

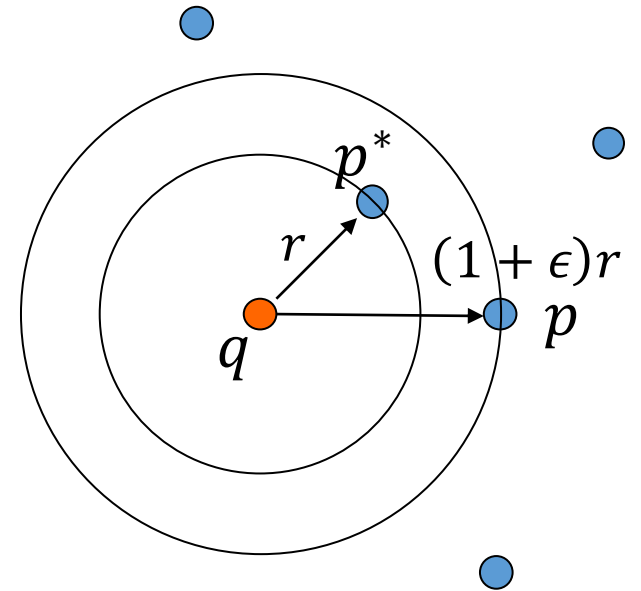
Locality-Sensitive Hashing (theory)

Similarity Search in High Dimensions via Hashing

Why LSH?

- Nearest- or near-neighbor query in high dimension

- “curse of dimensionality”
- approximate answer suffice



- Idea

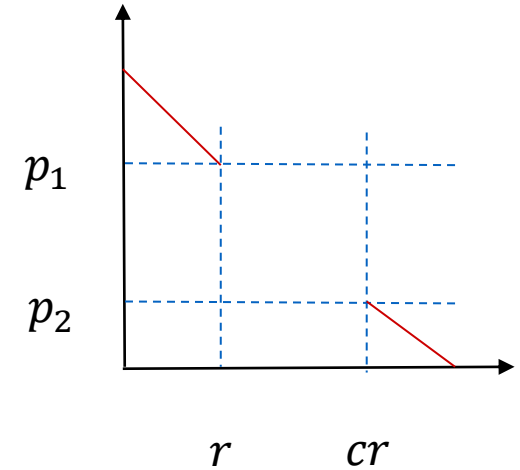
- “Probability of collision is much higher for points that are close to each other than for those that are far away”

sensitive

locality

LSH-family

- Fix a distance function $d : U \times U \rightarrow R$. For positive r, c , p_1, p_2 and $c > 1, p_1 > p_2$, a family of functions H is (r, cr, p_1, p_2) -sensitive if for uniformly chosen $h \in H$ and all $p, q \in U$:
 - If $d(p, q) \leq r$, then $\Pr[h(p) = h(q)] \geq p_1$
 - If $d(p, q) \geq cr$, then $\Pr[h(p) = h(q)] \leq p_2$



- H is **monotonic** if $\Pr[h(p)=h(q)]$ is a non-increasing function of $d(p,q)$
- Measure $\rho = \frac{\ln p_1}{\ln p_2}$: **smaller ρ , better distinguish ability**

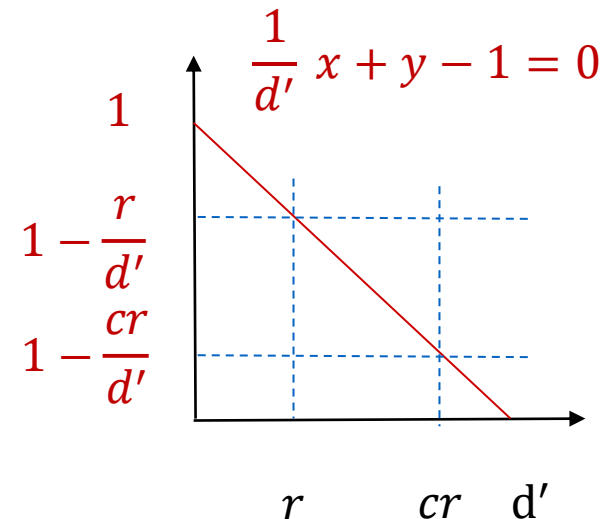
Hamming space

- Point Transformation $p \rightarrow v(p)$

$$(x_1, \dots, x_d) \rightarrow \underbrace{1 \dots 1}_{x_1} \underbrace{0 \dots 0}_{C-x_1} \dots \underbrace{1 \dots 1}_{x_d} \underbrace{0 \dots 0}_{C-x_d}$$

