Approximate Near Neighbor Search: Locality Sensitive Hashing

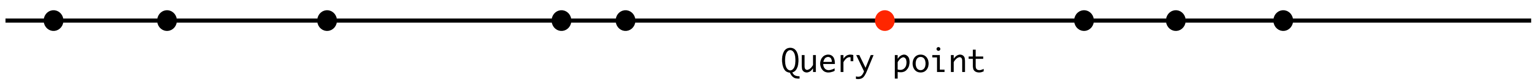
Inge Li Gørtz

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

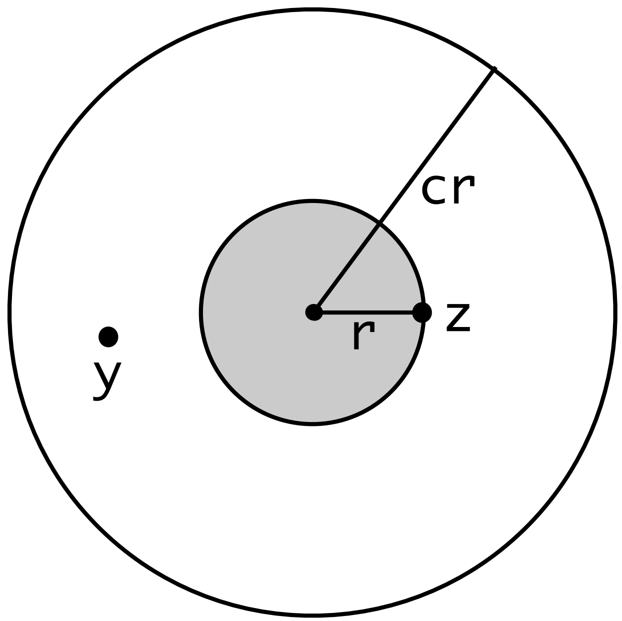


Approximate Near Neighbors

- ApproximateNearNeighbor(x): Return a point y such that $d(x, y) \leq c \cdot \min_{z \in P} d(x, z)$

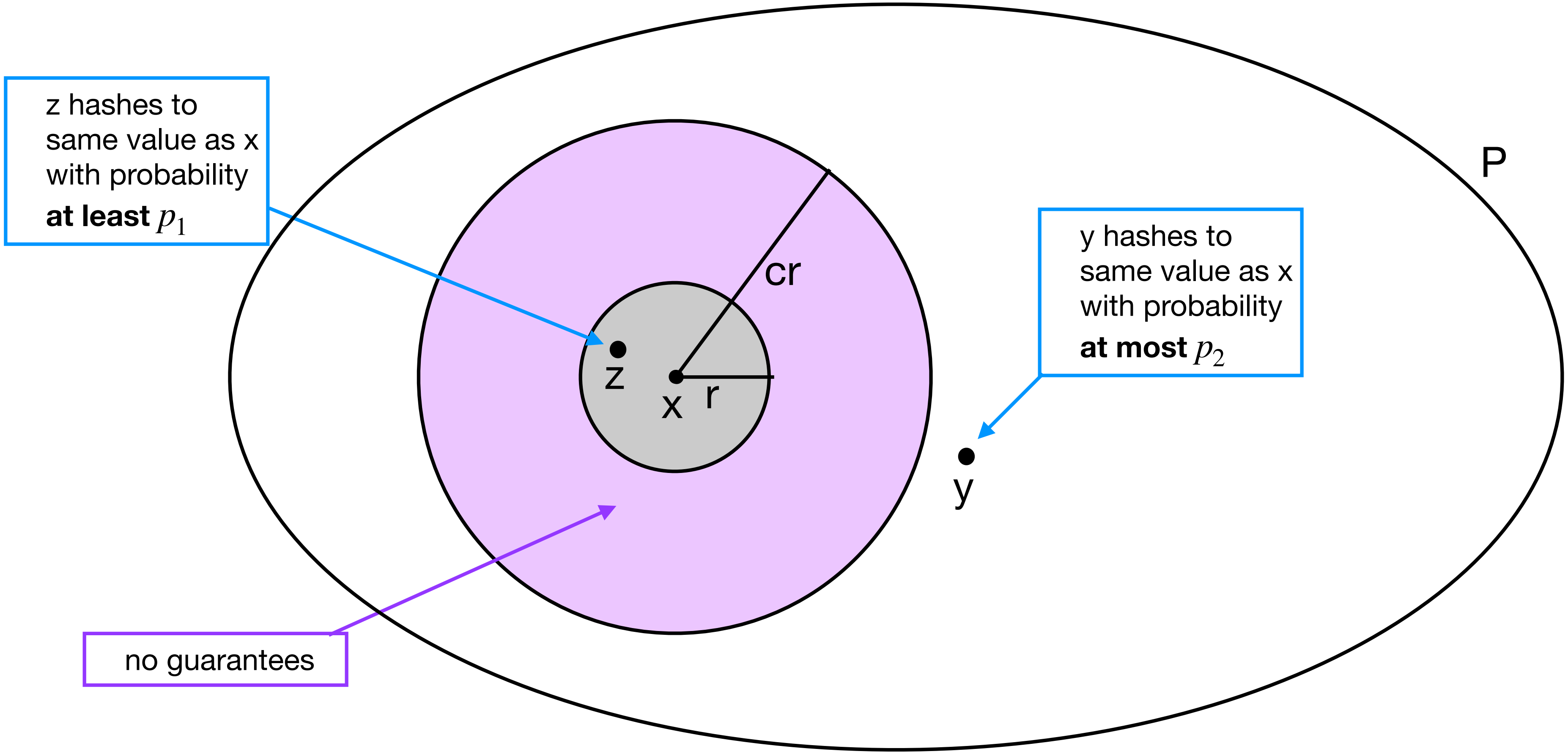
- c-Approximate r-Near Neighbor: Given a point x if there exists a point z in P $d(x, z) \leq r$ then return a point y such that $d(x, y) \leq c \cdot r$. If no such point z exists return Fail.

