Approximate Near Neighbor Search: Locality Sensitive Hashing

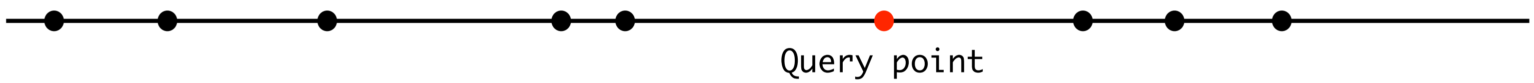
Inge Li Gørtz

Nearest Neighbor

- Nearest Neighbor. Given a set of points P in a metric space, build a data structure which given a query point x returns the point in P closest to x.

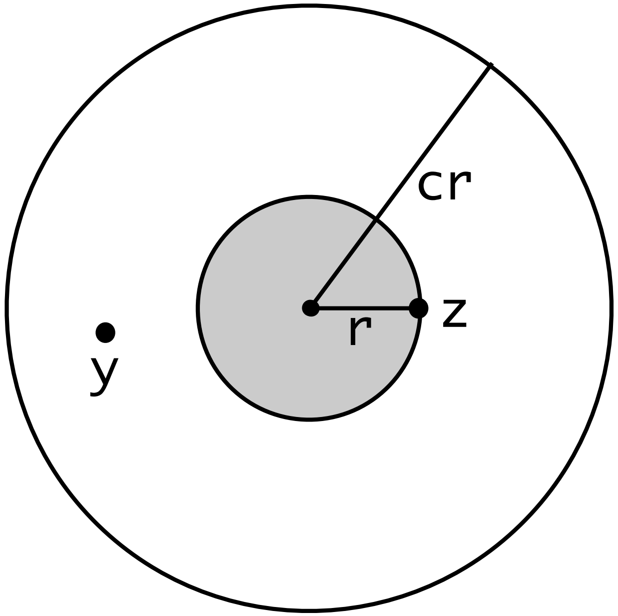
- Metric. Distance function d is a metric:
    1. $d(x,y) \geq 0$
    2. $d(x,y) = 0$ if and only if $x = y$
    3. $d(x,y) = d(y,x)$
    4. $d(x,y) \leq d(x,z) + d(z,y)$

- Warmup. 1D: Real line



Approximate Near Neighbors

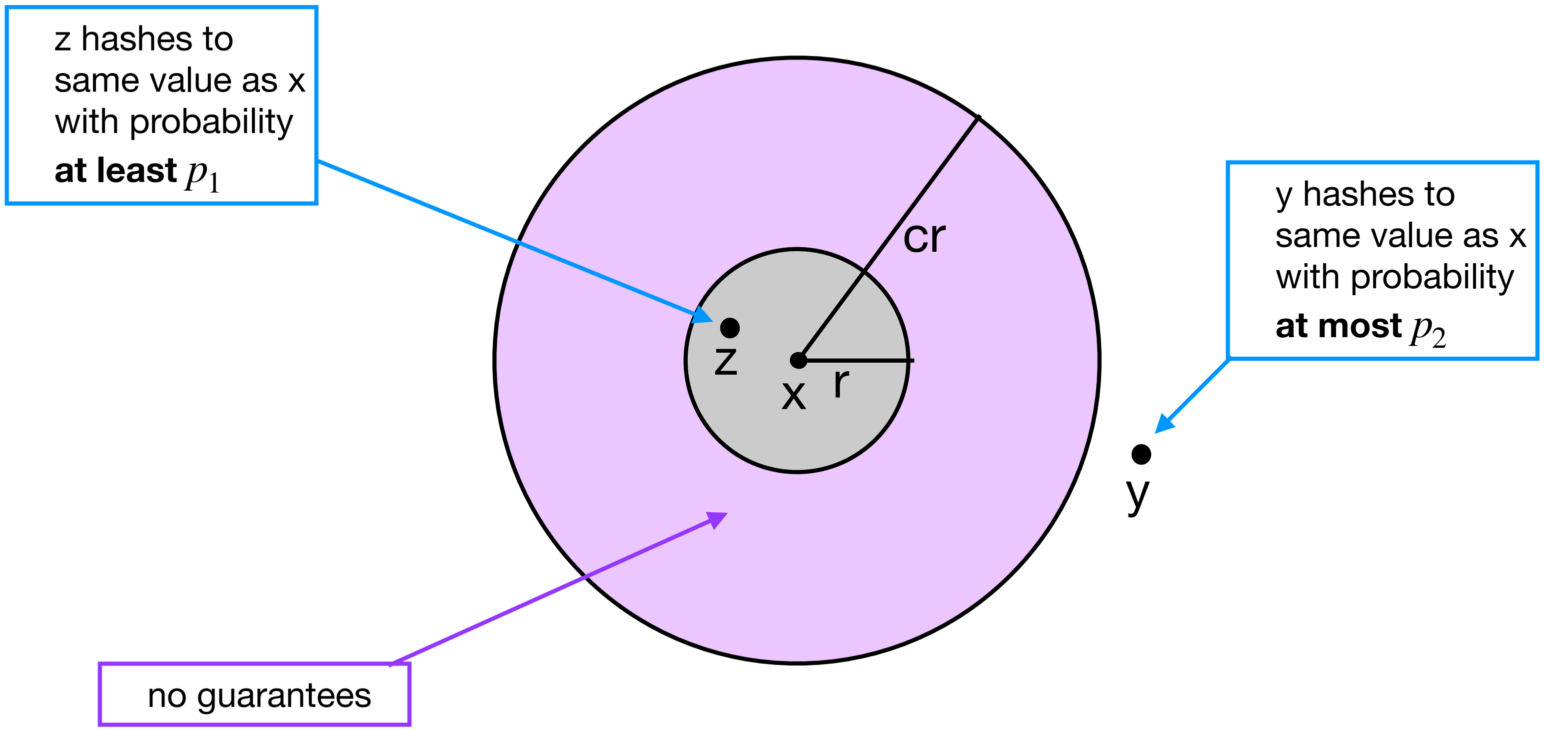
- ApproximateNearNeighbor(x): Return a point y such that $d(x, y) \leq c \cdot \min_{z \in P} d(x, z)$
- c-Approximate r-Near Neighbor: Given a point x if there exists a point z in P such that $d(x, z) \leq r$ then return a point y such that $d(x, y) \leq c \cdot r$. If no such point z exists return Fail.
- Randomised version: Return such an y with probability δ.



Locality Sensitive Hashing

- Locality sensitive hashing. A family of hash functions H is (r, cr, p_1, p_2)-sensitive with $p_1 > p_2$ and $c > 1$ if:
  - $d(x, y) \leq r \quad \Rightarrow \quad P[h(x) = h(y)] \geq p_1$ (close points)
  - $d(x, y) \geq cr \quad \Rightarrow \quad P[h(x) = h(y)] \leq p_2$ (distant points)

  for h chosen randomly from H.



Hamming Distance

- P set of n bit strings each of length d.
- Hamming distance. the number of bits where x and y differ:

$$d(x, y) = |\{i : x_i \neq y_i\}|$$

- Example.

$$\begin{array}{l}x = \boxed{1}\ \boxed{0}\ 1\ 0\ 0\ 1\ \boxed{0}\ 0 \\ y = \boxed{0}\ \boxed{1}\ 1\ 0\ 0\ 1\ \boxed{1}\ 0\end{array}$$

Hamming distance = 3

- Hash function. Chose $i \in \{1,\ldots,d\}$ uniformly at random and set $h(x) = x_i$.

- What is the probability that h(x) = h(y)?

