Advanced topics on Algorithms

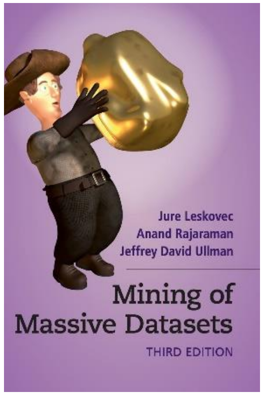
Luciano Gualà
www.mat.uniroma2.it/~guala/

Algorithms for Big Data
Episode IV

Finding similar items

Locality-Sensitive Hashing

reference
(Chapter 3)



The problem

Given N items, find pairs of them whose similarity is above a give threshold

main challenge: N is huge and a $\Theta(N^2)$-time solution is infeasible

additional challenge: high multidimensionality of each item
(obvious representation does not fit in main memory)

Finding similar documents:

- plagiarism: no simple process of comparing documents character by character will detect a sophisticated plagiarism;
- mirror pages: duplicated pages quite similar but rarely identical. Do not show them as a result of a search engine query;
- articles from the same source: essentially same article published in different web sites;
- documents about the same topic: content-based notion of similarity.

Matching fingerprints: find duplicates in a database.

Entity resolution: find different data records that refer to the same real-world entity.

Finding similar customers: detecting customers whose set of purchased products are similar.

keep this application in mind for the sake of concreteness

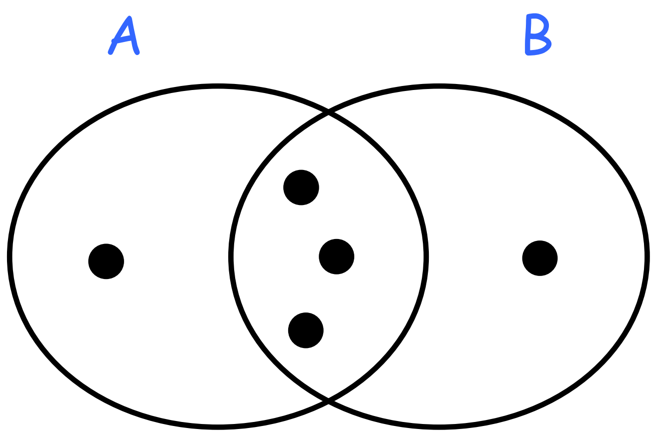
Each item is a set of elements of a given universe

e.g., each item is a customer, and the set represents the products he/she bought

when two sets A and B are similar?

Jaccard Similarity:

$$JS(A,B) = \frac{|A \cap B|}{|A \cup B|}$$



$JS(A,B)=3/5$

goal: find pairs of sets whose JS is at least a give threshold.

Two ingredients:

- a randomized representation of items that preserves similarity (it depends on the specific similarity measure you deal with)
- clever use of hash functions/tables allowing to map similar items to the same slot/bucket (locality-sensitive hashing, banding technique)

Matrix representation of sets

sets/customers

elements/products

	S_1	S_2	S_3	S_4
a	1	0	1	0
b	1	0	0	1
c	0	1	0	1
d	0	1	0	1
e	0	1	0	1
f	1	0	1	0
g	1	0	1	0

- convenient to "visualize" the problem
- not the actual way sets are maintained in memory (matrix usually sparse)

JS preserving representation for sets

minhashing and signatures

Minhashing

choose a random permutation π of the matrix rows

a column S is represented as:

$h_\pi(S)$= first row index (according to π) in which S has a 1

π
4
2
1
3
6
7
5

S_1	S_2	S_3	S_4
1	0	1	0
1	0	0	1
0	1	0	1
0	1	0	1
0	1	0	1
1	0	1	0
1	0	1	0

2	1	4	1

Lemma

For any two columns S and S', $\Pr(h_\pi(S)= h_\pi(S')) = JS(S,S')$.

proof

let i be the first index according to π in which S has a 1 or S' has a 1.

i belongs to $S \cup S'$

π	S	S'
4	1	0
2	0	0
1	0	0
3	1	1
6	1	1
7	0	0
5	0	1

for uniformly random π,
$\Pr(\ i=\text{"specific element of } S \cup S'\text{"})=1/|S \cup S'|$

$h_\pi(S)= h_\pi(S')$ iff i belongs to $S \cap S'$

$\Rightarrow$

$\Pr(h_\pi(S)=h_\pi(S')) = |S \cap S'|/|S \cup S'|=JS(S,S')$.