- 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)



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.

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

`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

- Pick k random indexes uniformly and independently with replacement. Let
  - $g(x) = x_{i_1} x_{i_2} \cdots x_{i_k}$

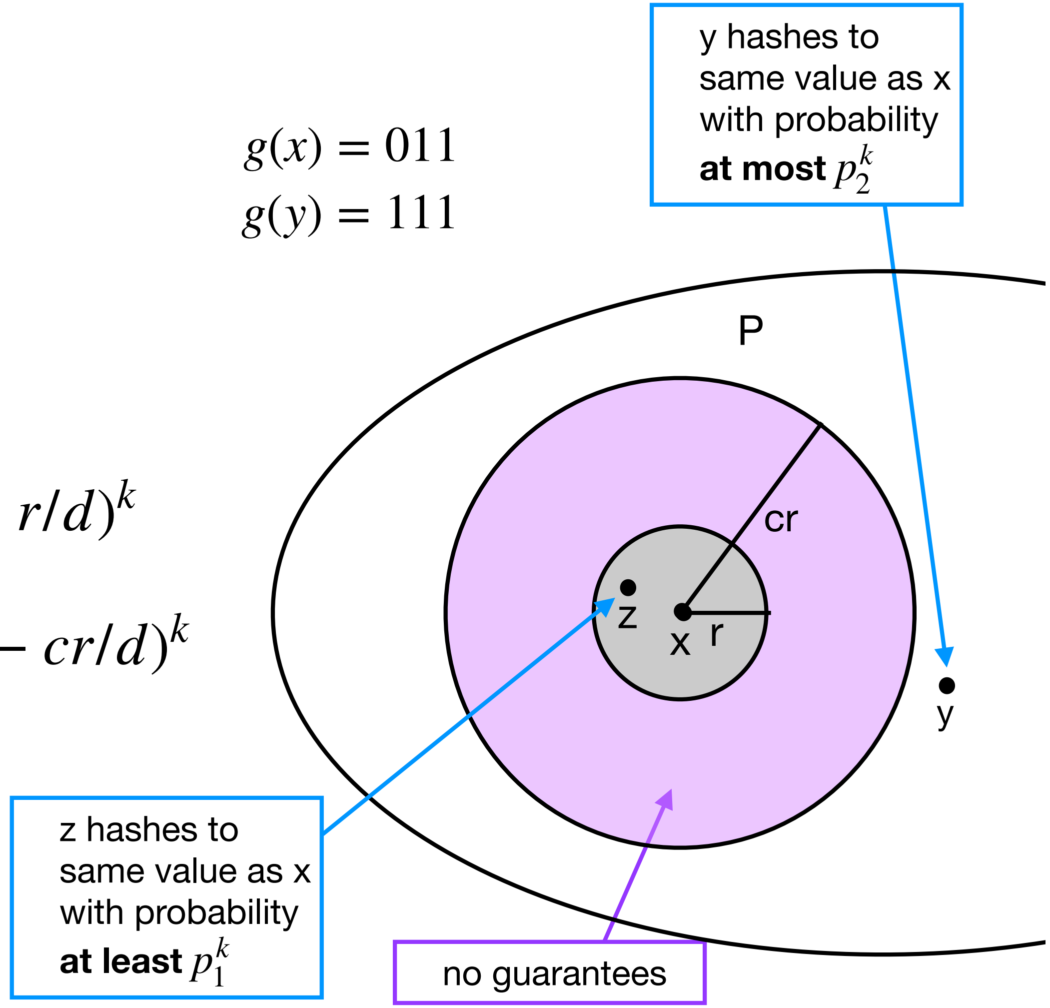
- Example. k = 3. $g(x) = x_2 x_3 x_6$

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



