Similarity Search

(Part 2)

OUTLINE

1. Introduction to similarity search (Part 1)
2. Similarity search in low dimensions (Part 1)
   - kd-trees
   - Curse of dimensionality
3. Similarity search in high dimensions
   - Approximate Near Neighbor (ANN) search
   - Locality Sensitive Hashing (LSH)

Similarity search in high dimensions

r-NNS in high dimensions

- The curse of dimensionality can be tackled resorting to approximate, randomized approaches.
- An approximate version of r-NNS might consider as a valid output any point that is not too far from the ball centered in the query q and with radius r.
- A randomized version of r-NNS might return:
  - A false positive: a point at distance $> r$ from q. This can be easily detected.
  - A false negative: a claim that there are no r-near neighbors, while some exist.

The goal is to minimize the false positive/negative rates.

(c, r)-Approximate Near Neighbor Search

Definition: (c, r)-Approximate Near Neighbor Search ((c, r)-ANNS)

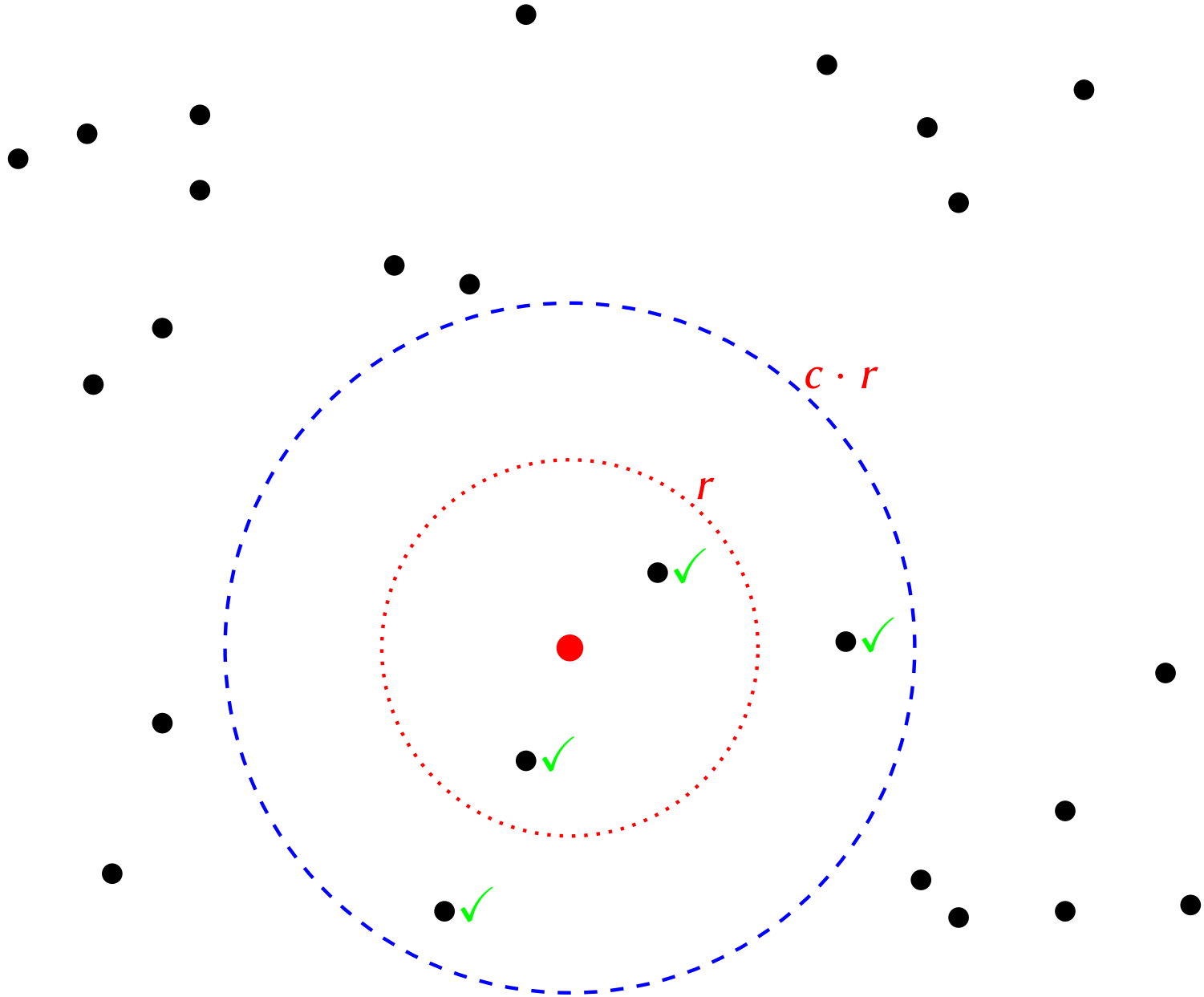
Given a set P of n points from the metric space (M, d), construct a data structure that, given a query point $q \in M$ and a distance threshold $r > 0$, provides the following answer for a certain constant $c \geq 1$:

- If there are points in $B_r(q) \cap P$, it returns a point $p \in P$ with $d(p, q) \leq c \cdot r$;
- If there are no points in $B_r(q) \cap P$ it may either a point $p \in P$ with $d(p, q) \leq c \cdot r$, if one exists, or `null`.

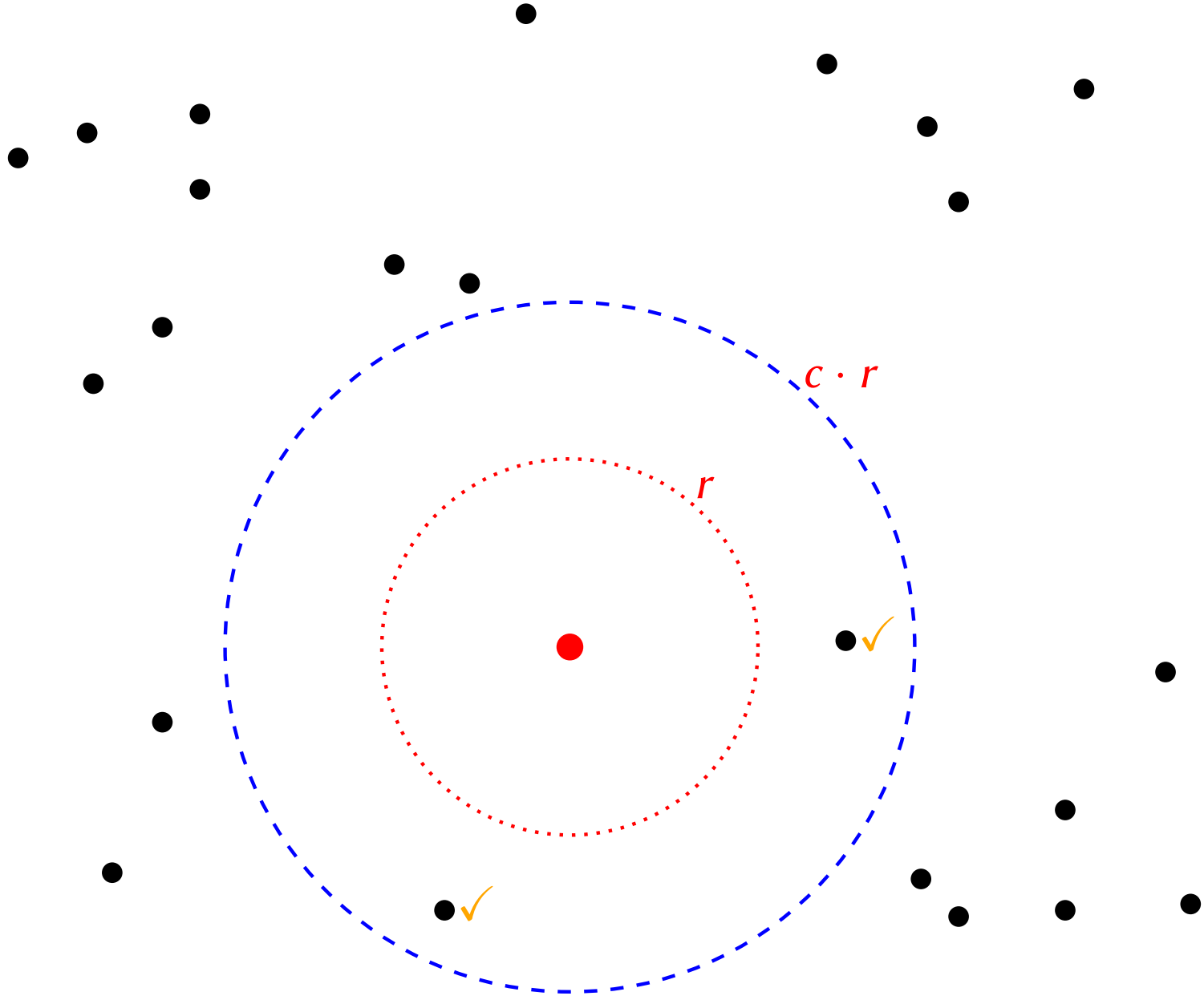
  Remark: *In this case, a* `null` *answer is always acceptable, even if a point $p \in P$ with $d(p, q) \leq c \cdot r$ exists.*