Minhash signature

choose n random permutations $\pi_1,\ldots,\pi_n$ of the matrix rows

given S, $h_i(S)$= first row index (according to π_i) in which S has a 1

a column S is represented as a (column) vector $[h_1(S),\ldots, h_n(S)]$

minhash signature of S

π_1	π_2	π_3	S_1	S_2	S_3	S_4
4	2	3	1	0	1	0
2	3	4	1	0	0	1
1	7	7	0	1	0	1
3	6	2	0	1	0	1
6	1	6	0	1	0	1
7	5	1	1	0	1	0
5	4	5	1	0	1	0

2	1	4	1
2	1	2	1
1	2	1	2

Notice:

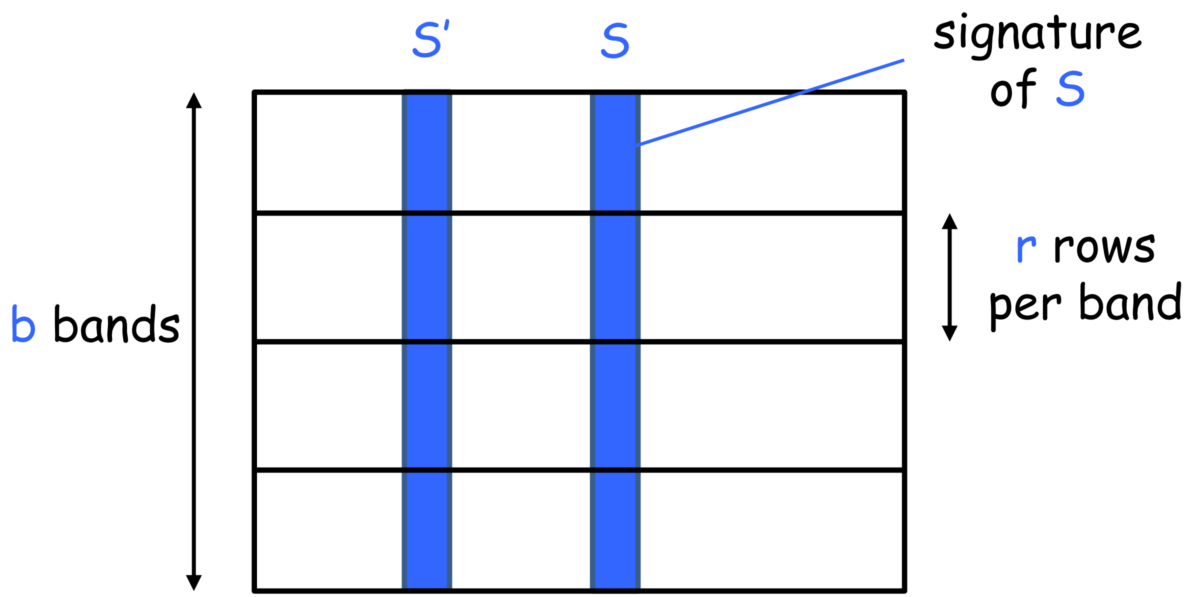
- usually much smaller representation (bunch of integers)
- expected fraction of minhash values where S,S' agree=$JS(S,S')$