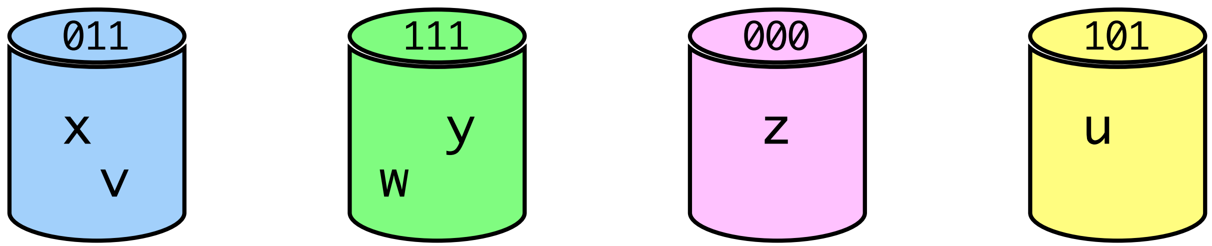
LSH with Hamming Distance

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

$g(x) = 011$ $\quad g(u) = 101$

$g(y) = 111$ $\quad g(v) = 011$

$g(z) = 000$ $\quad g(w) = 111$



LSH with Hamming Distance

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

$g(x) = 011$ $\quad$ $g(u) = 101$

$g(y) = 111$ $\quad$ $g(v) = 011$

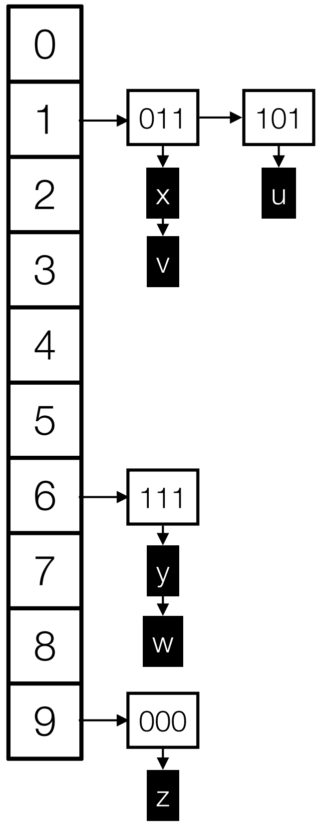
$g(z) = 000$ $\quad$ $g(w) = 111$

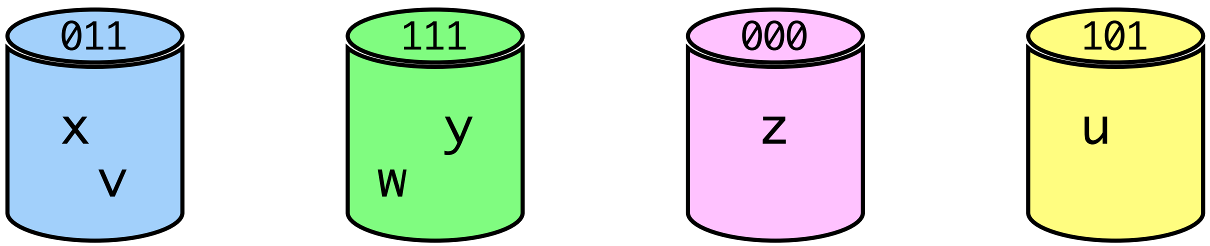