Observation: in all cases, the data structure never returns a point far away from the query point q (i.e., at distance $> c \cdot r$).

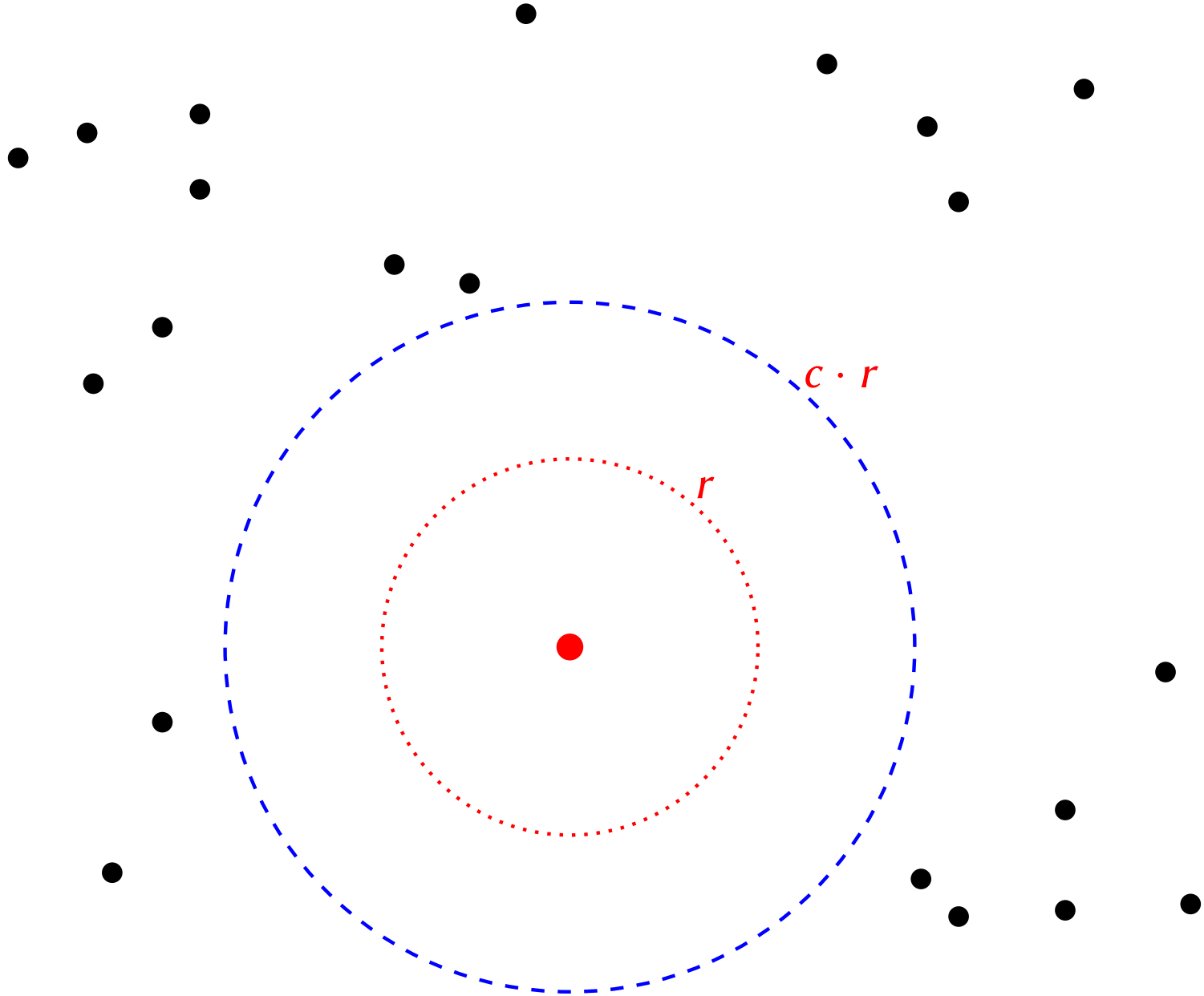
(c, r)-ANNS: example with near points



(c, r)-ANNS: example with no near points



(c, r)-ANNS: example with only far points



ANNS with Locality Sensitive Hashing

We present a solution for (c, r)-ANN which is based on *Locality Sensitive Hashing* (LSH), a popular tool for similarity searching in high dimensions.

Main ideas:

- The entire space is partitioned into regions through a hash function h randomly extracted from a suitable family $\mathcal{H}$.
- The partitioning ensures that: (1) two near points are likely to be mapped by h to the same region; (2) two far points are likely to be mapped by h to different regions;
- The neighbors of the query point q are searched for in the region identified by $h(q)$.

Observation: standard hashing aims at minimizing the collision probability for any pair of elements, whereas LSH makes the collision probability positivley related to the similarity between elements.

Definition of LSH

Consider a metric space (M, d) and a family of hash functions

$$\mathcal{H} = \{h : M \to S\},$$

where S is a given domain (e.g., indices in some range $[0, t]$). For any two points $p, q \in M$ denote by

$$\Pr_{h \in \mathcal{H}}[h(p) = h(q)]$$

the probability that a hash function h extracted from $\mathcal{H}$ uniformly at random maps p and q to the same value.

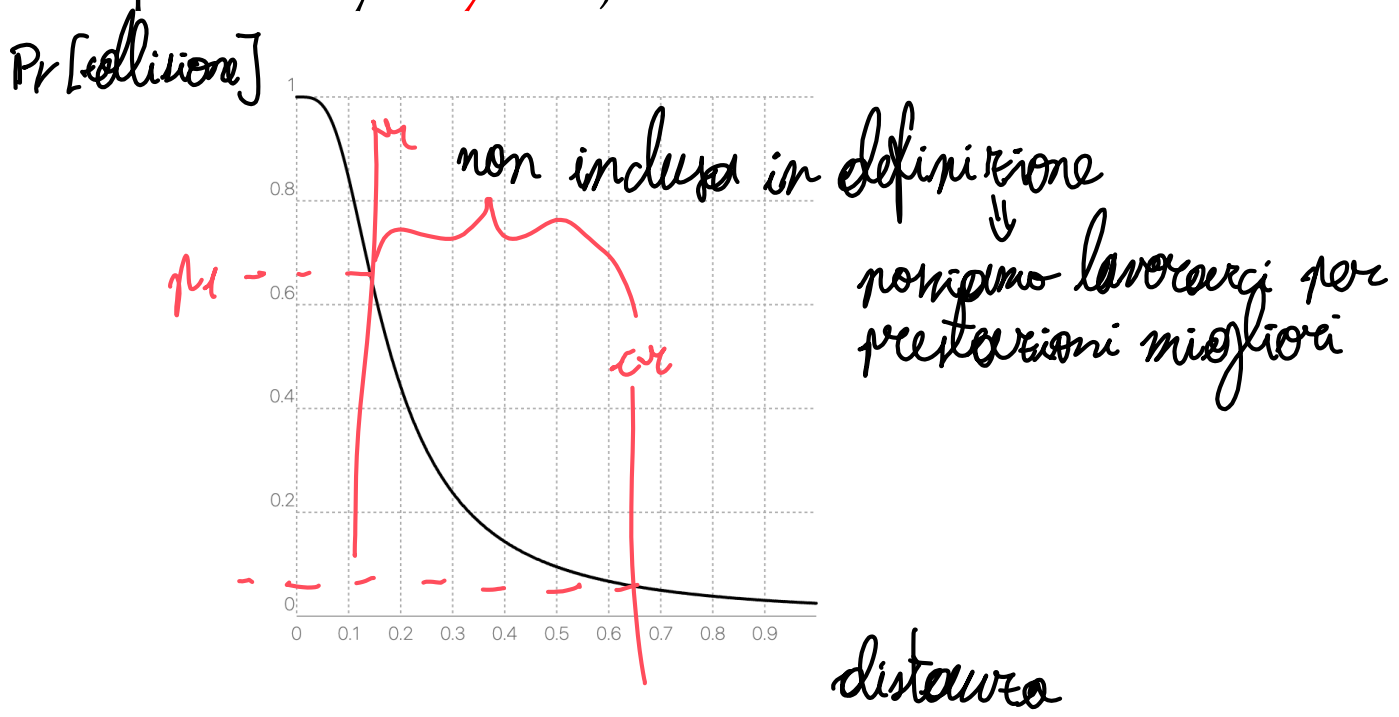
Definition: (p_1, p_2, c, r)-Locality Sensitive Hashing

