1

Hashing for nearest neighbor search

- Hashing generally works for “equality searches”
- ..can we use “hashes” for nearest-neighbor searches???
- ….if they are locality sensitive, then “yes”!

2

Locality Sensitive Hashing (LSH)

- What is “locality sensitive hashing”?
  - …a “grid” is a locality sensitive hash
  - …a space filling curve is a locality sensitive hash

  More specifically, these are deterministic functions that tend to map nearby points to the same or nearby values.
- Can we develop randomized locality sensitive hashes?

3

Locality Sensitive Hashing (LSH)

- Let *sim()* be a similarity function
- A locality sensitive hash corresponding to *sim()* is a function, *h()*, such that

$$prob(h(o1) = h(o2)) = sim(o1,o2)$$

- The challenge is to find the appropriate *h()* for a given *sim()*

4

Locality Sensitive Hashing (LSH)

- An LSH family, H, is (r, cr, P_1, P_2)-sensitive, if for any two objects o_i and o_j and for a randomly selected $h() \in H$
  - if $dist(o_i, o_j) \le r$ then $prob\ (h(o_i) = h(o_j)) \ge P_1$,
  - if $dist(o_i, o_j) \ge cr$ then $prob\ (h(o_i) = h(o_j)) \le P_2$ and
  - $P_1 > P_2$.

5

Locality Sensitive Hashing (LSH)

- Consider a (r, cr, P_1, P_2)-sensitive hash family, H
- Let's create L composite hash functions

$$g_j(o) = (h_{1,j}(o), \ldots, h_{k,j}(o)),$$

by picking $L \times k$ hash functions, $h_{i,j} \in H$, independently and uniformly at random from H.

6

Locality Sensitive Hashing (LSH)

- Let us be given $g_1()$ through $g_L()$ and database, D,
- Hash object o in D using $g_1()$ through $g_L()$ and include o in all matching hash buckets

$$g_1(o) = (h_{1,1}(o), \ldots, h_{k,1}(o)),$$
$$\ldots.$$
$$g_L(o) = (h_{1,L}(o), \ldots, h_{k,L}(o))$$

7

Locality Sensitive Hashing (LSH)

- Hash the query q in also using $g_1()$ through $g_L()$ and consider all objects in these hash buckets

$$g_1(q) = (h_{1,1}(q), \ldots, h_{k,1}(q)),$$
$$\ldots.$$
$$g_L(q) = (h_{1,L}(q), \ldots, h_{k,L}(q))$$

- Key result:
  - if $L = \log_{1-P_1^k}\delta$, then any object within range r is returned with probability at least $1-\delta$.

8

Locality Sensitive Hashing (LSH)

- Then, how do we create a (r, cr, P_1, P_2)-sensitive hash family, H ??
- ….depends on the underlying *sim()* or $\delta()$ function…

9

Locality Sensitive Hashing (LSH)

- Assume d-dimensional binary vector; e.g. (0,1,1,1,0,…,1)
- Let δ() be the hamming distance (number of differing dimensions between two vectors)
- H contains all projections of the input point x on one of the coordinates; i.e., $h_i(x) = x_i$

10

Locality Sensitive Hashing (LSH)

- Let
  - p and q be two vectors in d-dimensional binary vector space
  - δ() is the hamming distance
  - H contains $h_i(x) = x_i$
- Note that *prob[h(q) = h(p)]* is equal to the fraction of coordinates on which *p* and *q* agree.
- Then, if we select
  - $P_1 = 1 - (r/d)$ and $P_2 = 1 - c(r/d)$
  - such that $c>1$
    we have $P_1 > P_2$.

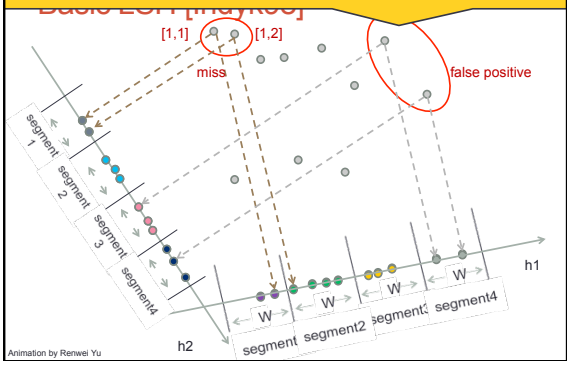
11

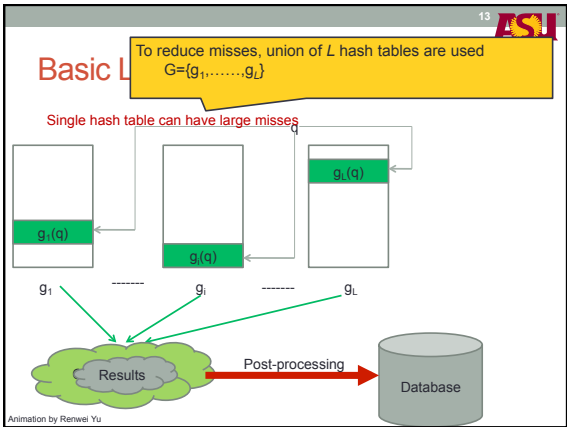
Locality Sensitive Hashing (LSH)

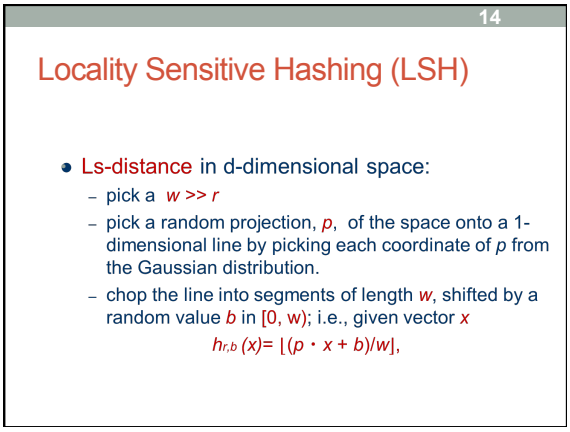
- L1-distance in d-dimensional space:
  - pick a $w >> r$
  - impose a randomly shifted grid with cells of width w
    - pick random $s, s_1,\ldots,s_d$ in [0, w)
    - define $h_{s1, s2,\ldots sd}(x)=(\lfloor(x_1 - s_1)/w\rfloor,\ldots,\lfloor(x_d - s_d)/w\rfloor)$.

To eliminate false positives, hash function of each table is the intersection of *multiple* element hashes h()

$g()=[h_1(),\ldots\ldots,h_c()]$ …but this can increase the number of misses!!!!!




13
Basic L
To reduce misses, union of L hash tables are used
G={g1,……,gL}
Single hash table can have large misses
q
g1(q)
gi(q)
gL(q)
g1
gi
gL
Results
Post-processing
Database
Animation by Renwei Yu


14
Locality Sensitive Hashing (LSH)
Ls-distance in d-dimensional space:
pick a w >> r
pick a random projection, p, of the space onto a 1-dimensional line by picking each coordinate of p from the Gaussian distribution.
chop the line into segments of length w, shifted by a random value b in [0, w); i.e., given vector x
hr,b (x)= ⌊(p • x + b)/w⌋,