$h_T(011) = 1$

$h_T(111) = 6$

$h_T(000) = 9$

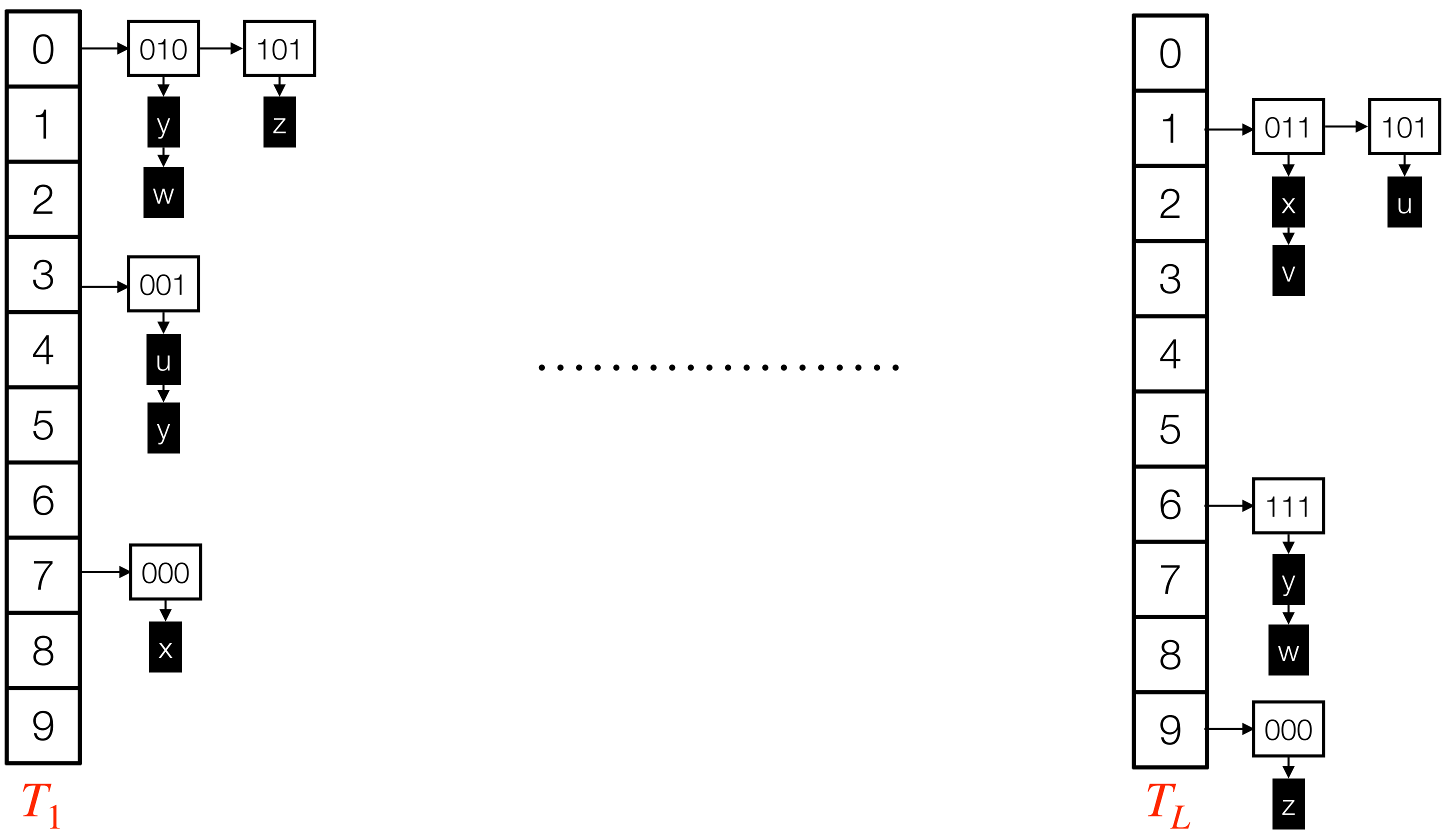
$h_T(101) = 1$





LSH with Hamming Distance: 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):** 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.



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.*

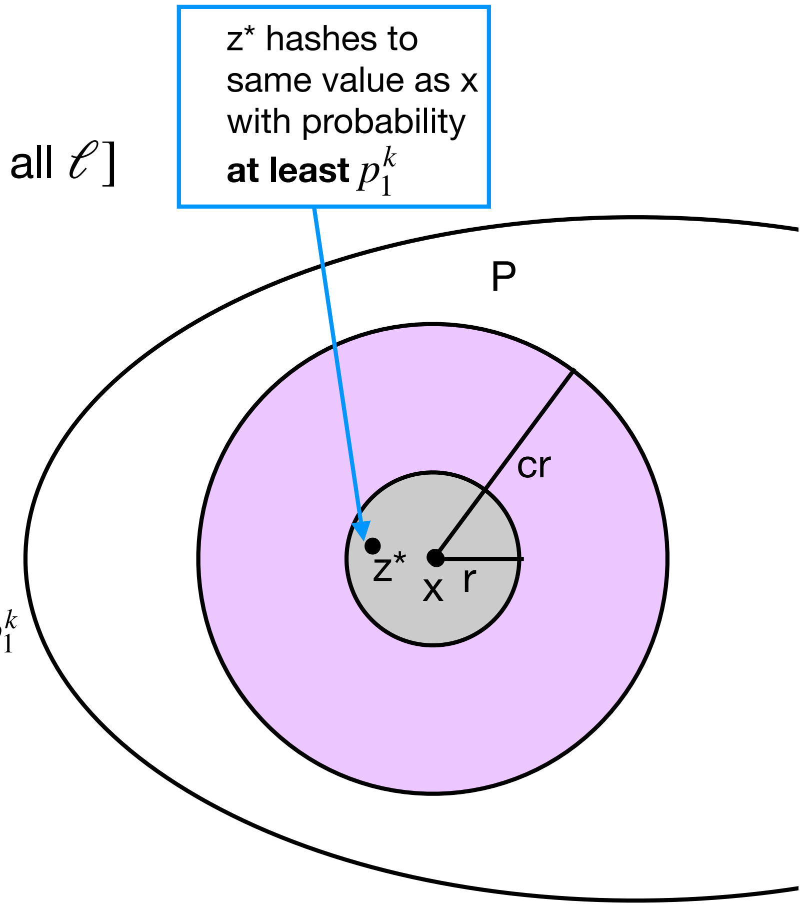
- Probability that z* collides with x:
  - $P[\exists \ell : g_\ell(x) = g_\ell(z^*)] \; = 1 - P[g_\ell(x) \neq g_\ell(z^*)$ for all $\ell]$