Given parameters $p_1, p_2 \in [0, 1]$ with $p_1 > p_2$, $c > 1$ and $r > 0$, we say that $\mathcal{H}$ is (p_1, p_2, c, r)-locality sensitive if for any $p, q \in M$:

- $\Pr_{h \in \mathcal{H}}[h(p) = h(q)] \geq p_1$ if $d(p, q) \leq r$;
- $\Pr_{h \in \mathcal{H}}[h(p) = h(q)] \leq p_2$ if $d(p, q) > c \cdot r$.

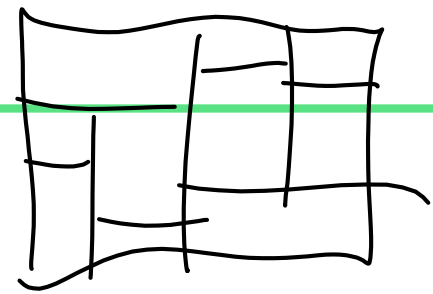
Comments of LSH

- The collision probability for points with distance in $(r, cr]$ is not specified.
- Usually, the collision probability between two points p, q can be expressed as a monotonically decreasing function of $d(p, q)$.
- Example: given a pair of points with (normalized) Hamming distance r, there exists an LSH with the following collision probability (r on x-axis, collision probability on y-axis):



LSH for (c, r)-ANNS

A (p_1, p_2, c, r)-LSH $\mathcal{H}$ can be used for solving (c, r)-ANNS on the input set P, as follows:



Construction

- Randomly select h from $\mathcal{H}$.
- Construct a hash table T with points in P using h: let $T[j]$ be the (potentially empty) bucket containing all points in P with hash value j (i.e., $T[j] = \{p \in P : h(p) = j\}$).

Query

- For a given query q, scan $T[h(q)]$ until a $c \cdot r$-near point p is found and return it; if there is no such point, return `null`.

LSH for (c, r)-ANNS: example

Input: 8 boolean vectors of 9 dimensions; Hamming distance; $r = 2$

Construction

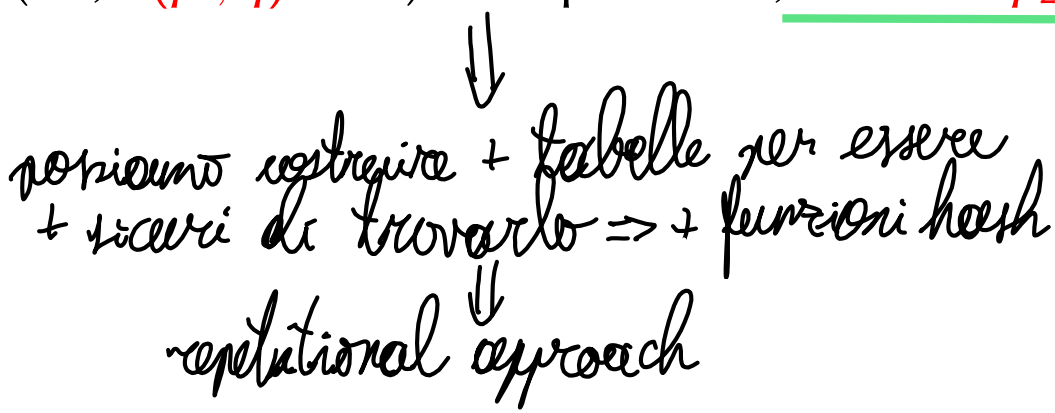
Query

ANNS with LSH

Consider a query point q. Which points do we expect to see in $T[h(q)]$?

- A near point p (i.e., $d(p, q) \leq r$) will be in $T[h(q)]$ with probability *at least* p_1, but it might also end up in a different bucket.
- A far point p' (i.e., $d(p', q) > c \cdot r$) might end up in $T[h(q)]$, but only with probability *at most* p_2.

Worst case scenario: there exists only one near point $p \in P$ ($d(q, p) \leq r$) and $n - 1$ far points p' (i.e., $d(p', q) > cr$). In expectation, at most np_2 far points collide with q.



ANNS with LSH: performance

We make the following reasonable assumptions:

- $M = \mathbb{R}^D$ for some (large) D
- Each point of M requires $O(D)$ words to be stored.
- A hash value $h(p)$ for some $p \in M$ can be computed in $O(D)$ time.
- The set S of possible hash values is $\ll n$, where n is the number of points of P.

Theorem

Let P be a set of n points in a metric space (M, d), and let $\mathcal{H}$ be a (p_1, p_2, c, r)-locality sensitive family of hash functions. Using $\mathcal{H}$ and the above approach to (c, r)-ANNS, a query is answered successfully with probability $\geq p_1$. Moreover the following performance is obtained:

- **Construction time:** $O(Dn)$
- **Space:** $O(Dn)$
- **Query time:** $O(Dnp_2)$ *in expectation.*

Improving the data structure

Ideally, LSH should have p_1 close to 1 and p_2 close to 0. However, a famili $\mathcal{H}$ might not provide this type of guarantees.

We now outline two improvements to respectively increase the collision probability of near points and decrease the collision probability of far points, which, combined together, yield better LSH.

Improvement 1: increase collision probability of near points

Idea: repeat search $\ell > 1$ times using ℓ distinct hash tables based on independent hash functions chosen uniformly at random from a (p_1, p_2, c, r)-locality sensitive family $\mathcal{H}$.

- If we set $\ell = p_1^{-1}$, then a near point p collides with the query point q in at least one table in expectation.
- If we set $\ell = \Theta\left(p_1^{-1} \log n\right)$, then a near point p collides with q in at least one table with probability at least $1 - 1/n$.
- In each table, the expected number of collisions of q with far points in the same bucket is at most np_2. $\Rightarrow \ln p_2$ in aspettazione totale

This technique is called repetition or OR construction.

Improvement 2: decrease collision probability of far points

Idea: use a family $\mathcal{G}$ of hash functions obtained by concatenating $k \geq 1$ independent hash functions chosen uniformly at random from a (p_1, p_2, c, r)-locality sensitive family $\mathcal{H}$. Namely,

$$\mathcal{G} = \{g \in \mathcal{H}^k\} = \{g(p) = (h_1(p), \ldots, h_k(p)), \text{ with } h_i \in \mathcal{H}\}$$

It is immediate to establish that for a random $g \in \mathcal{G}$ and any two points p, q with $p \neq q$,

- $\Pr[g(p) = g(q)] \geq p_1^k$, if $d(p, q) \leq r$;
- $\Pr[g(p) = g(q)] \leq p_2^k$, if $d(p, q) > c \cdot r$.

This technique is called concatenation or AND construction.