Locality-Sensitive Hashing

banding technique

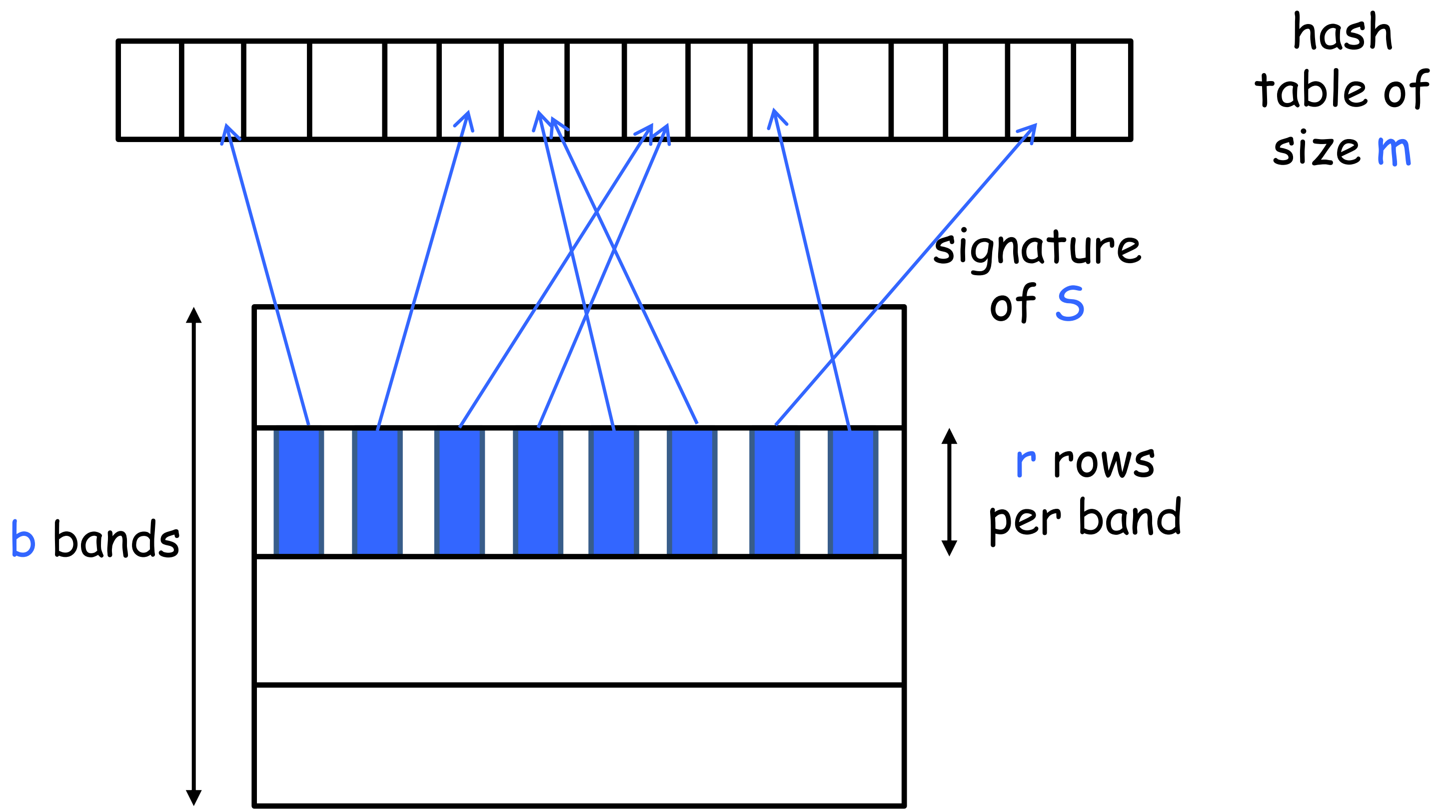
Idea:

- group the n minhash values into b bands of r rows each ($n = b \cdot r$)
- declare two columns candidate (to be similar) if they agree on ≥1 band
- to discover candidates: use the bands as keys for a hash table with the purpose to map columns agreeing on ≥1 band to the same slot of the table.



Idea:

- use the value of a band as key for a hash table of size m
- two columns with the same value for that band are mapped to the same slot
- also columns with different values for the band might be mapped to the same slot
  - choose m as large as possible to minimize accidental collisions



Analysis

assumption: two columns are mapped to the same slot iff they have the same band value

- simplifies the analysis
- almost met in practice if you choose m large enough and use a good enough hash function

fix two columns S and S' and let $s = JS(S, S')$

probability that the signatures disagree in at last one row of a particular band $\quad 1-s^r$