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

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

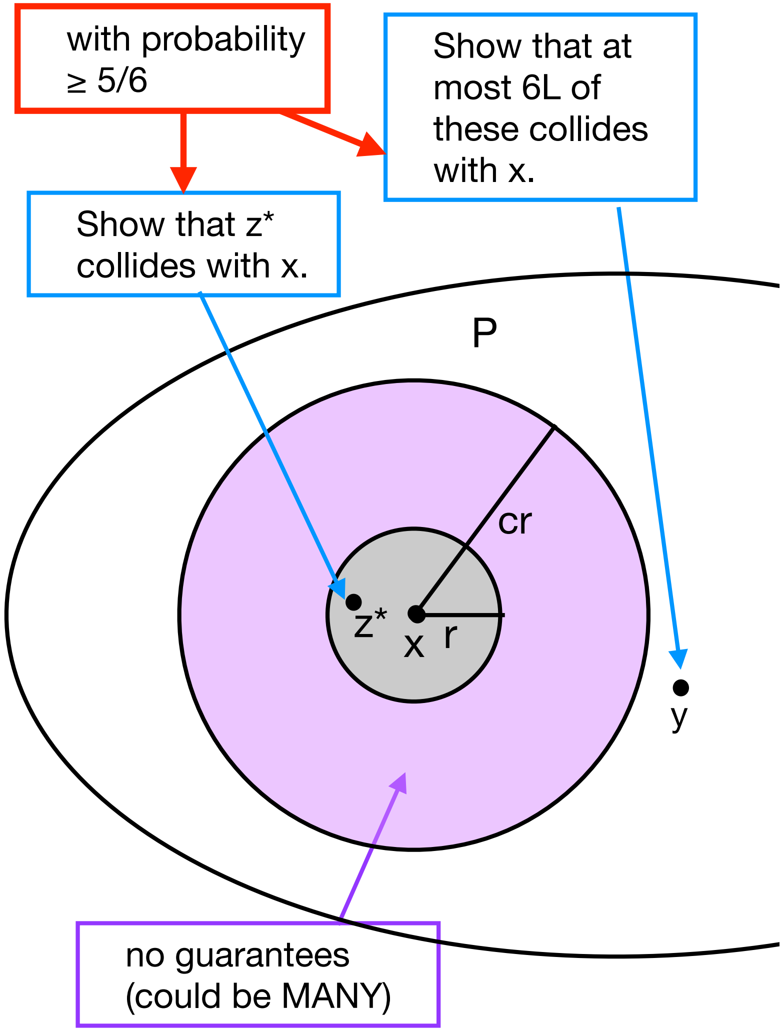
$$\geq 1 - \prod_{\ell=1}^{L} (1 - p_1^k) \; = 1 - (1 - p_1^k)^L \; \geq 1 - e^{-Lp_1^k}$$

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



LSH with Hamming Distance

- Check strings in buckets until we find one that is at most Cr away from x. Return closest.
- To ensure a query time of $O(L)$ we stop checking strings in the buckets after we have checked 6L+1 and return FAIL.
- Theorem. *If there exists a string z* in P with d(x,z*) ≤ r then with probability at least 2/3 we will return some y in P for which d(x,y) ≤ cr.*
- Proof idea.
  - Show that with probability at least 5/6 there are at most 6L strings far away 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.



Locality Sensitive Hashing

- Locality sensitive hashing. 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 $\mathscr{H}$.