Improvement 2: decrease collision probability of far points

Suppose that we set

$$k = \log_{1/p_2} n = \frac{\log_2 n}{\log_2(1/p_2)}.$$

Then, for a random $g \in \mathcal{G}$ and any two points p, q with $p \neq q$,

- $\Pr[g(p) = g(q)] \leq p_2^k = 1/n$, if $d(p, q) > c \cdot r$.
- $\Pr[g(p) = g(q)] \geq p_1^k = 1/n^\rho$, if $d(p, q) \leq r$;

where $\rho = \log_2 p_1 / \log_2 p_2$. Equation $p_1^k = 1/n^\rho$ is justified as follows:

$$p_1^k = p_1^{\frac{\log_2 n}{\log_2(1/p_2)}} = p_1^{\frac{\log_2(1/n)}{\log_2 p_2}} = p_1^{\frac{\log_2(1/n)}{\log_2 p_1}\frac{\log_2 p_1}{\log_2 p_2}} = p_1^{\log_{p_1}(1/n)\frac{\log_2 p_1}{\log_2 p_2}} = (1/n)^{\frac{\log_2 p_1}{\log_2 p_2}} = 1/n^\rho.$$

Observations:

- Since we assume $p_1 > p_2$, we have that $\rho \in (0, 1)$. This quantity is usually employed for evaluating the quality of an LSH.
- With the above choice of k the expected number of collisions of a query point q with far points in the same bucket is at most 1.

LSH and (c, r)-ANNS: a general schema

We merge the two improvements to get the following schema. Let $\mathcal{H}$ be a (p_1, p_2, c, r)-locality sensitive family of hash functions, and let k and ℓ be two values that will be set later.

Construction

- Construct ℓ hash functions $g_1 \dots g_\ell$: each g_i consist of the concatenation of k hash functions randomly and independently selected from $\mathcal{H}$.
- For each g_i, construct a hash table T_i of points in P using g_i. We let $T_i[j]$ be the (potentially empty) bucket containing all points in P with hash value j when using g_i.

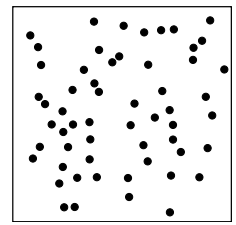
spazio totale: $O(n\ell + nd)$

Query q

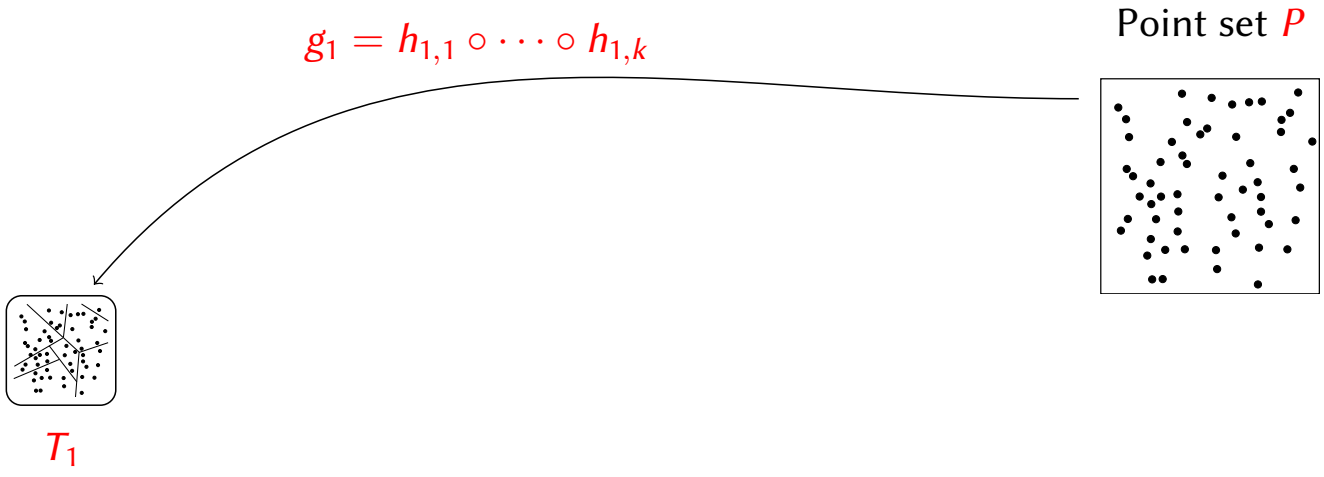
- Scan $T_1[g_1(q)], \dots T_\ell[g_\ell(q)]$ until a $c \cdot r$-near point p is found and return it; if no such point is found, return `null`.

