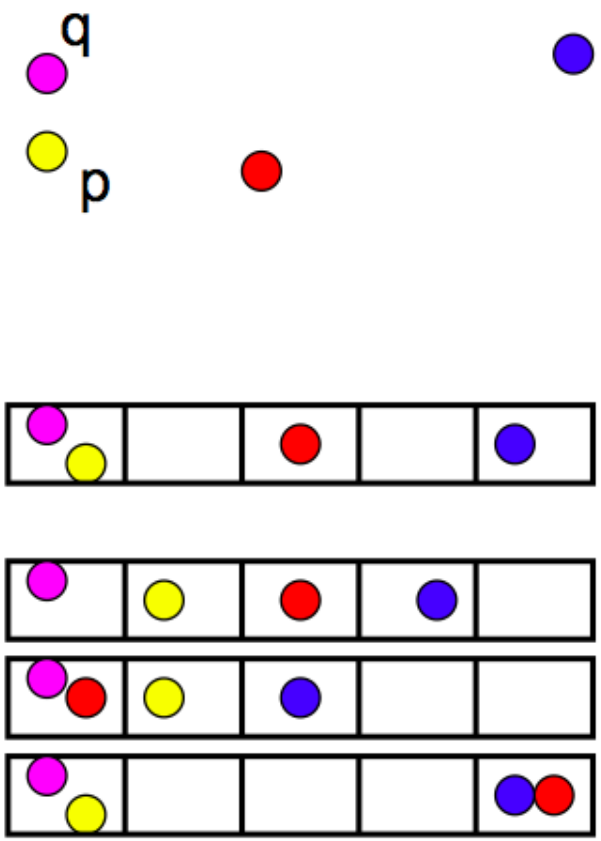


Example: Hamming distance

- LSH functions: $h(p)=p_i$, the value of the i-th coordinate of p

Probabilities: Pr[h(p)=h(q)] = 1-D(p,q)/d

- p=10010010
- q=11010110



LSH Algorithm based on Hamming Distance

- We use functions of the form

  $G(p)=<h_1(p),h_2(p),\ldots,h_m(p)>$
- Preprocessing:
  - Select $G_1(p),\ldots,G_L(p)$,
  - For all $p \in D$, hash p to buckets $G_1(p),\ldots,G_L(p)$
- Query:
  - Retrieve the points from buckets $G_1(q),\ldots,G_L(q)$ until
    - Either the points from all L buckets have been retrieved
    - Total number of points retrieved exceeds 2L

Projection-based LSH

- Define $h_{a,b}(p)=\lfloor(<p,a>+b)/w\rfloor$

– a=(a_1…a_d) , where a_i is chosen from:

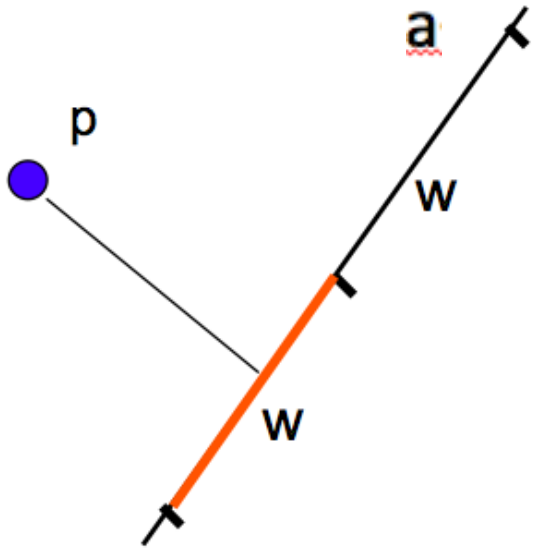
  - Gaussian distribution(for l_2 norm)
  - “s-stable” distribution* (for l_s norm)

– b is a real number chosen uniformly from the range [0,w]

u = $||p-q||_s$

$$p(u) = P_r[h(p) = h(q)] = \int_0^w \frac{1}{u} f_s(\frac{t}{u})(1 - \frac{t}{w})\,dt$$

P_1 = p(1) and P_2 = p(c) So it is a (R,cR,P_1,P_2)-sensitive



fs(t) is the probability density function of the absolute value of the s-stable distribution

LSB-TREE

- d-dimensional dataset D
- Each point o$\in$D $\rightarrow$ m-dimensioanl point G(o)
- Obtain the Z-order value z(o) of G(o)
- Index Z-order values with a conventional B-tree.
- The coordinates of o are stored along with its leaf entry.