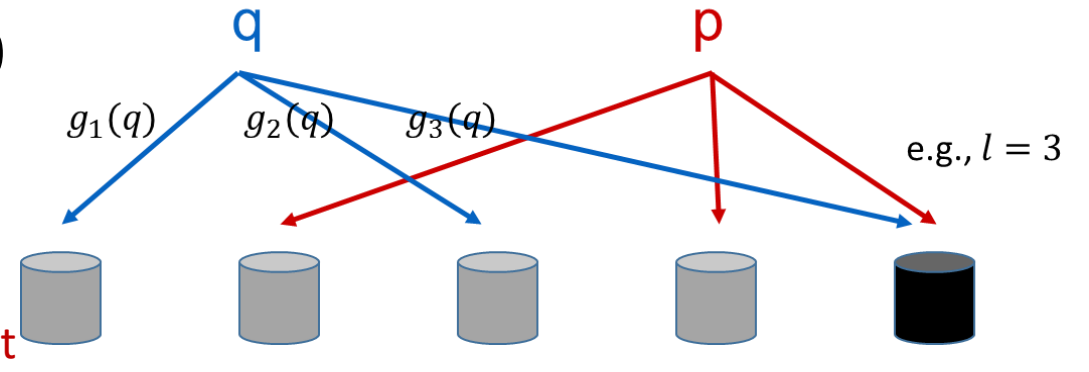
the largest coordinate in all points

- $d' = Cd$
 - $d_1(p, q) = d_H(v(p), v(q))$ # bits on which $v(p)$ and $v(q)$ differ
- Dimension reduction
 - $H_{d'} = \{h_i: h_i(p) \text{ is the } i\text{-th bit of } v(p), i = 1, 2, \dots, d'\}$



$(r, cr, 1 - \frac{r}{d'}, 1 - \frac{cr}{d'})$ -sensitive, $\rho < \frac{1}{c}$ (proof omitted)

Boosting Probability Gap



- Define $G = \{g_j | g_j = (h_{j_1}, h_{j_2}, \dots, h_{j_k}), j = 1, 2, \dots, l\}$
 - $h_{j_1}, h_{j_2}, \dots, h_{j_k}$ are randomly chosen from $H_{d'}$ with replacement

$$\Pr[p, q \text{ collide}] = 1 - \left(1 - \left(1 - \frac{d(p, q)}{d'} \right)^k \right)^l$$