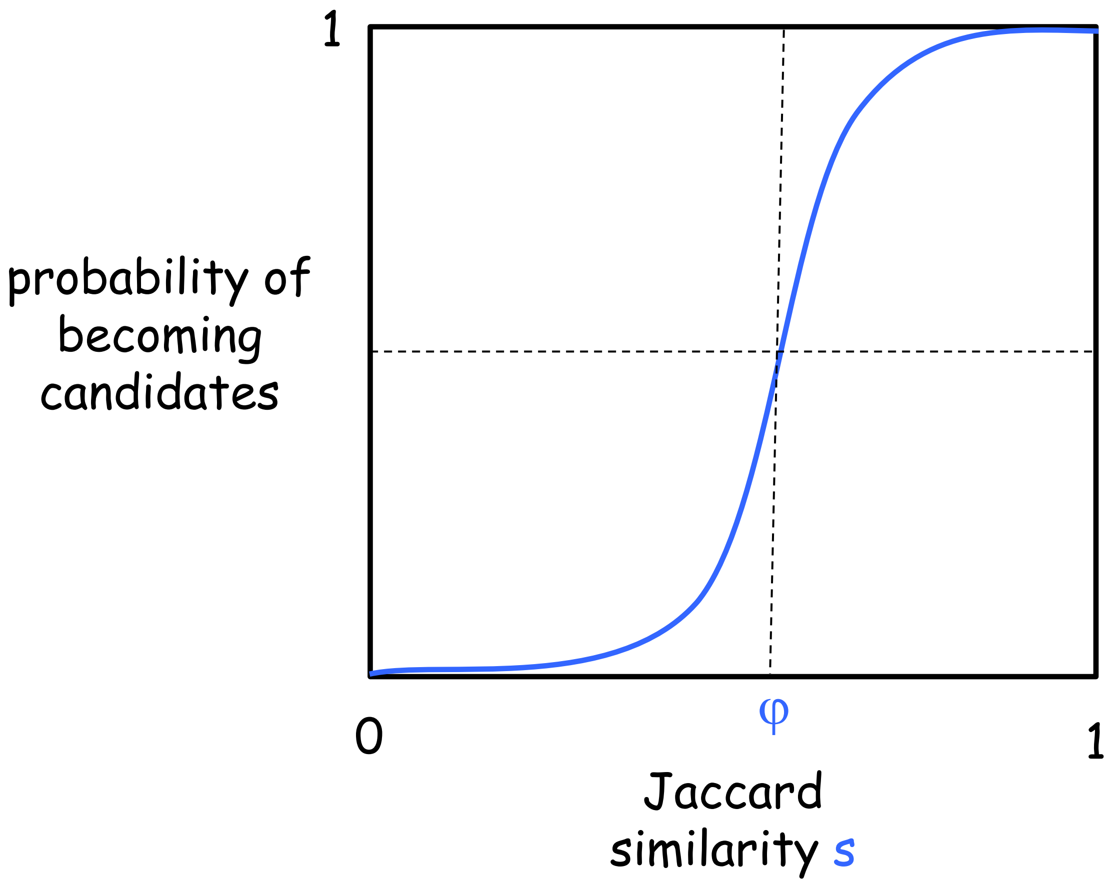
probability that the signatures disagree in at last one row of each of the bands $\quad (1-s^r)^b$

probability that the signatures agree in all rows of at last one band (and hence become S and S' candidates) $\quad 1-(1-s^r)^b$

$f(s) = 1-(1-s^r)^b$ is an S-curve

$b=20 \quad r=5$

$\varphi \approx 0.509$



s	f(s)
.2	.006
.3	.047
.4	.186
.5	.470
.6	.802
.7	.975
.8	.9996

threshold φ: value of s such that $f(s)=1/2 \quad \approx (1/b)^{1/r}$

some implementation tricks

notice: picking a random permutation of the k rows is time-consuming

idea: pick a (random) hash function $h:\{1,\ldots,k\} \longrightarrow \{1,\ldots,k\}$ instead