Jaccard distance and Min Hash

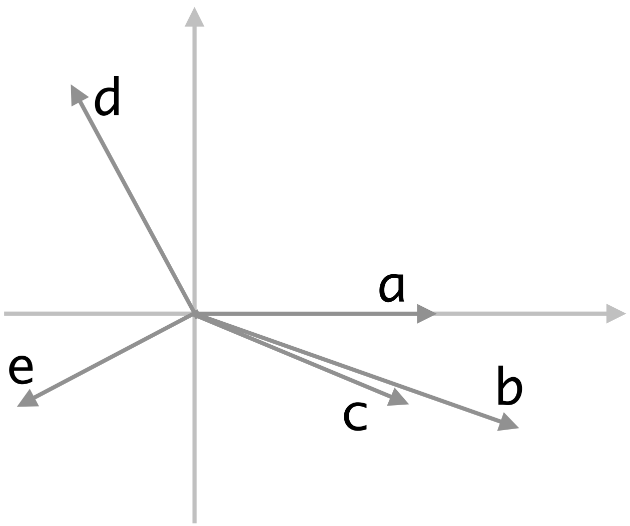
- Jaccard distance. Jaccard similarity: $\text{Jsim}(A, B) = \dfrac{|A \cap B|}{|A \cup B|}$
  - Jaccard distance: 1- Jsim(A,B).
  - Hash function: *Min Hash*. (exercise)

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

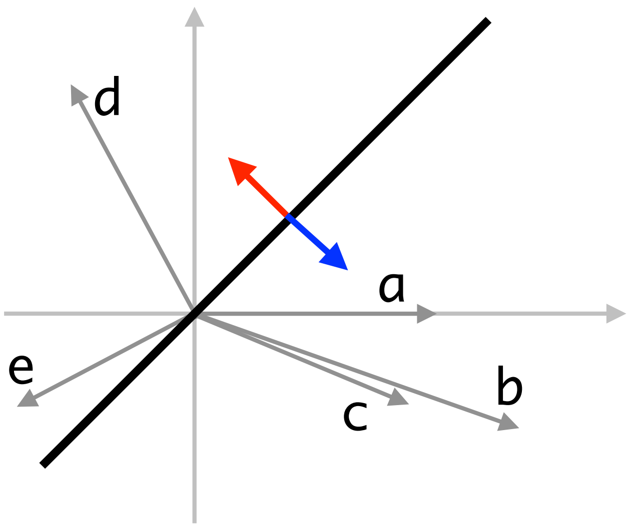
- Collection of vectors.
- Distance between two vectors is the angular distance between them $\text{dist}(u, v) = \angle(u, v)/\pi$.
    - Assume u and v are unit vectors. Then $u \cdot v = \cos(\angle(u, v))$
- Hash function: Sim Hash.
    - Random projection: Take a random vector r and set $h_r(u)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

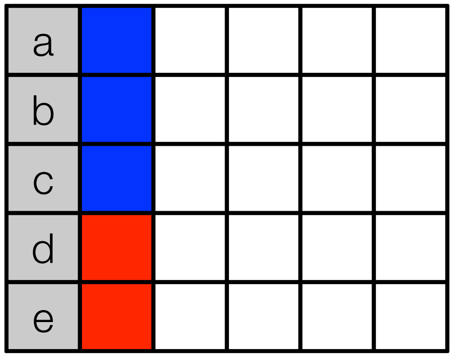


a					
b					
c					
d					
e					

Angular Distance and Sim Hash

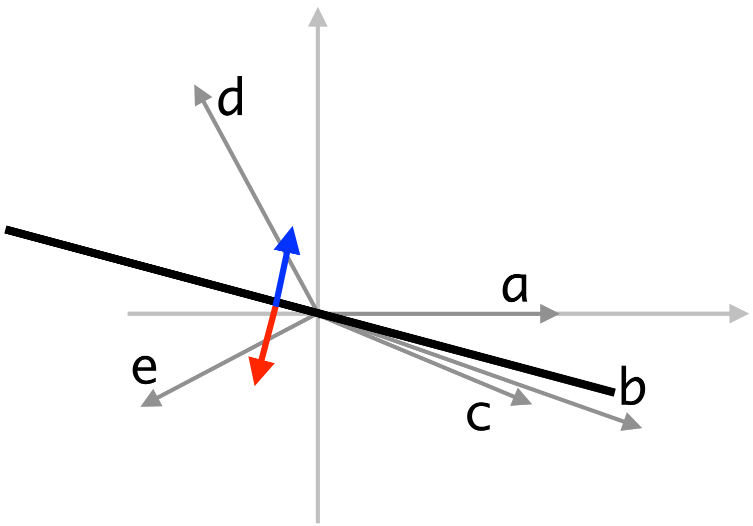
- Collection of vectors.
- Distance between two vectors is the angular distance between them $\text{dist}(u, v) = \angle(u, v)/\pi$.
    - Assume u and v are unit vectors. Then $u \cdot v = \cos(\angle(u, v))$
- Hash function: Sim Hash.
    - Random projection: Take a random vector r and set $h_r(u)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