  - $d(x, y) \leq r \Rightarrow P[h(x) = h(y)] \geq 1 - r/d$

  - $d(x, y) \geq cr \Rightarrow P[h(x) = h(y)] \leq 1 - cr/d$

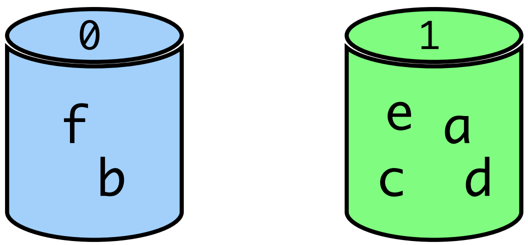
LSH with Hamming Distance: Solution 1

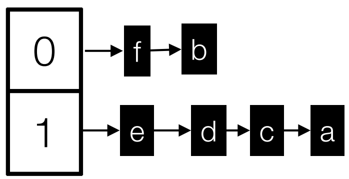
- Pick random index i uniformly at random. Let $h(x) = x_i$.
- Bucket: Strings with same hash value $h(x)$.
- Insert(x): Insert x in the list $A[h(x)]$
- NearNeighbour(x): Compute Hamming distance from x to all bitstrings in $A[h(x)]$ until find one that is at most cr away. If no such string found return FAIL.

$h(x) = x_3$

$a = 0011101$	$h(a) = 1$	$d = 0110011$	$h(d) = 1$
$b = 0101001$	$h(b) = 0$	$e = 1011101$	$h(e) = 1$
$c = 0010010$	$h(c) = 1$	$f = 1101101$	$h(f) = 0$

Query time: O(nd).





LSH with Hamming Distance: Solution 2

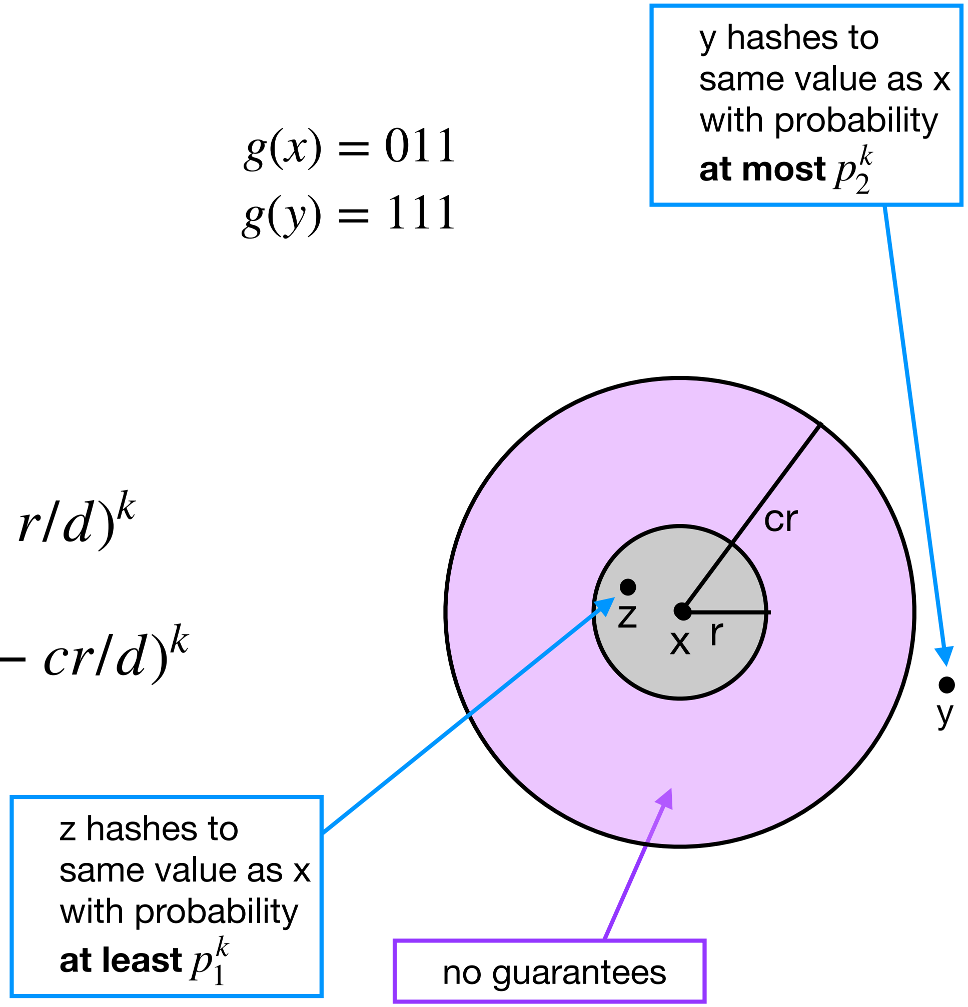
- Pick k random indexes uniformly and independently at random with replacement:
  - $g(x) = x_{i_1} x_{i_2} \cdots x_{i_k}$
- Example. $k = 3$. $g(x) = x_2 x_3 x_6$

x =	1	$\boxed{0}$	$\boxed{1}$	0	0	$\boxed{1}$	0	0
y =	0	$\boxed{1}$	$\boxed{1}$	0	0	$\boxed{1}$	1	0

$g(x) = 011$

$g(y) = 111$

- Probability that g(x) = g(y)?
  - $d(x, y) \leq r \Rightarrow P[g(x) = g(y)] \geq (1 - r/d)^k$
  - $d(x, y) \geq cr \Rightarrow P[g(x) = g(y)] \leq (1 - cr/d)^k$



LSH with Hamming Distance: Solution 2

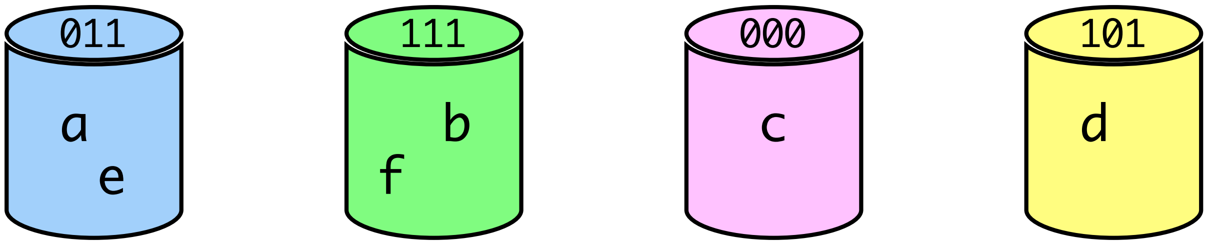
- Pick k random indexes uniformly and independently at random with replacement:
  - $g(x) = x_{i_1} x_{i_2} \cdots x_{i_k}$
- Bucket: Strings with same hash value $g(x)$.

$g(x) = x_2 x_4 x_7$

$a = 0011101$ $g(a) = 011$ $d = 0110011$ $g(d) = 101$

$b = 0101001$ $g(b) = 111$ $e = 1011101$ $g(e) = 011$

$c = 0010010$ $g(c) = 000$ $f = 1101101$ $g(f) = 111$



LSH with Hamming Distance: Solution 2

- Pick k random indexes uniformly and independently at random with replacement:
  - $g(x) = x_{i_1} x_{i_2} \cdots x_{i_k}$
- Bucket: Strings with same hash value $g(x)$.
- Save buckets in a hash table T with hash function h_T.