LSH and (c, r)-ANNS: example

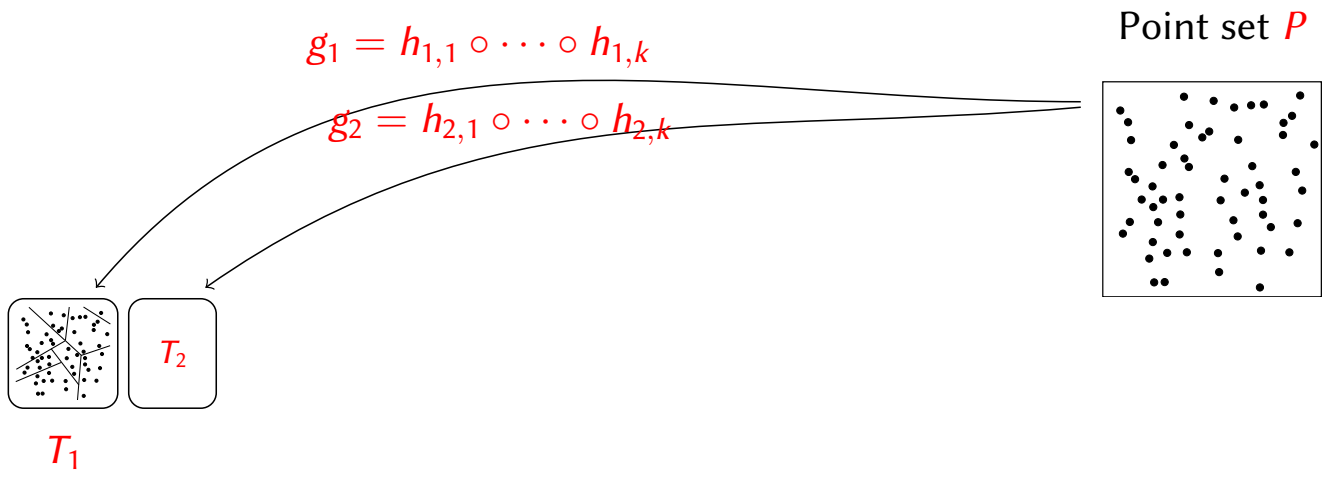
Point set P



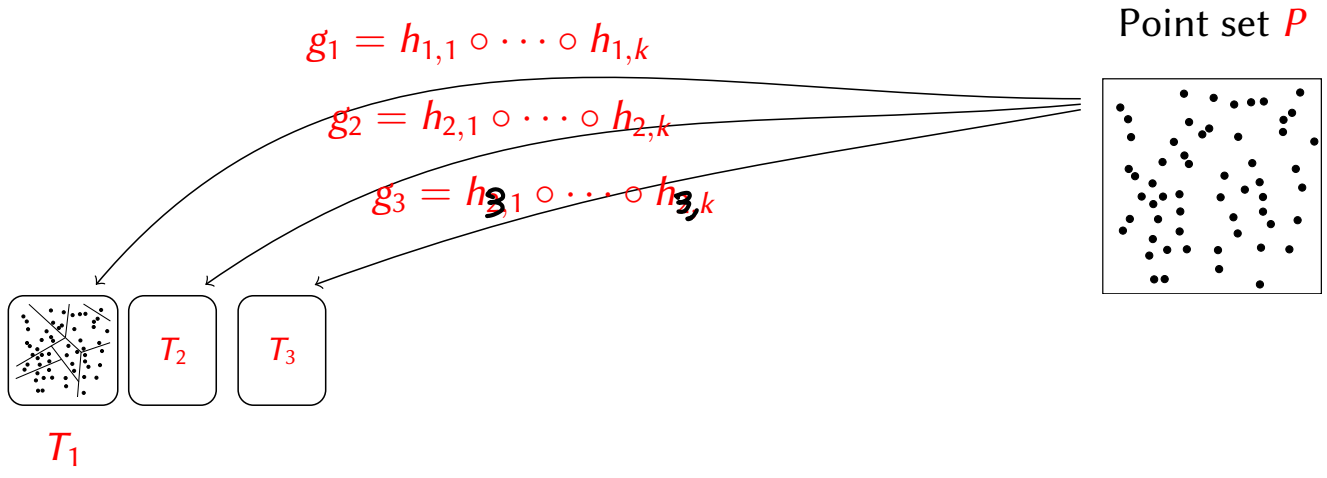
LSH and (c, r)-ANNS: example



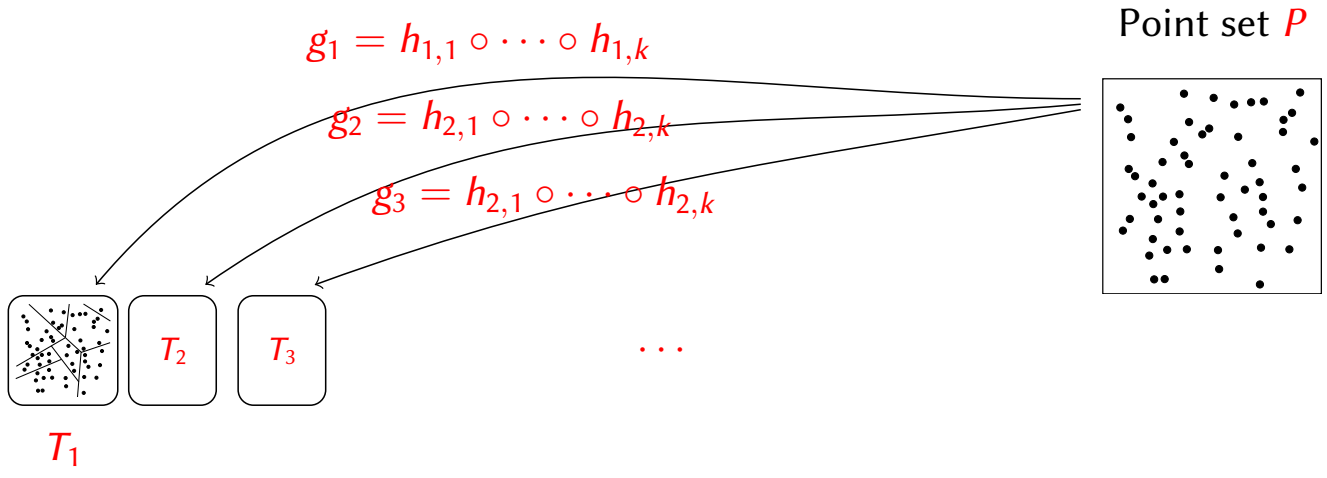
LSH and (c, r)-ANNS: example



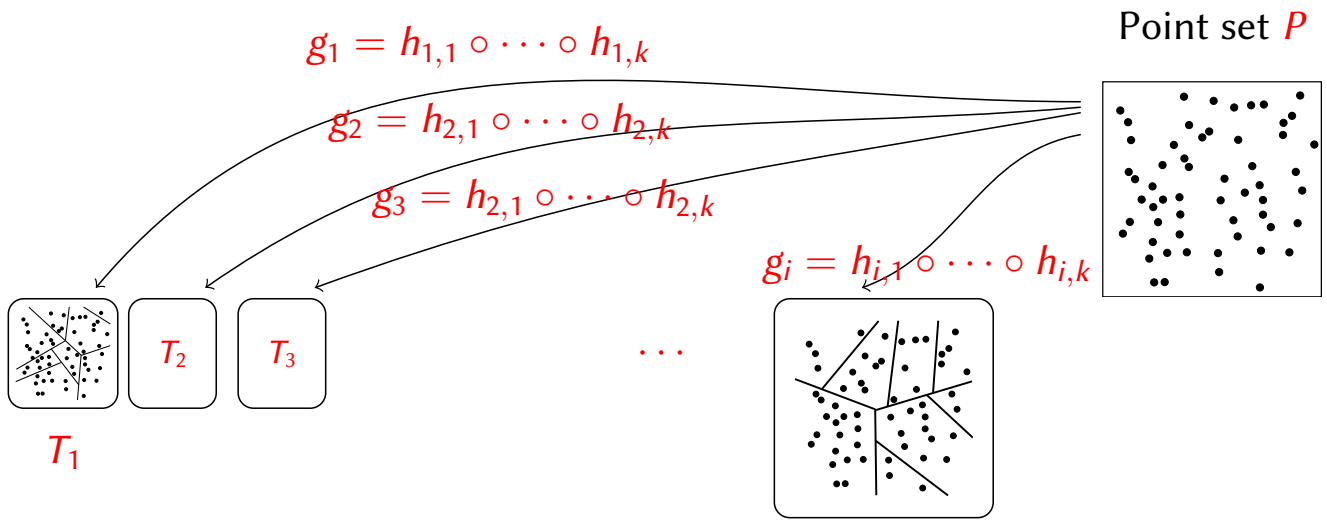
LSH and (c, r)-ANNS: example



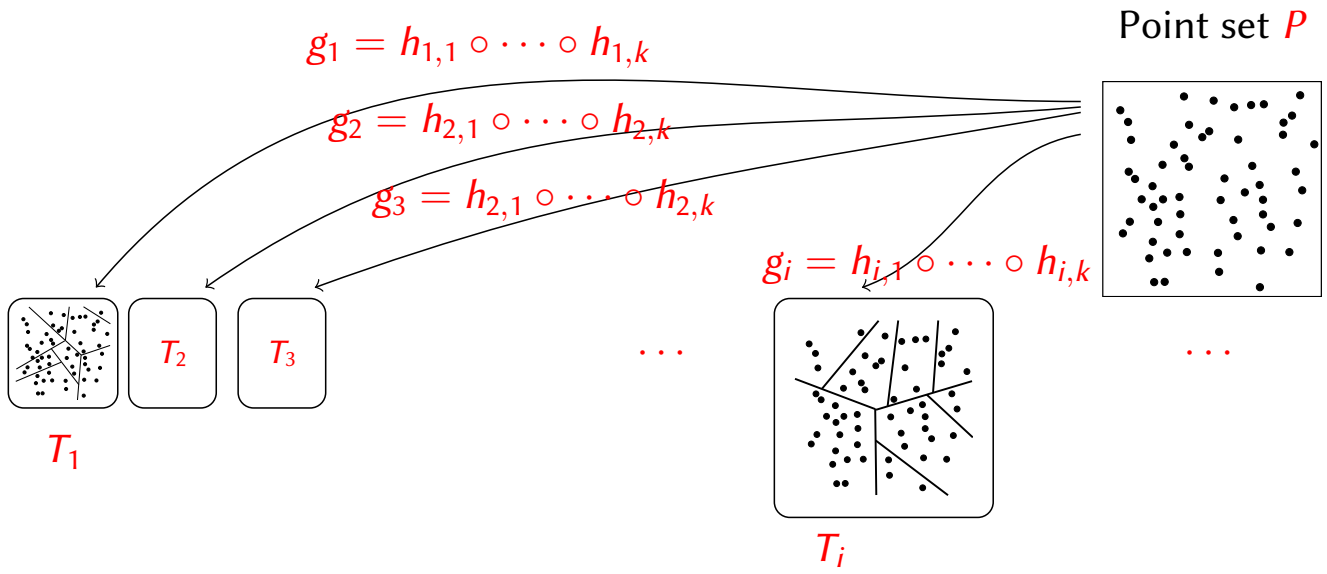
LSH and (c, r)-ANNS: example



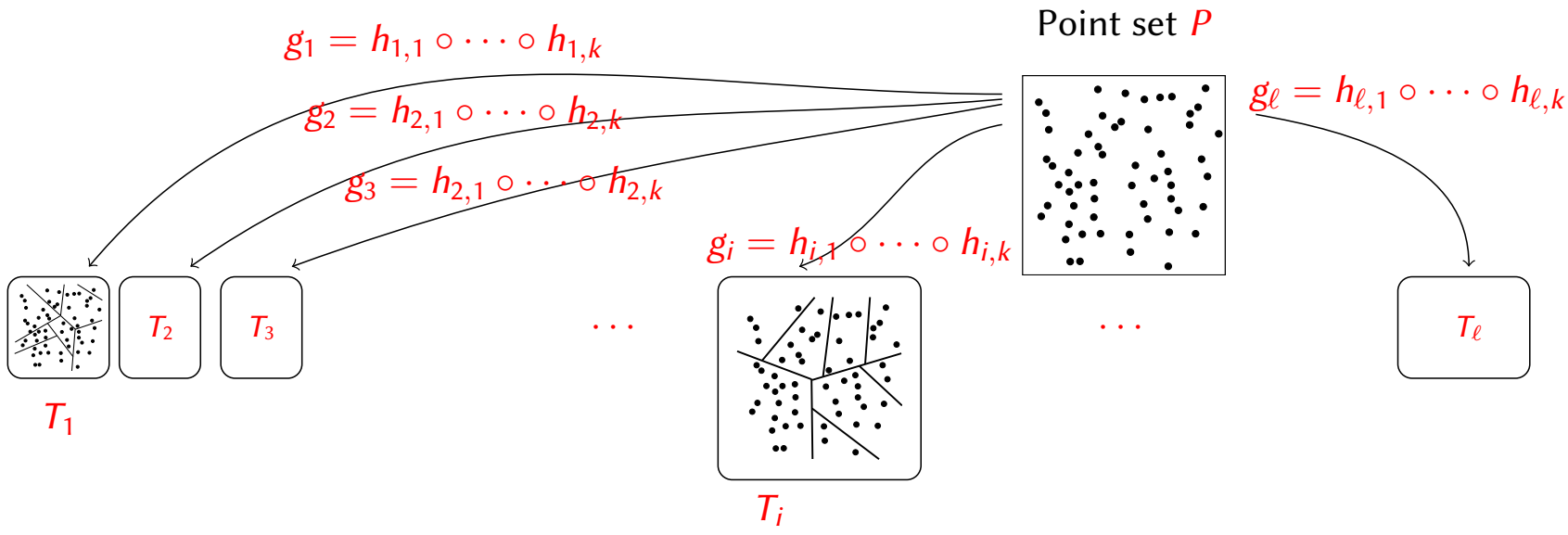
LSH and (c, r)-ANNS: example



LSH and (c, r)-ANNS: example



LSH and (c, r)-ANNS: example



LSH and (c, r)-ANNS: performance

Theorem

Let P be a set of n points in a metric space (M, d), and let $\mathcal{H}$ be a (p_1, p_2, c, r)-locality sensitive family of hash functions. Fix

$$k = \log_{1/p_2} n \quad \text{and}$$
$$\ell = 2p_1^{-k} = 2n^{\rho},$$

dovrebbero essere interi nella pratica

where $\rho = \log_2 p_1 / \log_2 p_2$. Using the above approach to (c, r)-ANNS, a query is answered successfully with probability $\geq 1/2$. Moreover the following performance is obtained:

- **Construction time:** $O\left(Dn^{1+\rho} \log_{1/p_2} n\right)$ (# labelle)
- **Space:** $O\left(Dn + n^{1+\rho} \log_{1/p_2} n\right)$ $\Rightarrow \rho \simeq 0$: vicino a 1
- **Query time:** $O\left(Dn^{\rho} \log_{1/p_2} n\right)$ *in expectation.*

$\rho \simeq 0$: sublineare

Observations

Observations

Observations

LSH for Hamming and Euclidean distances

LSH for Hamming distance: bit sampling