collide under at least one g_j

collide under g_j

all $g_1, g_2, \dots, g_l$ fail

$$p_1 \rightarrow 1 - (1 - p_1^k)^l$$

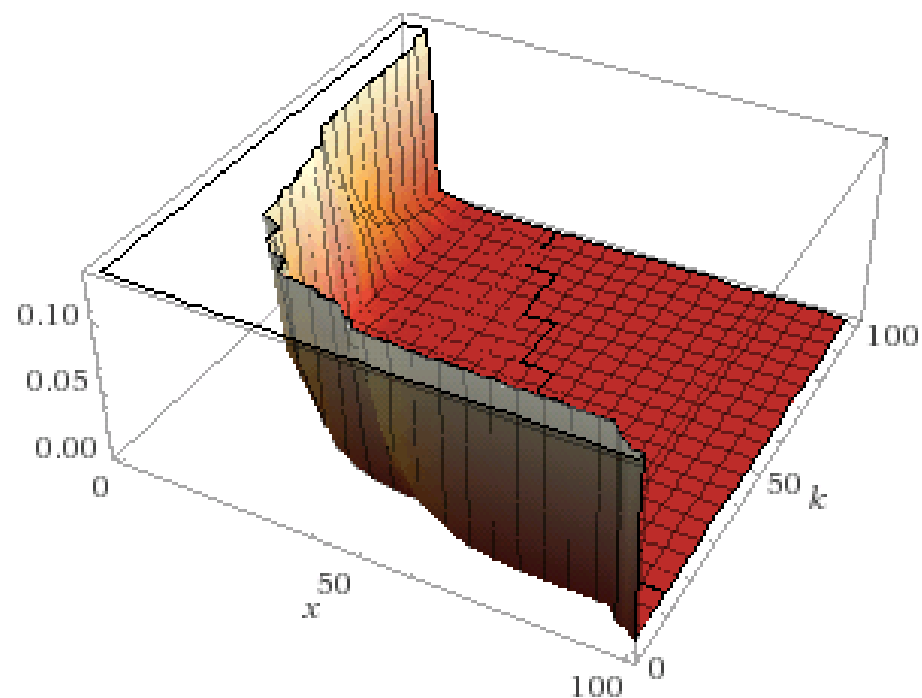
$$p_2 \rightarrow 1 - (1 - p_2^k)^l$$

Fix $d' = 100, l = 10$

Input interpretation:

plot	$1 - \left(1 - \left(1 - \frac{x}{100}\right)^k\right)^{10}$	$x = 0 \text{ to } 100$
------	--	-------------------------

3D plot:

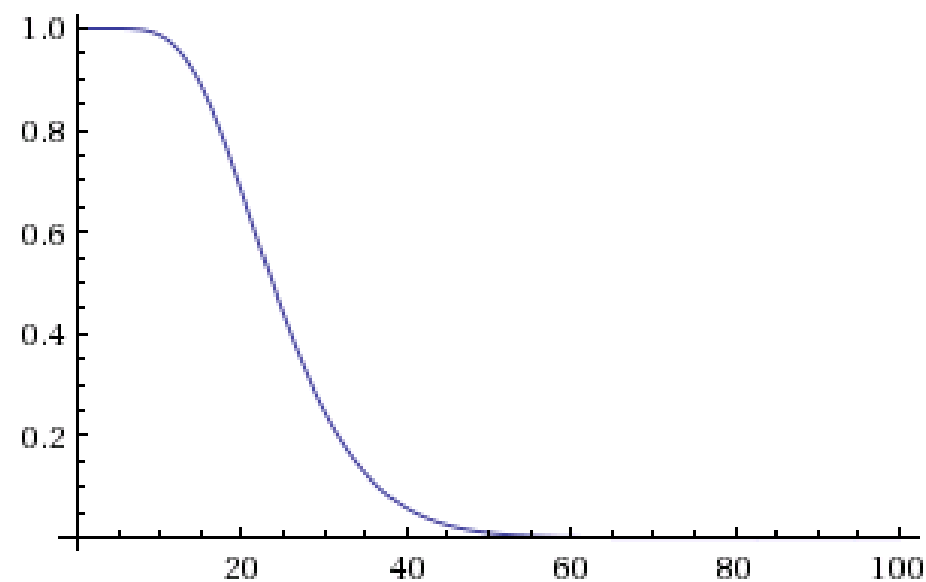


Fix $d' = 100, l = 10, k = 10$

Input interpretation:

plot	$1 - \left(1 - \left(1 - \frac{x}{100}\right)^{10}\right)^{10}$	$x = 0 \text{ to } 100$
------	---	-------------------------

Plot:



ϵ -Nearest Neighbor Search

Explore r from $r_{min}, (1 + \epsilon)r_{min}, (1 + \epsilon)^2 r_{min}, \dots, r_{max}$

- (r, ϵ) -Neighbor
 - whether there exists a point p within distance r of q , if yes, return p' within distance $(1 + \epsilon)r$ from q .
 - Or whether all points are at least $(1 + \epsilon)r$ away from q

Preprocessing: $O(dN + kN)$

- point transformation
- hash table initialization
- point storage

Query: $O(kl + dl)$

- check buckets $g_1(q), g_2(q), \dots, g_l(q)$
- report nearest of (say) first $2l$ points

Space: $O(kN)$

(r, ϵ) -neighbor by LSH

Success condition:

- if there exists p such that $d(p, q) \leq r$, then p, q collide with some bucket;
- the total number of collisions between q and far neighbors is $\leq 2l$

Setting $k = \log_{\frac{1}{p_2}} N$, $l = N^\rho$ guarantees the probability of success is $\geq \frac{1}{2} - \frac{1}{e}$

- preprocessing time/space: $O(N^{1+\rho})$
- query time: $O(dN^\rho) = O(dN^{\frac{1}{c}})$, sublinear
- external memory: $k = \log_{\frac{1}{p_2}} \frac{N}{B}$, $l = (\frac{N}{B})^\rho$, $O(N^\rho)$ I/Os