$h_T(011_2) = 1$

$h_T(111_2) = 6$

$h_T(000_2) = 9$

$h_T(101_2) = 1$

$g(x) = x_2 x_4 x_7$

$a = 0011101 \quad g(a) = 011$

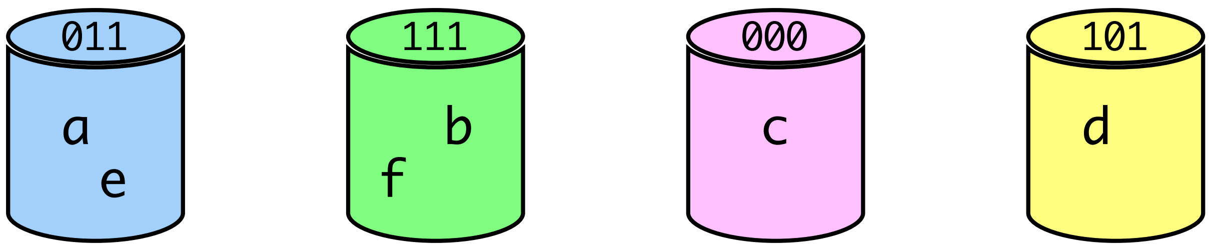
$b = 0101001 \quad g(b) = 111$

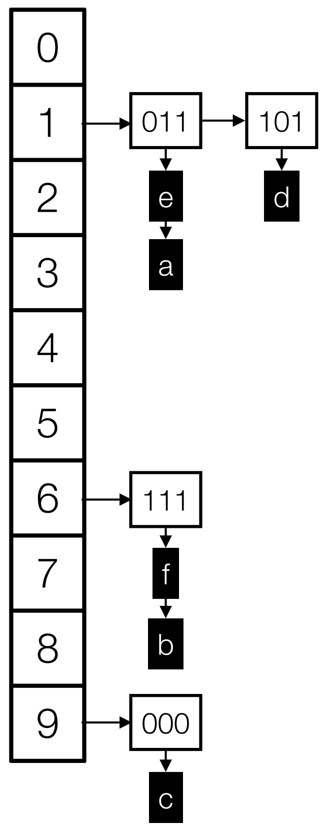
$c = 0010010 \quad g(c) = 000$

$d = 0110011 \quad g(d) = 101$

$e = 1011101 \quad g(e) = 011$

$f = 1101101 \quad g(f) = 111$





LSH with Hamming Distance: Solution 2

- Pick k random indexes uniformly and independently at random with replacement:
  - $g(x) = x_{i_1} x_{i_2} \cdots x_{i_k}$
- Bucket: Strings with same hash value $g(x)$.
- Save buckets in a hash table T with hash function h_T.
- Insert(x): Insert x in the list of $g(x)$ in T.
- NearNeighbour(x): Compute Hamming distance from x to all bitstrings in $g(x)$ until find one that is at most cr away. If no such string found return FAIL.

$h_T(011_2) = 1$

$h_T(111_2) = 6$

$h_T(000_2) = 9$

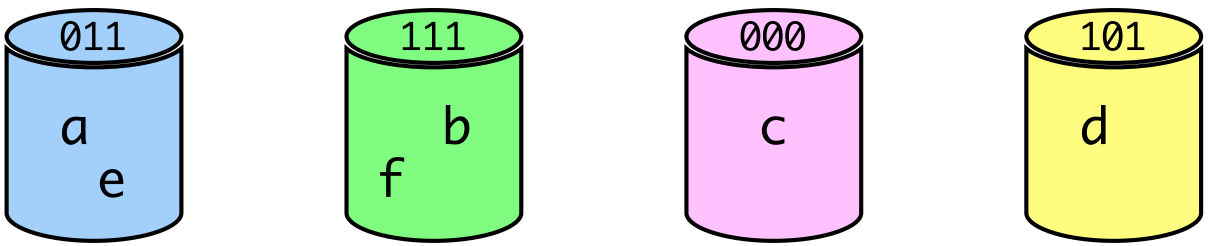
$h_T(101_2) = 1$

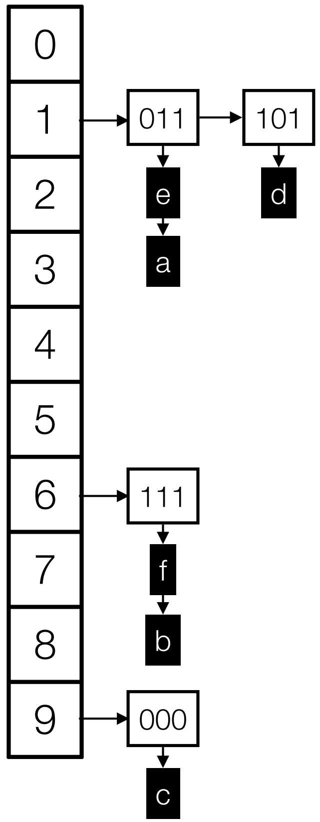
$g(x) = x_2 x_4 x_7$

$a = 0011101$ $\quad g(a) = 011$ $\qquad d = 0110011$ $\quad g(d) = 101$

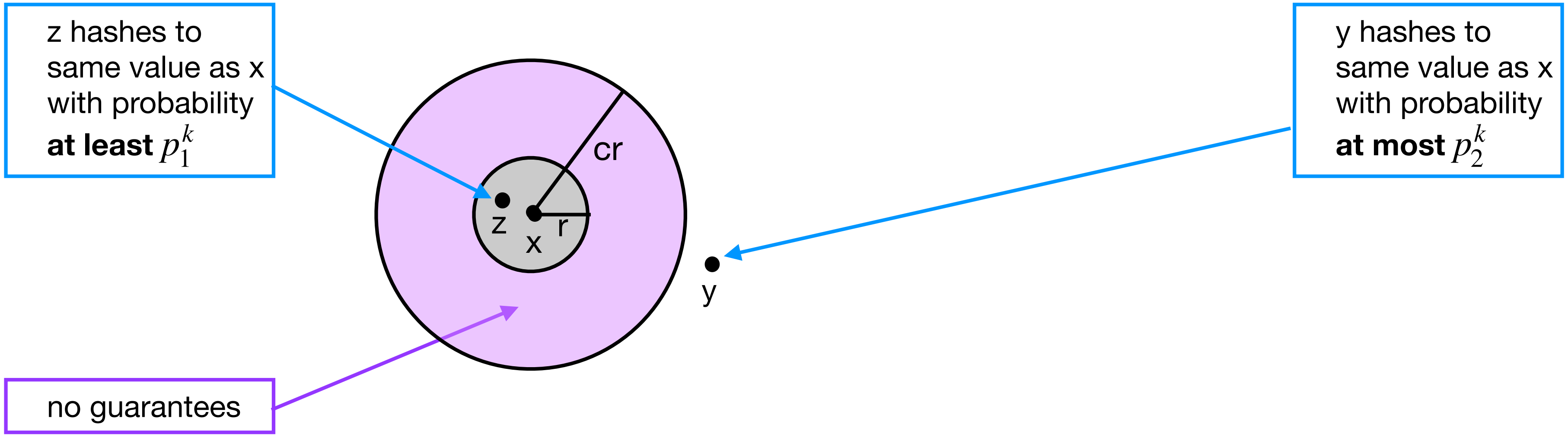
$b = 0101001$ $\quad g(b) = 111$ $\qquad e = 1011101$ $\quad g(e) = 011$