- Let p be a D-dimensional Boolean vector and let p_i its i-th coordinate. Consider the following family of hash functions:

$$\mathcal{H}_H = \{h_i(p) = p_i\}. \Rightarrow |\mathcal{H}_H| = D$$

- Given two points p, q, the probability of collision is

$$\Pr_{h \in \mathcal{H}_H}[h(p) = h(q)] = 1 - \frac{d_H(p,q)}{D},$$

where $d_H()$ denotes the Hamming distance.

bit 0 1 2 3 4

$p = (0,1,1,0,1)$, $q = (0,1,1,1,0)$

prendo $h_0 \Rightarrow h_0(p) = h_0(q)$

// $h_3 \Rightarrow h_3(p) \neq h_3(q)$

LSH for Hamming distance: bit sampling

For any two points p, q we have:

- $\Pr_{h \in \mathcal{H}_H}[h(p) = h(q)] \geq 1 - r/D$, if $d_H(p, q) \leq r$;
- $\Pr_{h \in \mathcal{H}_H}[h(p) = h(q)] \leq 1 - c \cdot r/D$, if $d_H(p, q) \geq c \cdot r$;

Therefore $\mathcal{H}_H$ is $(1 - r/D, 1 - cr/D, c, r)$-locality sensitive.

The ρ factor for bit sampling is:

$$\rho = \frac{\log_2 p_1}{\log_2 p_2} = \frac{\log_2(1 - r/D)}{\log_2(1 - cr/D)} \sim \frac{r/D}{cr/D} = 1/c.$$

$\Downarrow$ serie di Taylor

LSH for Hamming distance: concatenating bit sampling

Concatenating k bit sampling LSH consists in randomly selecting k indexes, and yields the following collision probabilities:

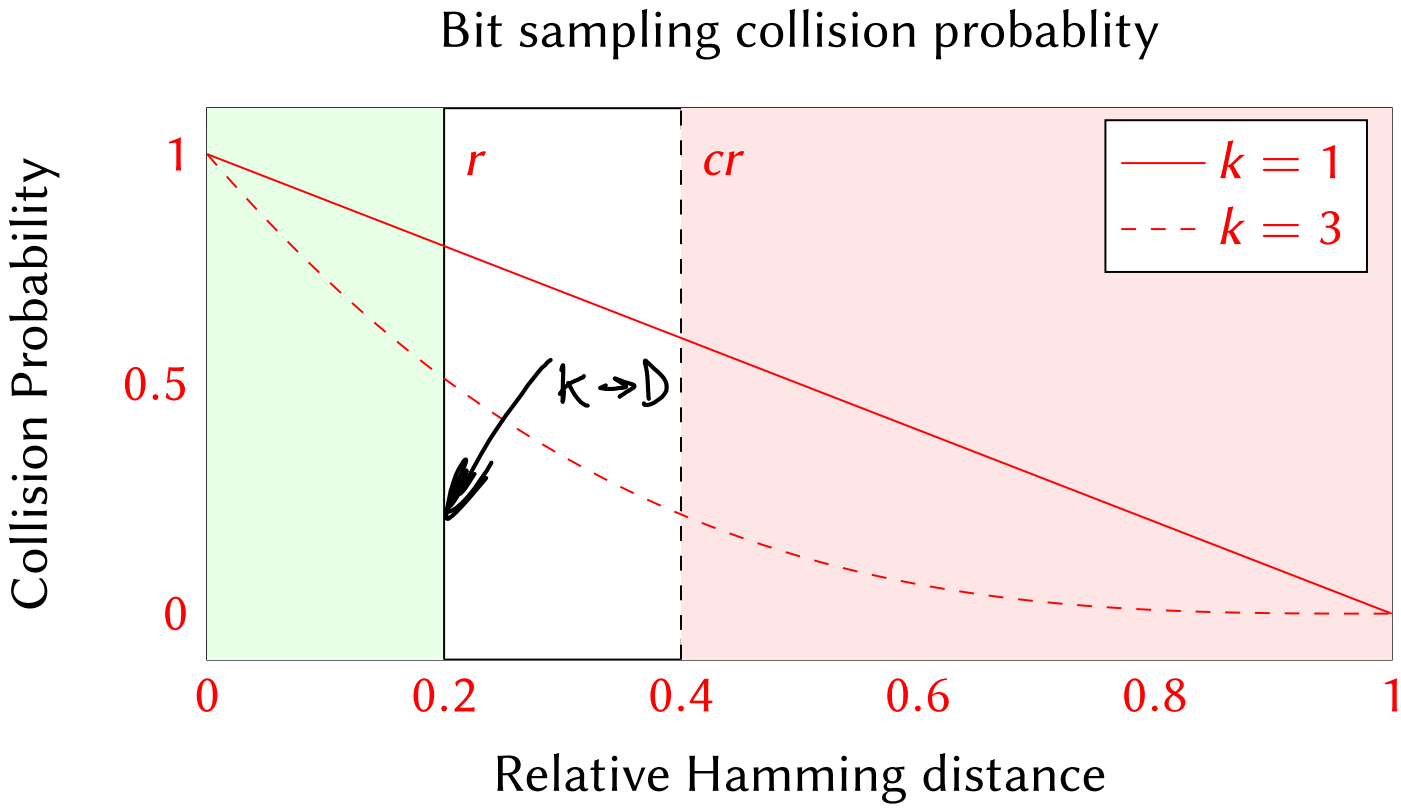
- $\Pr_{h \in \mathcal{H}_H}[h(p) = h(q)] \geq (1 - r/D)^k$, if $d_H(p, q) \leq r$;
- $\Pr_{h \in \mathcal{H}_H}[h(p) = h(q)] \leq (1 - c \cdot r/D)^k$, if $d_H(p, q) \geq c \cdot r$;