LSB-TREE

- Hash function:

  $H_{a,b}(o)=<o,a>+b^*$

Value b* is uniformly distributed in $[0, 2^f w^2)$

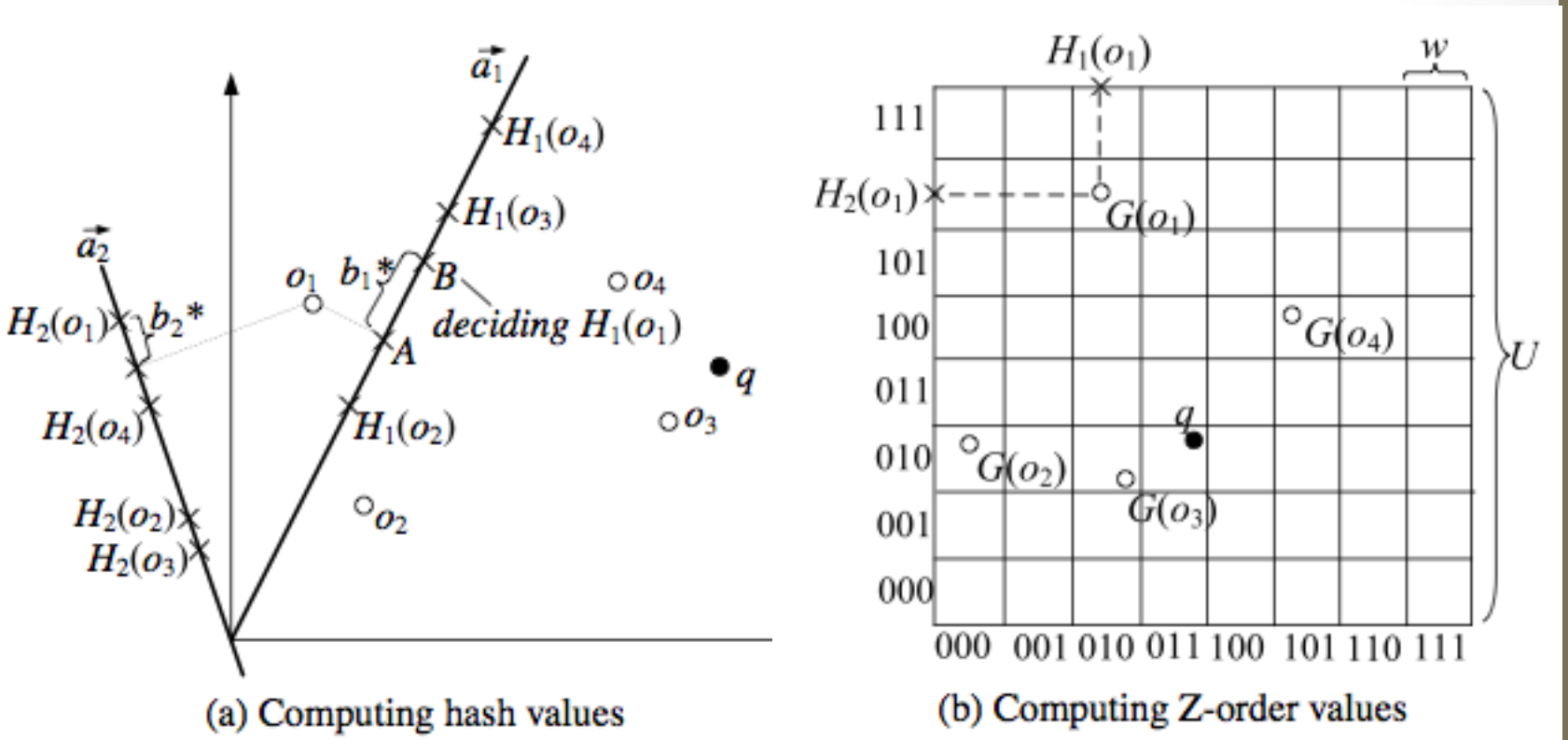
$$f = \lceil \log_2 d + \log_2 t \rceil$$

t is the largest coordinate in each dimension.