$c = 0010010$ $\quad g(c) = 000$ $\qquad f = 1101101$ $\quad g(f) = 111$



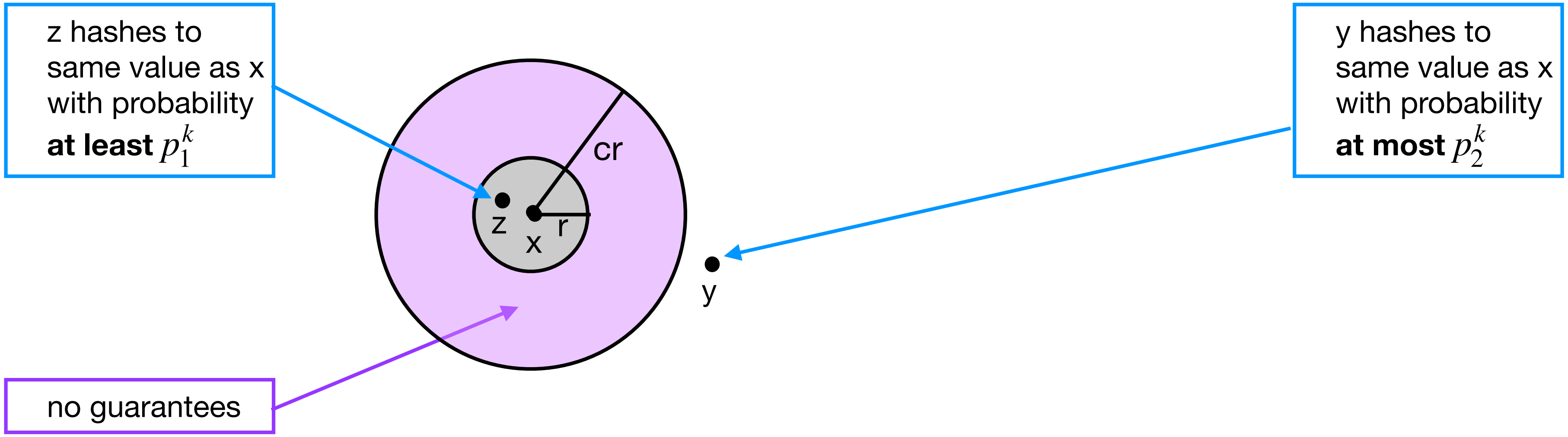


LSH with Hamming Distance: Solution 2



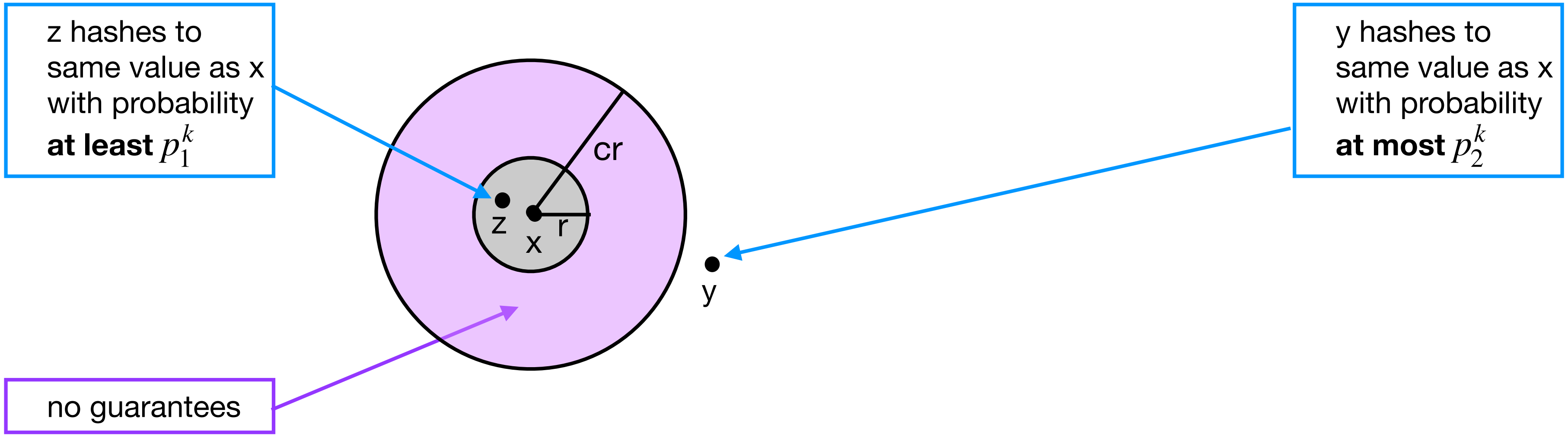
- What happens when we increase k?
  - Far away strings:

LSH with Hamming Distance: Solution 2



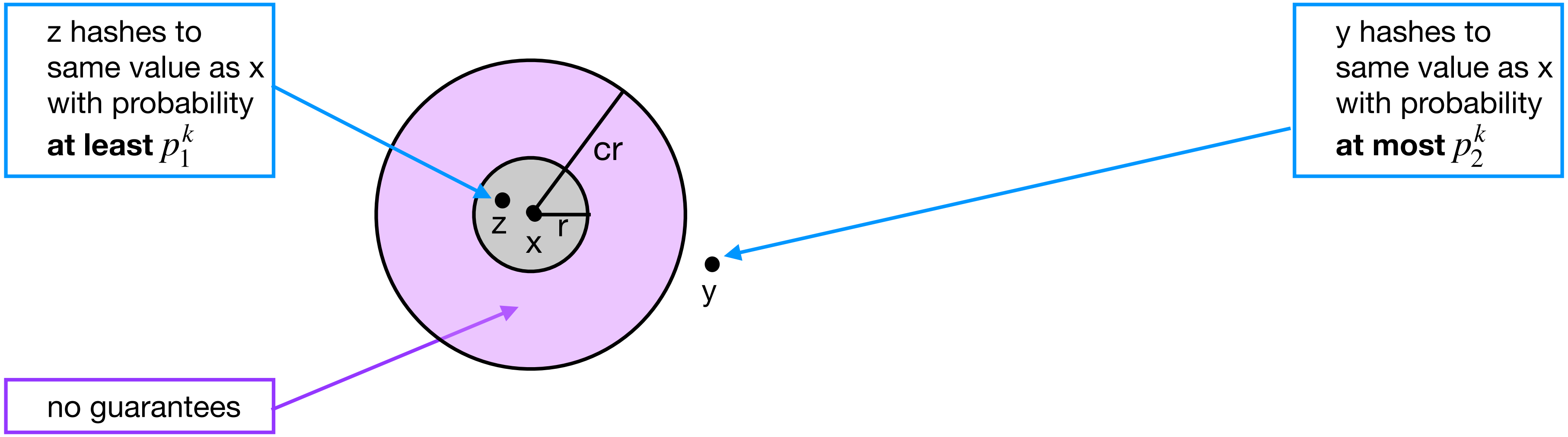
- What happens when we increase k?
  - Far away strings: Probability that a far away string hashes to the same bucket as x decrease.

LSH with Hamming Distance: Solution 2



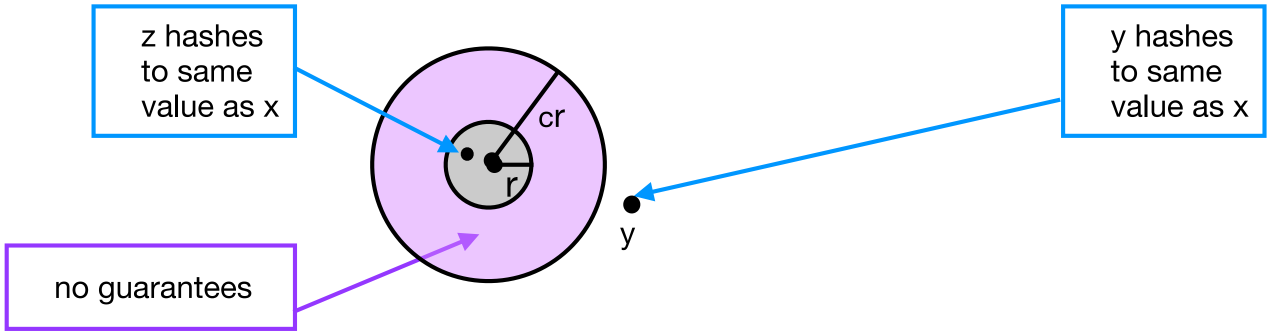
- What happens when we increase k?
  - Far away strings: Probability that a far away string hashes to the same bucket as x decrease.
  - Close strings:

LSH with Hamming Distance: Solution 2



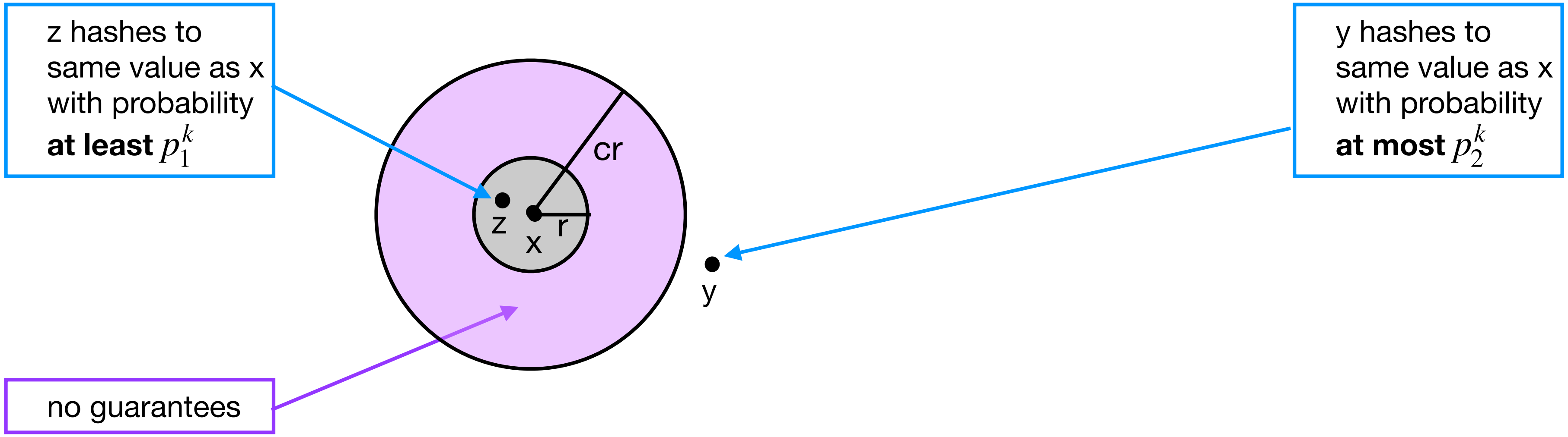
- What happens when we increase k?
  - Far away strings: Probability that a far away string hashes to the same bucket as x decrease.
  - Close strings: Probability that a close string hashes to the same as x decrease.

LSH with Hamming Distance: Solution 2



- Expected number of far away strings that hashes to same bucket as x:
  - $F = \{y : d(x, y) > cr\}$.
- For $y \in F$ we want $P[g(y) = g(x)] \leq 1/n$:
  - Set $k = \lg n / \lg(1/p_2)$
- $X_y = \begin{cases} 1 & y \text{ collides with } x \\ 0 & \text{otherwise} \end{cases}$
- #far away strings colliding with x: $X = \sum_{y \in F} X_y$
- $E[X] = \sum_{y \in F} E[X_y] = \sum_{y \in F} 1/n \leq 1$.
- Markov: $P[X > 6] < E[X]/6 \leq 1/6$.

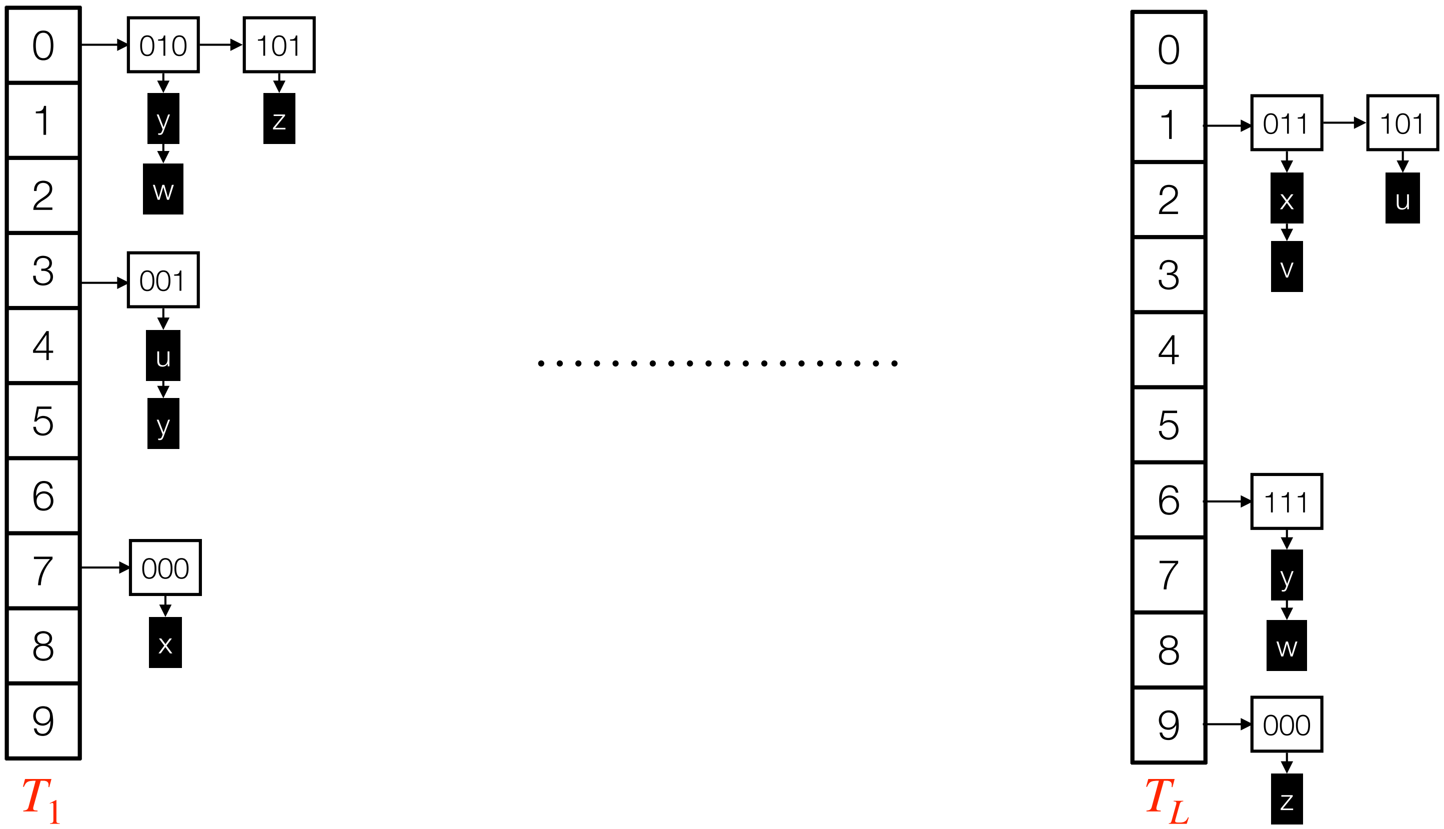
LSH with Hamming Distance: Solution 2



- What happens when we increase k?
  - Probability that a far away string hashes to the same bucket as x decrease.
    - $k = \lg n / \lg(1/p_2)$ $\Rightarrow$ with probability ≥ 5/6 at most 6 far away strings hashes to x's bucket.
  - Probability that a close string hashes to the same as x decrease. 😕

LSH with Hamming Distance: Solution 3 (Amplification)

- Construct L hash tables T_j. Each table T_j has its own independently chosen hash function h_j and its own independently chosen locality sensitive hash function g_j.
- **Insert(x):** For all $1 \leq j \leq L$ insert x in the list of $g_j(x)$ in T_j.
- **Query(x):** For all $1 \leq j \leq L$ check each element in bucket $g_j(x)$ in T_j. Return the one closest to x if it is at most cr away. Otherwise return FAIL.