The concatenation does not change the ρ value.

Bitsampling: Project to random subset of dimensions.

x	001010010<u>1</u>0	y	1<u>01</u>011000<u>1</u>0
$h(x)$	011	$h(y)$	011

LSH for Hamming distance: concatenating bit sampling



LSH for Euclidean distance: random projection

Consider points in $\mathbb{R}^D$. For a fixed value $w > 0$ define the family $\mathcal{H}_E$ of hash functions

$$h_{a,w}(p) = \left\lceil \frac{\langle a, p \rangle + b}{w} \right\rceil, \quad (\langle , \rangle \text{ denotes the inner product})$$

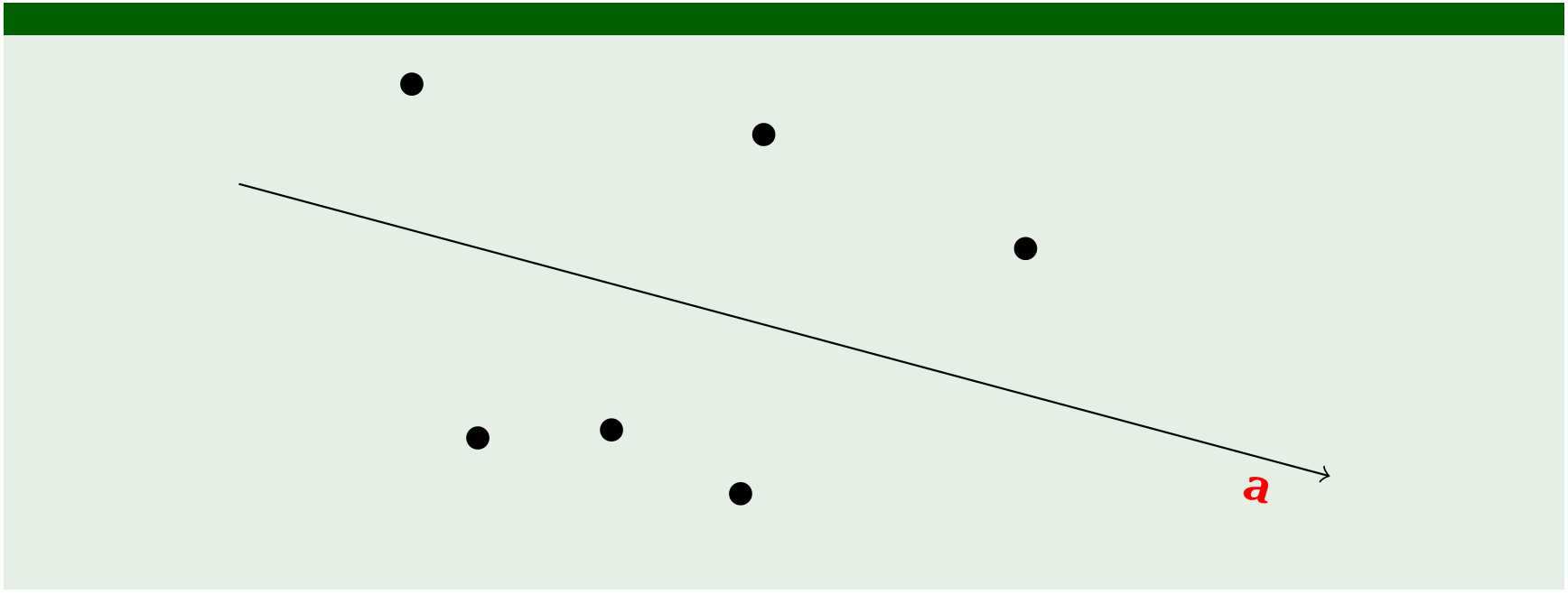
where a and b are sampled as: $a \sim \mathcal{N}^D(0, 1)$ (normal distribution), $b \sim U[0, w]$ (uniform distribution).

It can be shown that $\mathcal{H}_E$ is (p_1, p_2, c, r)-locality sensitive with $\rho = O(1/c)$. There are better LSH families of hash functions for Euclidean distance with $\rho = O(1/c^2)$.

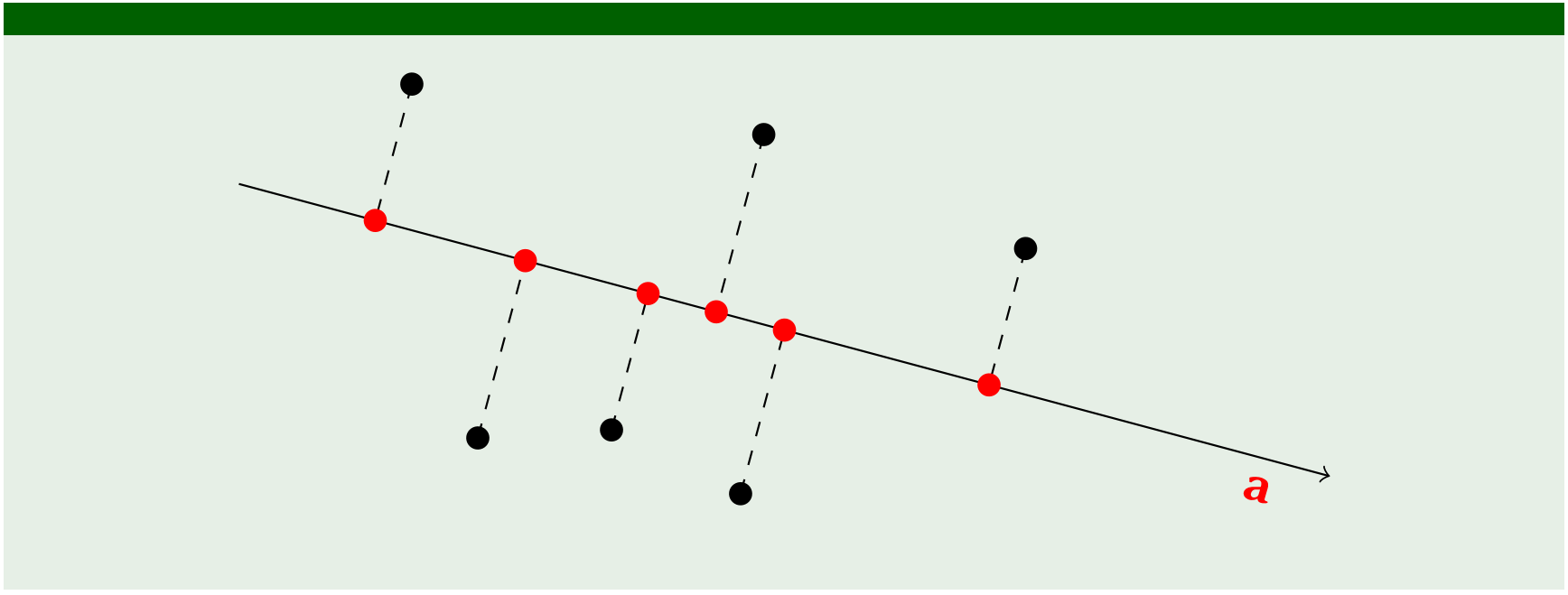
query time $\sim O(n^{1/c^2})$ for $\rho = O(1/c^2)$