- h "permutes" row r to position $h(r)$ in the permuted order

notice: two rows can be mapped to the same slot/position

- not so important as long as k is large and not too many collisions

h_1	h_2	h_3	S_1	S_2	S_3	S_4
4	2	3	1	0	1	0
2	4	4	1	0	0	1
1	7	7	0	1	0	1
3	6	2	0	1	0	1
6	1	6	0	1	0	1
7	5	1	1	0	1	0
6	4	5	1	0	1	0

c

i				
	2	1	4	1
	2	1	2	1
	1	2	1	2

$$SIG[i,c] = \min_{\substack{r \text{ s.t.} \\ M[r,c]=1}} h_i(r)$$

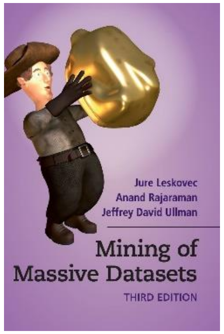
one-pass algorithm

1. SIG[i,c]=∞ for each i and c
2. for each row r
   1. compute $h_1(r),\ldots, h_n(r)$
   2. for each column c with M[r,c]=1
   3. for each i do
      SIG[i,c]=min{SIG[i,c], $h_i(r)$}

an additional trick:

- not compute $h_i(r)$ for all r
- divide the k rows into k/m groups of m rows (for some parameter m)
- compute $h_i(r)$ only for the i-th group

notice: some entry SIG[i,c] might be ∞
(thus be careful when comparing two columns c and c')



a more detailed discussion
on this and other tricks
can be found here

similarity-preserving representations for other notions of similarity

Hamming distance

- each item is a vector of size k
- two vectors are similar if the hamming distance between them is small

hamming distance between S and S':

dist(S,S')= number of entries in which S and S' differ

pick a random $i \in \{1,2,\ldots,k\}$,

$h_i(S)=S[i]$

⇨

$Pr(h_i(S)=h_i(S'))= 1 - dist(S,S')/k$

S = [G G C T A A T C G G T T A]

S'= [G G C T T A T C G C A T A]

dist(S,S') = 3

cosine similarity

- each item is a vector in a certain space (e.g., a document is a vector in the space of the terms)
- two vectors are similar if they have high cosine similarity

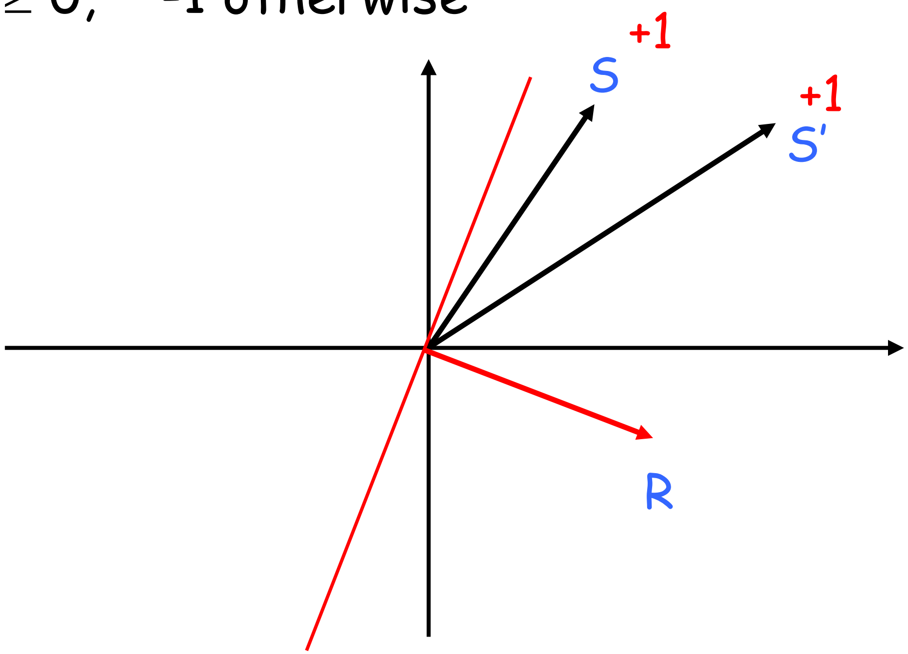
cosine similarity between S and S':

$CS(S,S')$= cosine of the angle between S and S'

pick a random vector R,

$h_R(S)=1$ if $S \cdot R \geq 0$, -1 otherwise

dot product



cosine similarity

- each item is a vector in a certain space
  (e.g., a document is a vector in the space of the terms)
- two vectors are similar if they have high cosine similarity