LSH with Hamming Distance

Let $k = \dfrac{\lg n}{\lg(1/p_2)}$, $\rho = \dfrac{\lg(1/p_1)}{\lg(1/p_2)}$, and $L = \lceil 2n^{\rho} \rceil$, where $p_1 = 1 - r/d$ and $p_2 = 1 - cr/d$.

- Claim 1. *If there exists a string z* in P with d(x,z*) ≤ r then with probability at least 5/6 we will return some z in P for which d(x,z) ≤ r.*

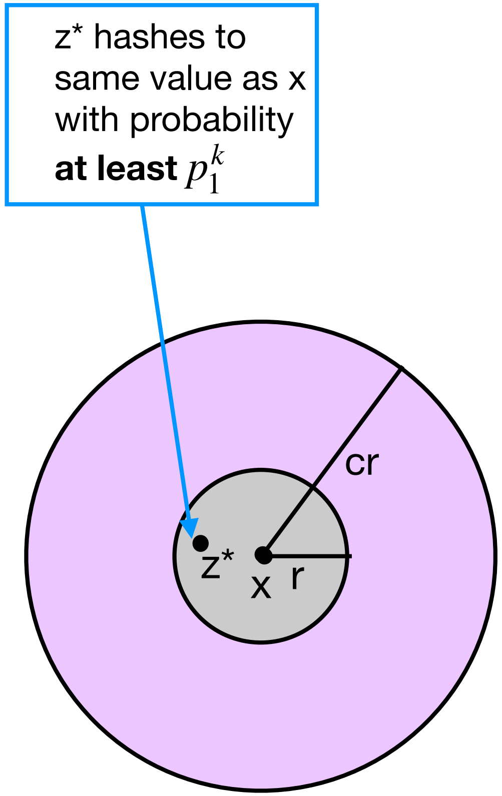
- Probability that z* collides with x:
  - $P[\exists i : g_i(x) = g_i(z^*)] \ = 1 - P[g_i(x) \neq g_i(z^*) \text{ for all } i]$

$$= 1 - \prod_{i=1}^{L} P[g_i(x) \neq g_i(z^*)]$$

$$= 1 - \prod_{i=1}^{L} \left(1 - P[g_i(x) = g_i(z^*)]\right)$$

$$\geq 1 - \prod_{i=1}^{L} (1 - p_1^k) \ = 1 - (1 - p_1^k)^L \ \geq 1 - e^{-Lp_1^k}$$

$$\geq 1 - \frac{1}{e^2} \geq 1 - 1/6 = 5/6$$

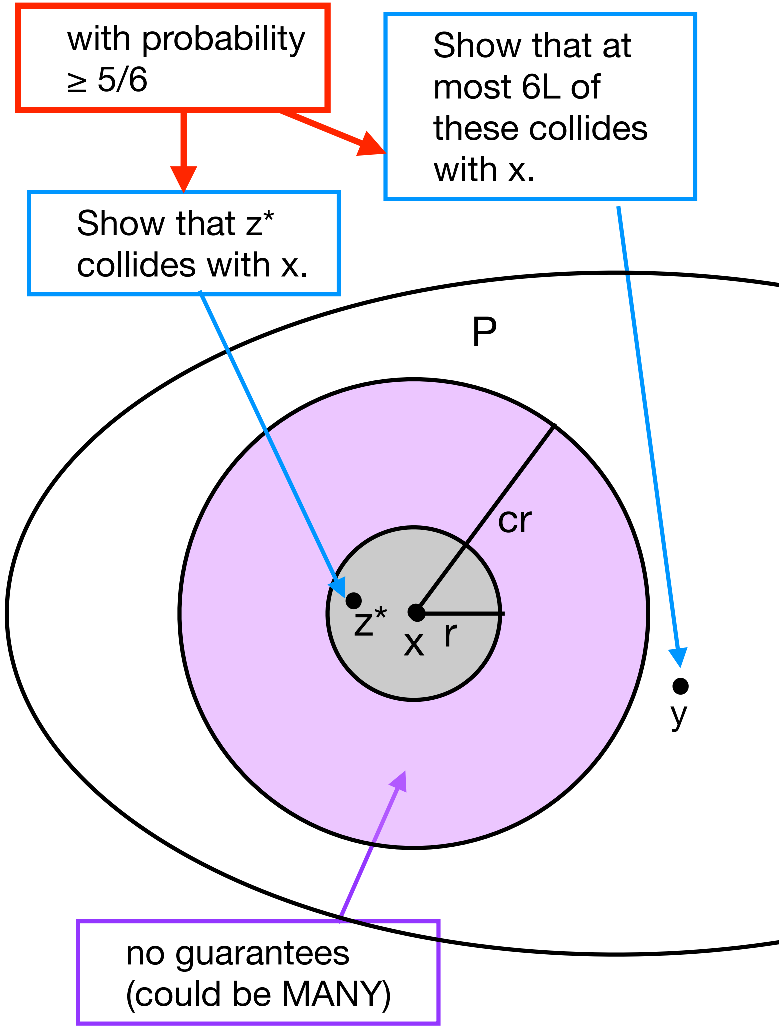


LSH with Hamming Distance

- Fast query time.
  - Check at most $6L + 1$ strings and return FAIL if no close string found.
  - Otherwise return closest string found.

- Theorem. *If there exists a string z^* in P with $d(x, z^*) \leq r$ then with probability at least $2/3$ we will return some y in P for which $d(x, y) \leq cr$.*

- Proof idea.
  - Show that with probability at least $5/6$ there are at most $6L$ *far away* strings that collides with x.
  - Already showed the probability that z^* is in the same bucket as x in at least one of the L hash tables is at least $5/6$.



LSH with Hamming Distance

- Insert time $O(kL)$.
- Expected query time $O(L(k+d))$.
  - $O(L)$ checks.
  - Each check takes $O(k+d)$ time.

Locality Sensitive Hashing

- Locality sensitive hash function. A family of hash functions $\mathscr{H}$ is (r, cr, p_1, p_2)-sensitive with $p_1 > p_2$ and $c > 1$ if:
  - $d(x,y) \leq r \quad \Rightarrow \quad P[h(x) = h(y)] \geq p_1$ (close points)
  - $d(x,y) \geq cr \quad \Rightarrow \quad P[h(x) = h(y)] \leq p_2$ (distant points)

- Amplification.
  - Choose L hash functions $g_j(x) = h_{1,j}(x) \cdot h_{2,j}(x) \ \cdots \ h_{k,j}(x)$, where $h_{i,j}$ is chosen independently and uniformly at random from $\mathscr{H}$.

- Locality sensitive hashing scheme.
  - Construct L hash tables T_j .
  - **Insert(x):** For all $1 \leq j \leq L$ insert x in the list of $g_j(x)$ in T_j .
  - **Query(x):** For all $1 \leq j \leq L$ check each element in bucket $g_j(x)$ in T_j. Return the one closest to x. Check at most $6L + 1$ elements. If no element found at distance less than $c \cdot r$ from x return FAIL.