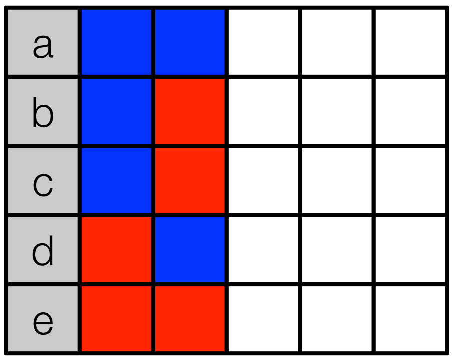




Angular Distance and Sim Hash

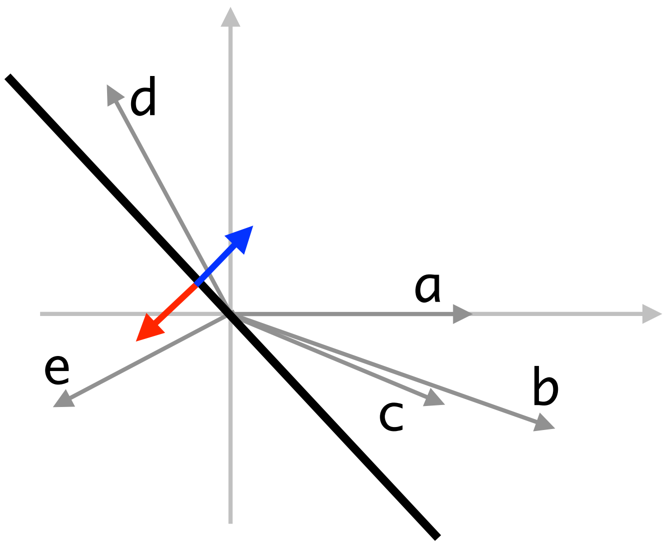
- Collection of vectors.
- Distance between two vectors is the angular distance between them $\text{dist}(u, v) = \angle(u, v)/\pi$.
  - Assume u and v are unit vectors. Then $u \cdot v = \cos(\angle(u, v))$
- Hash function: Sim Hash.
  - Random projection: Take a random vector r and set $h_r(u) = \text{sign}(r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

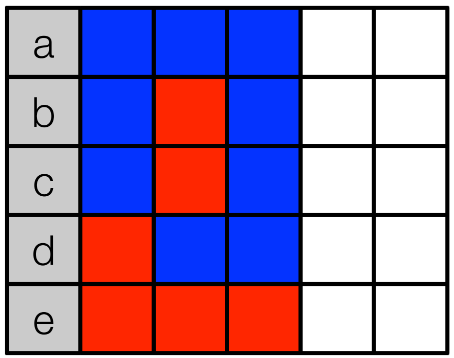




Angular Distance and Sim Hash

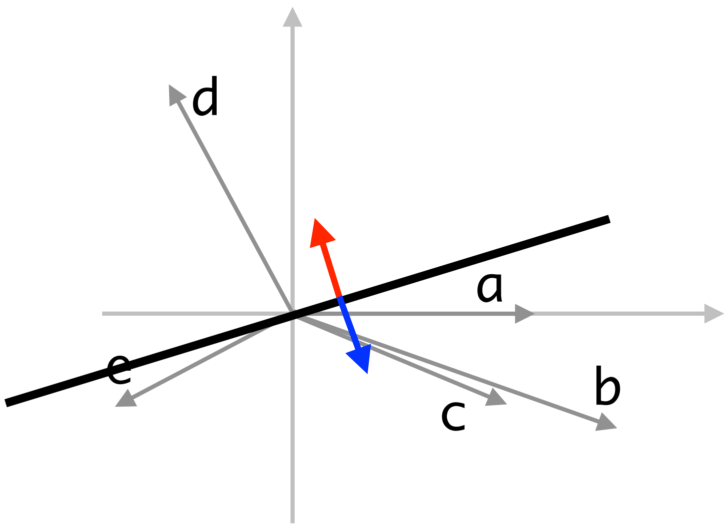
- Collection of vectors.
- Distance between two vectors is the angular distance between them $\text{dist}(u, v) = \angle(u, v)/\pi$.
  - Assume u and v are unit vectors. Then $u \cdot v = \cos(\angle(u, v))$
- Hash function: Sim Hash.
  - Random projection: Take a random vector r and set $h_r(u) = \text{sign}(r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