- $G(o)=<H_1(o),H_2(o),...,H_m(o)>$
- From G(o) to z(o). Let U be the axis length of the m-dimensional space G(o) falls in.

  u = log2(U/w). z(o) is a binary string with um bits.

Example



H1(o1) = 010 H2(o1)=110 z(o1)=011100

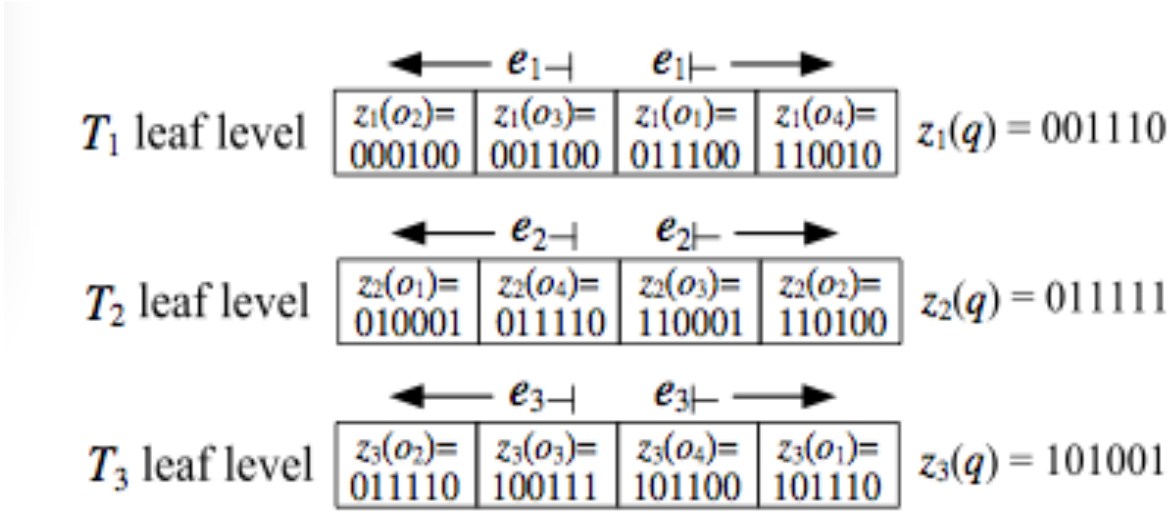
LSB-forest

- Denote the L trees as T_1, T_2, …, T_L, call them collectively as a LSB-forest.
- z_j (o) represent the Z-order value of o in tree T_j(j = 1, …, L)
- Given a NN query q, get z_j(q) in each tree
- LLCP(z_j(o),z_j(q)) is the length of the longest common prefix.
- z_j (o)=100101 z_j(q)=100001 then LLCP(z_j(o),z_j(q))=3

Algorithm NN

1. **repeat**
2. pick, from all the trees T_1, ..., T_l, the leaf entry with the next greatest LLCP
3. **until** condition E_1 or E_2 holds
4. return the nearest point found so far

- Finding the next greatest LLCP
  - $e_{j\vdash}$ with the lowest Z-order value at least $z_j(q)$
  - Let $e_{j\dashv}$ be the leaf entry immediately preceding $e_{j\vdash}$



  - the greatest LLCP must be in the set S = {$e_{1\vdash}$, $e_{1\dashv}$, ..., $e_{l\vdash}$, $e_{l\dashv}$}

- **Terminating condition.** Algorithm *NN* terminates when one of two events E1 and E2 happens.
- E1: the total number of leaf entries accessed from all l LSB-trees has reached 4Bl/d.
- E2:the nearest point found so far(from all the leaf entries already inspected) has distance to q at most 2^x.
- $x = 2^{u - \lfloor v/m \rfloor + 1}$

  u=3 m=2 $||o_3-q||=3$ $||o_4-q||=5$ x = 4

  $z_2(o_4)$ has the largest LLCP v=5
- $||o_4-q|| = 5 > 2^x=4$

  $||o_3-q|| = 3 < 2^x=4$
- $z_1(o_3)$ next largest LLCP v =4 E2 hold, NN terminates by return o_3

kNN search with a single tree

- LSB-trees: large space consumption and update overhead.
- Use single LSB-TREE to process kNN queries.
- E1 is ignored.
- E2 is changed to q is within distance 2^x to the k nearest points found so far.

Thanks!