Jaccard distance and Min Hash

- Jaccard distance. Jaccard similarity: Jsim$(A, B) = \dfrac{|A \cap B|}{|A \cup B|}$
    - Jaccard distance: 1- Jsim(A,B).
    - Hash function: *Min Hash*. (exercise)

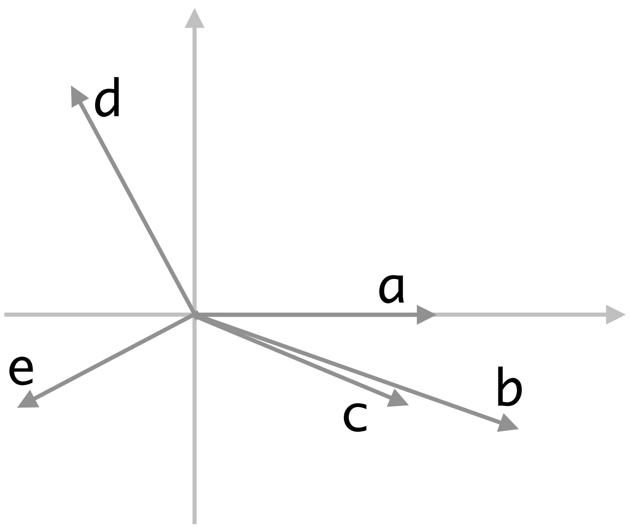
Exercises

Angular Distance and Sim Hash

- Collection of vectors.
- Distance between two vectors is the angular distance between them dist$(u, v) = \angle(u, v)/\pi$.
  - Assume u and v are unit vectors. Then $u \cdot v = \cos(\angle(u, v))$
- Hash function: Sim Hash.
  - Random projection: Take a random vector r and set $h_r(u) = \text{sign}(r \cdot u)$

Angular Distance and Sim Hash

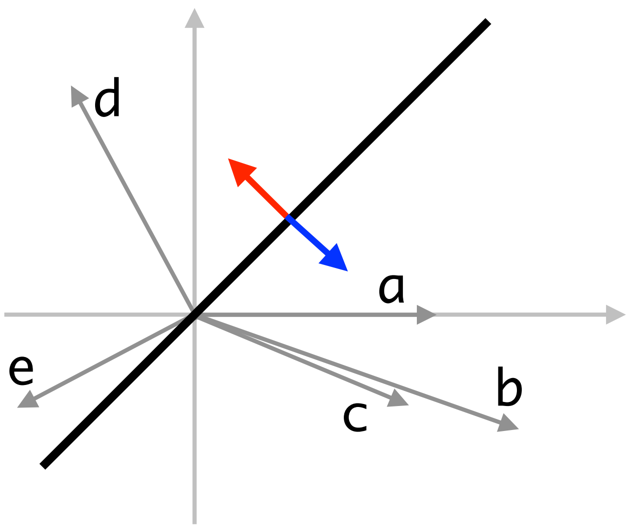
- Collection of vectors.
- Distance between two vectors is the angular distance between them $\text{dist}(u, v) = \angle(u, v)/\pi$.
  - Assume u and v are unit vectors. Then $u \cdot v = \cos(\angle(u, v))$
- Hash function: Sim Hash.
  - Random projection: Take a random vector r and set $h_r(u) = \text{sign}(r \cdot u)$

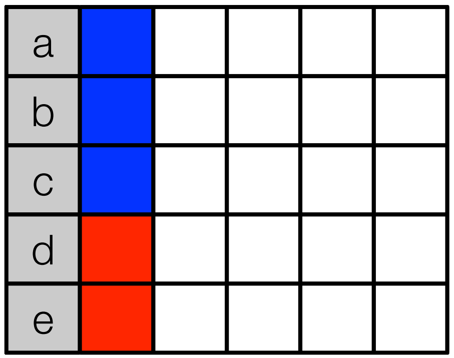


a					
b					
c					
d					
e					

Angular Distance and Sim Hash

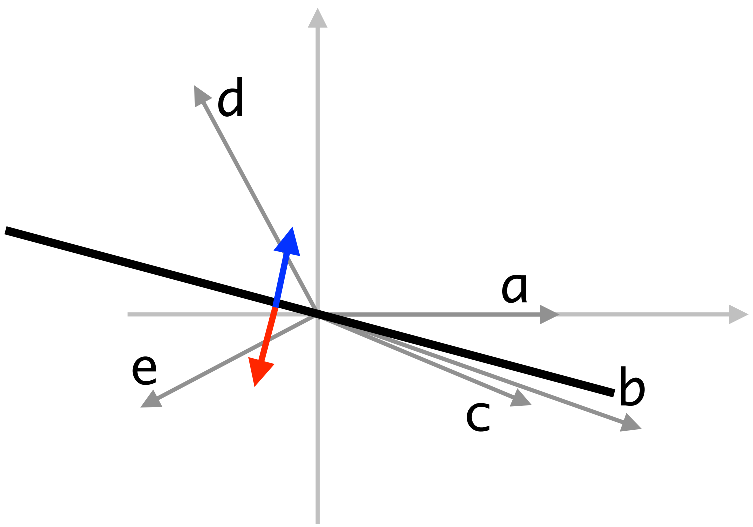
- Collection of vectors.
- Distance between two vectors is the angular distance between them $\mathrm{dist}(u, v) = \angle(u, v)/\pi$.
    - Assume u and v are unit vectors. Then $u \cdot v = \cos(\angle(u, v))$
- Hash function: Sim Hash.
    - Random projection: Take a random vector r and set $h_r(u) = \mathrm{sign}(r \cdot u)$

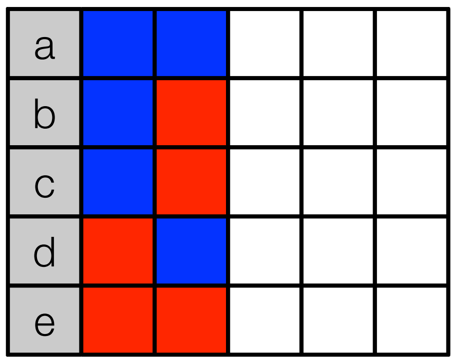




Angular Distance and Sim Hash

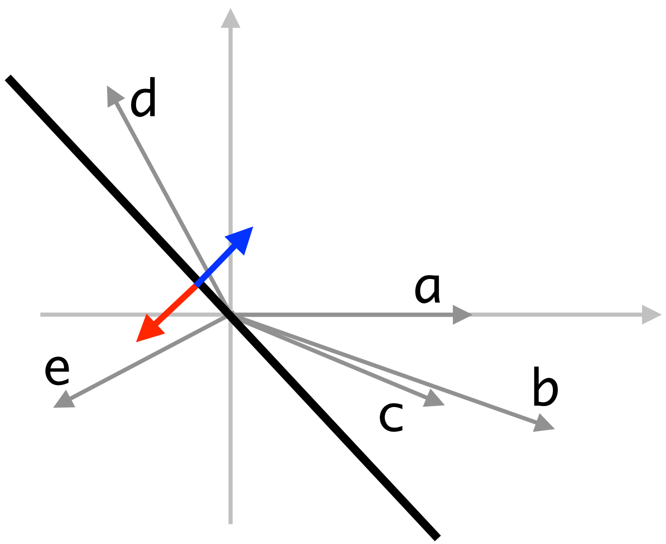
- Collection of vectors.
- Distance between two vectors is the angular distance between them $\text{dist}(u, v) = \angle(u, v)/\pi$.
  - Assume u and v are unit vectors. Then $u \cdot v = \cos(\angle(u, v))$
- Hash function: Sim Hash.
  - Random projection: Take a random vector r and set $h_r(u) = \text{sign}(r \cdot u)$