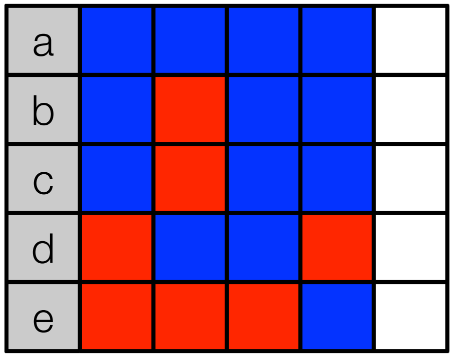




Angular Distance and Sim Hash

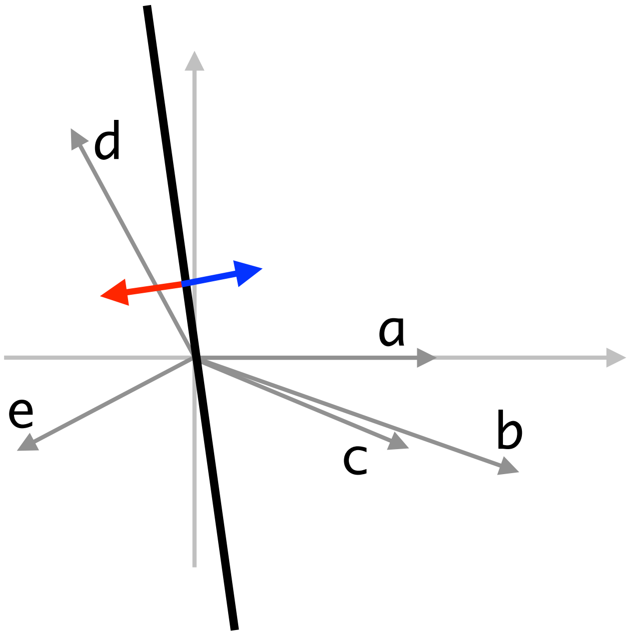
- Collection of vectors.
- Distance between two vectors is the angular distance between them $\mathrm{dist}(u, v) = \angle(u, v)/\pi$.
  - Assume u and v are unit vectors. Then $u \cdot v = \cos(\angle(u, v))$
- Hash function: Sim Hash.
  - Random projection: Take a random vector r and set $h_r(u) =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

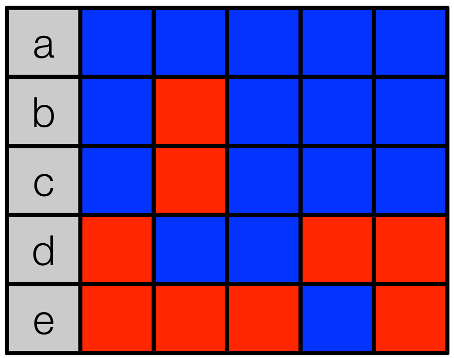




Angular Distance and Sim Hash

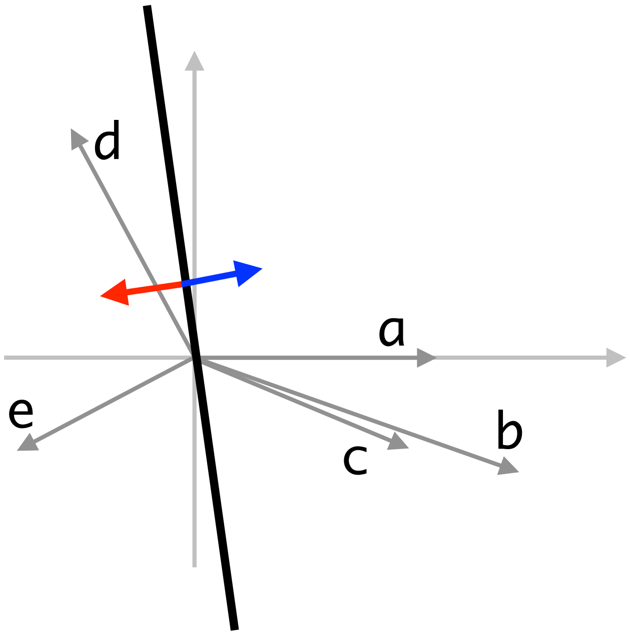
- Collection of vectors.
- Distance between two vectors is the angular distance between them $\mathrm{dist}(u, v) = \angle(u, v)/\pi$.
    - Assume u and v are unit vectors. Then $u \cdot v = \cos(\angle(u, v))$
- Hash function: Sim Hash.
    - Random projection: Take a random vector r and set $h_r(u)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

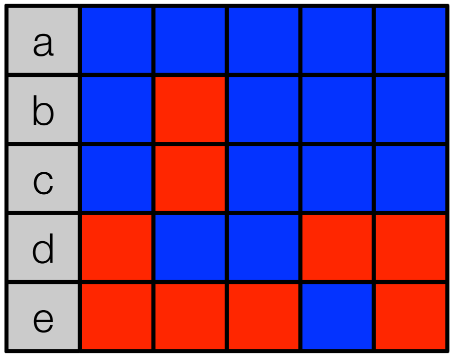




Angular Distance and Sim Hash

- 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 $h_r(u) = \text{sign}(r \cdot u)$





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