$O(n^{1/c})$ // // $O(1/c)$

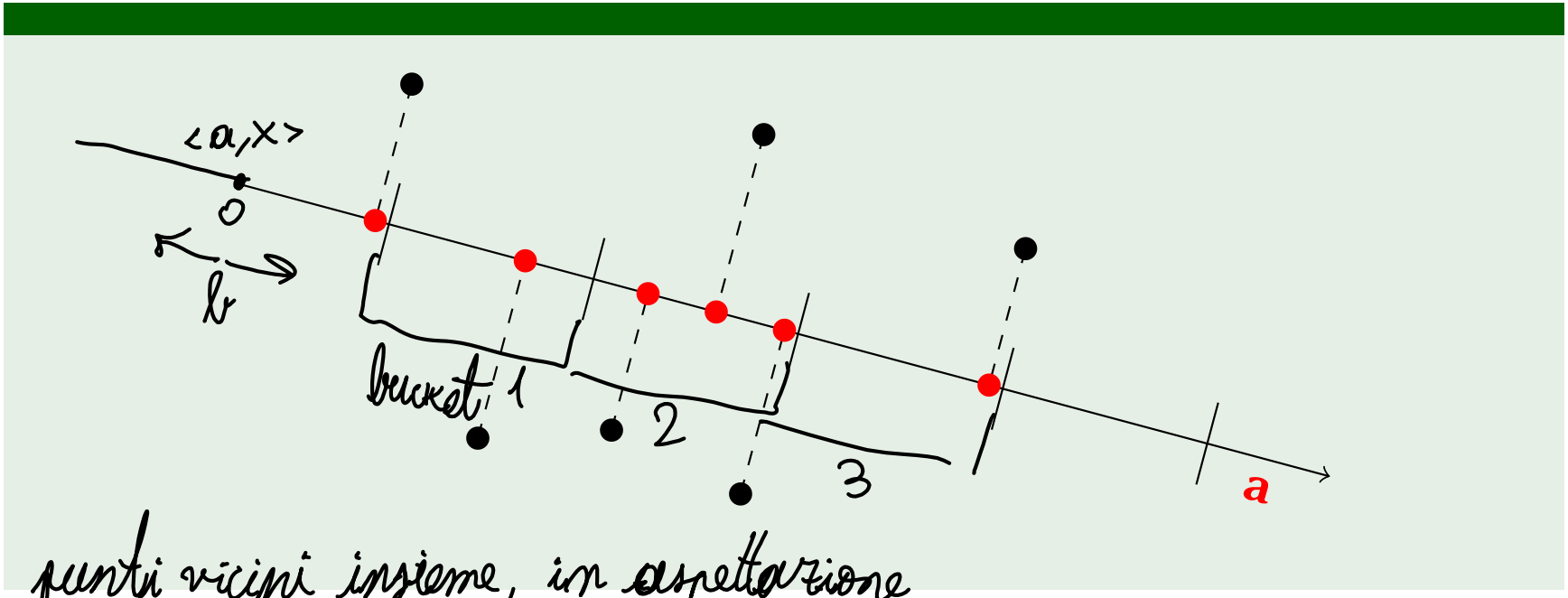
LSH for Euclidean distance: example



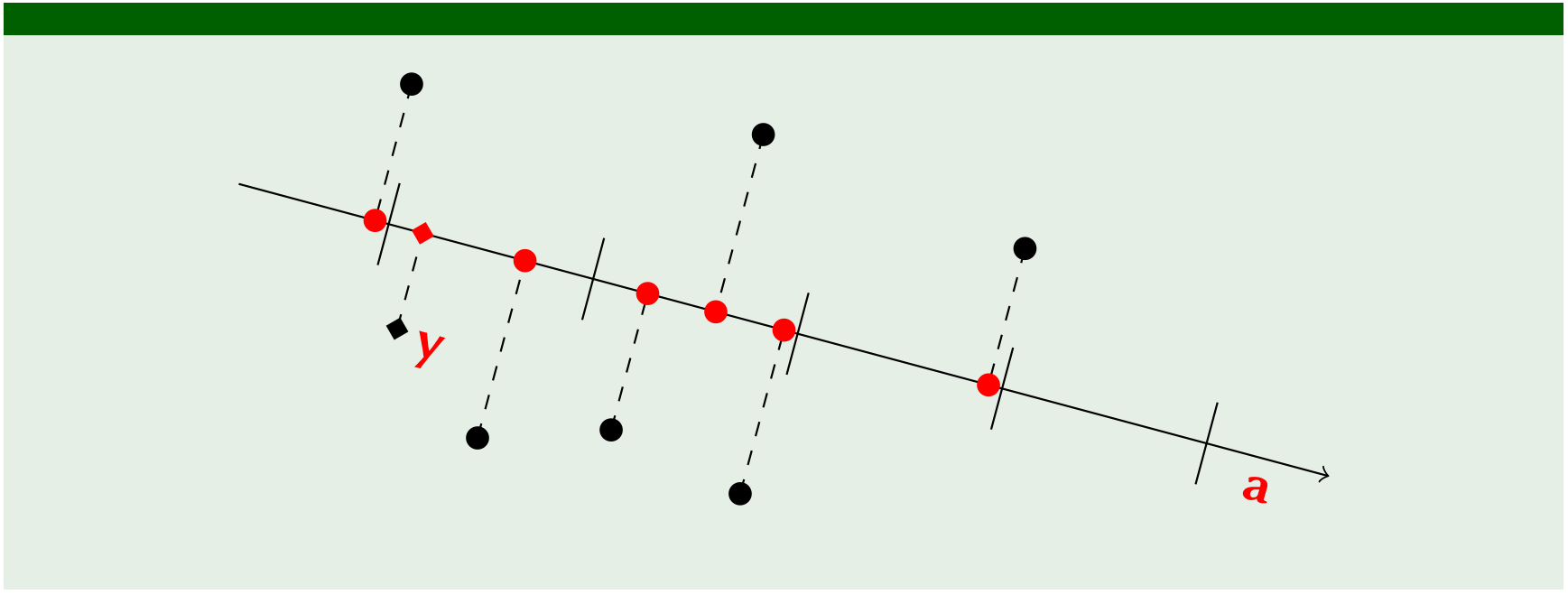
LSH for Euclidean distance: example



LSH for Euclidean distance: example



LSH for Euclidean distance: example



Summary

- Similarity search: r-NNS and RR problems.
- k-d tree for similarity search in low dimensions.
- Curse of dimensionality for similarity search.
- (c, r)-ANNS problem.
- LSH approach to the (c, r)-ANNS problem.
  - Definition of $(p_1, p_2, c, r))$-locality sensitive hash functions.
  - Solving (c, r)-ANNS through (p_1, p_2, c, r)-locality sensitive hash functions.
  - Improving collision probabilities with repetition and concatenation, and their combination.
  - $(p_1, p_2, c, r))$-locality sensitive hash functions for Hamming and Euclidean distances (bit sampling and random projection).

References

LRU14 J. Leskovec, A. Rajaraman and J. Ullman. Mining Massive Datasets. Cambridge University Press, 2014. Section 3.6.

BCKO08 Mark de Berg, Otfried Cheong, Marc van Kreveld, and Mark Overmars. Computational Geometry: Algorithms and Applications (3rd ed. ed.). Springer-Verlag, 2008. Section 5.3

AI08 Alexandr Andoni and Piotr Indyk. 2008. Near-optimal hashing algorithms for approximate nearest neighbor in high dimensions. Commun. ACM 51, 1 , 2008.