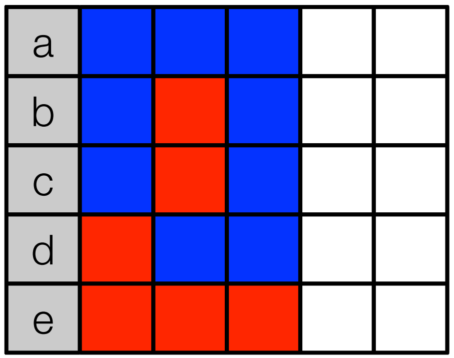




Angular Distance and Sim Hash

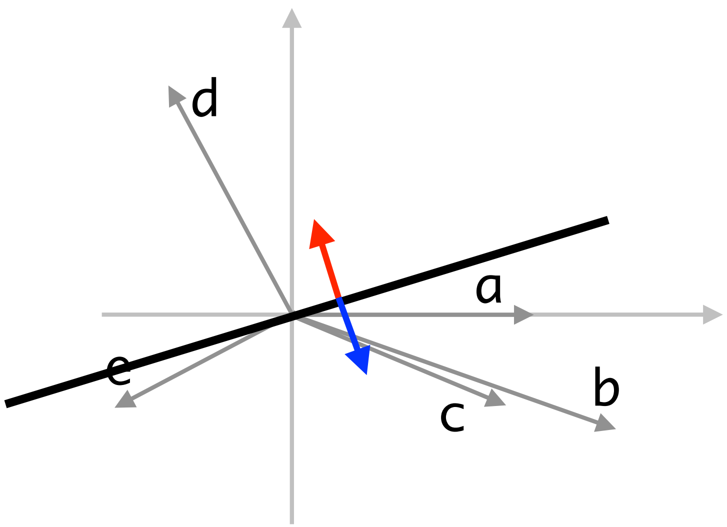
- Collection of vectors.
- Distance between two vectors is the angular distance between them $\text{dist}(u, v) = \angle(u, v)/\pi$.
  - Assume u and v are unit vectors. Then $u \cdot v = \cos(\angle(u, v))$
- Hash function: Sim Hash.
  - Random projection: Take a random vector r and set $h_r(u) = \text{sign}(r \cdot u)$

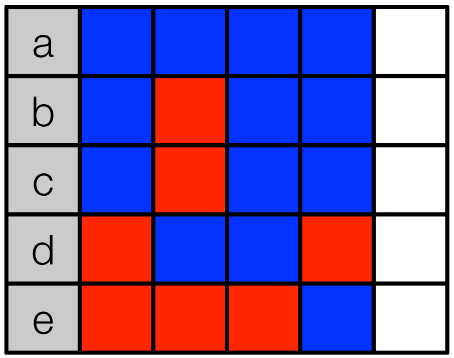




Angular Distance and Sim Hash

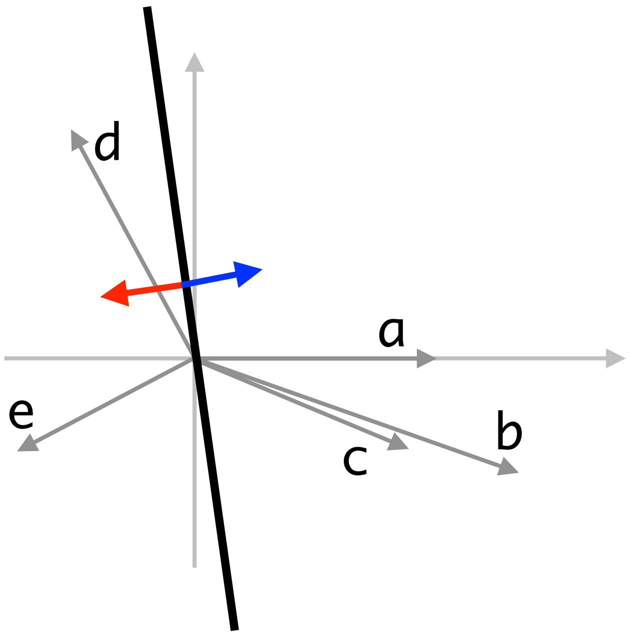
- Collection of vectors.
- Distance between two vectors is the angular distance between them $\text{dist}(u, v) = \angle(u, v)/\pi$.
  - Assume u and v are unit vectors. Then $u \cdot v = \cos(\angle(u, v))$
- Hash function: Sim Hash.
  - Random projection: Take a random vector r and set $h_r(u) = \text{sign}(r \cdot u)$

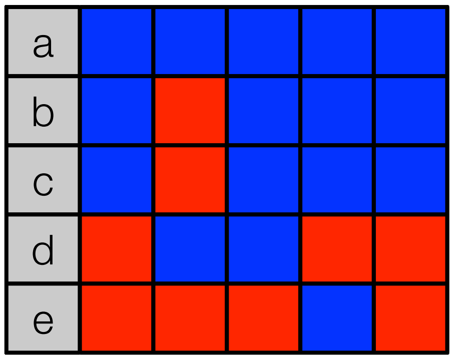




Angular Distance and Sim Hash

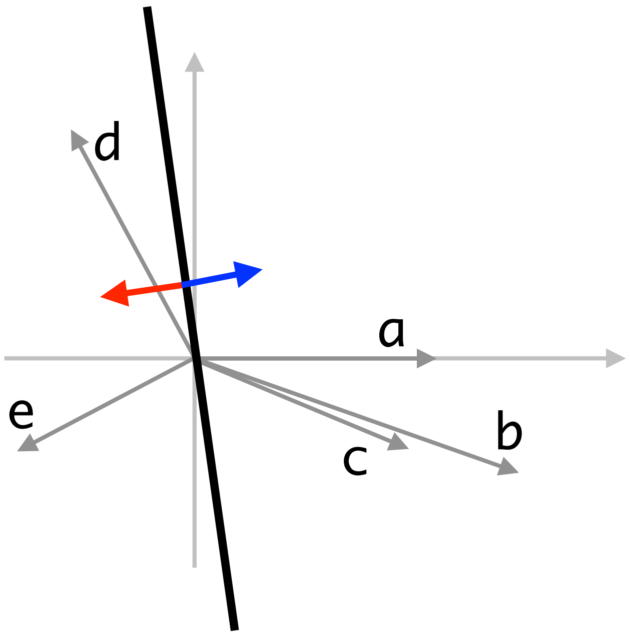
- Collection of vectors.
- Distance between two vectors is the angular distance between them dist$(u, v) = \angle(u, v)/\pi$.
    - Assume u and v are unit vectors. Then $u \cdot v = \cos(\angle(u, v))$
- Hash function: Sim Hash.
    - Random projection: Take a random vector r and set $h_r(u) = \mathrm{sign}(r \cdot u)$

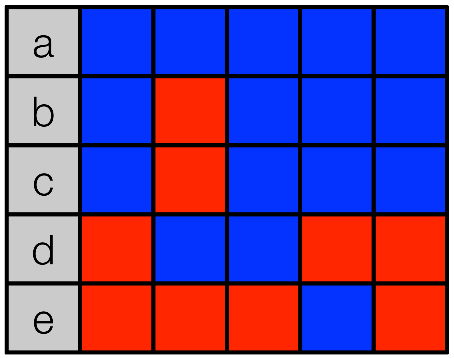




Angular Distance and Sim Hash

- Collection of vectors.
- Distance between two vectors is the angular distance between them $\text{dist}(u, v) = \angle(u, v)/\pi$.
    - Assume u and v are unit vectors. Then $u \cdot v = \cos(\angle(u, v))$
- Hash function: Sim Hash.
    - Random projection: Take a random vector r and set $h_r(u) = \text{sign}(r \cdot u)$





- Can show that $P[h(u) = h(v)] = 1 - \angle(u, v)/\pi$.