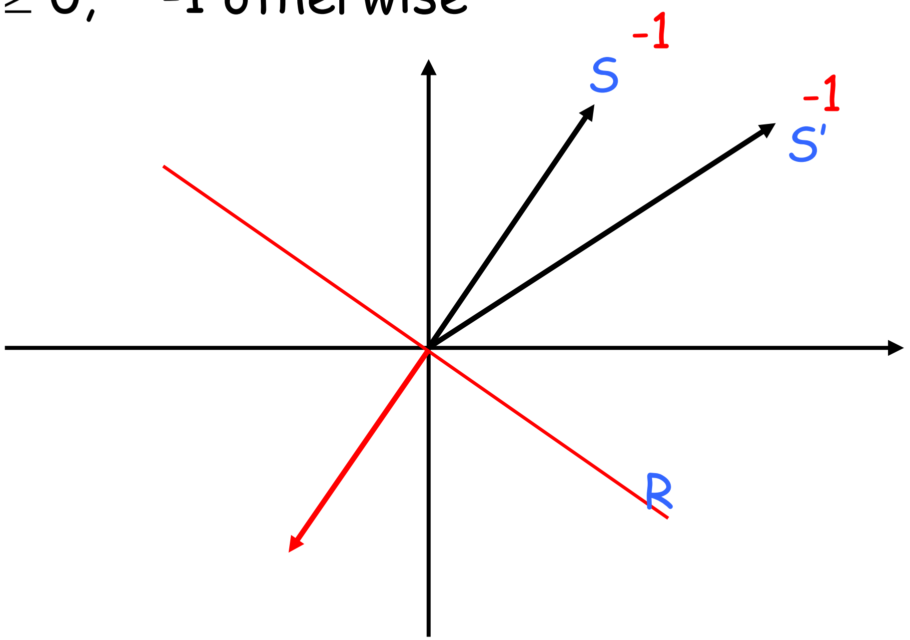
cosine similarity between S and S':

$CS(S,S')$ = cosine of the angle between S and S'

pick a random vector R,

$h_R(S) = 1$ if $S \cdot R \geq 0$, -1 otherwise

dot product



cosine similarity

- each item is a vector in a certain space (e.g., a document is a vector in the space of the terms)
- two vectors are similar if they have high cosine similarity

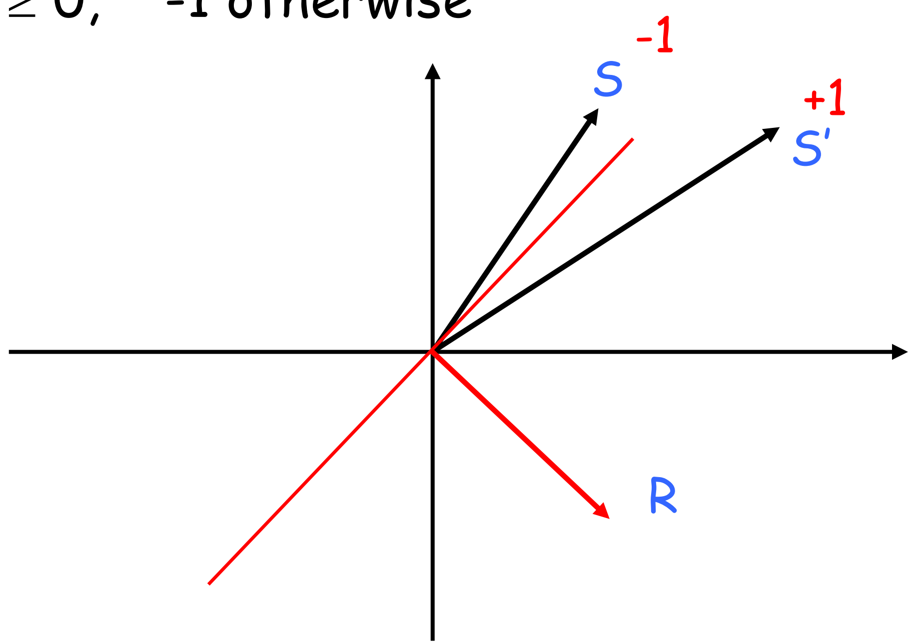
cosine similarity between S and S′:

CS(S,S′)= cosine of the angle between S and S′

pick a random vector R,

$h_R(S)=1$ if $S \cdot R \geq 0$, -1 otherwise

dot product

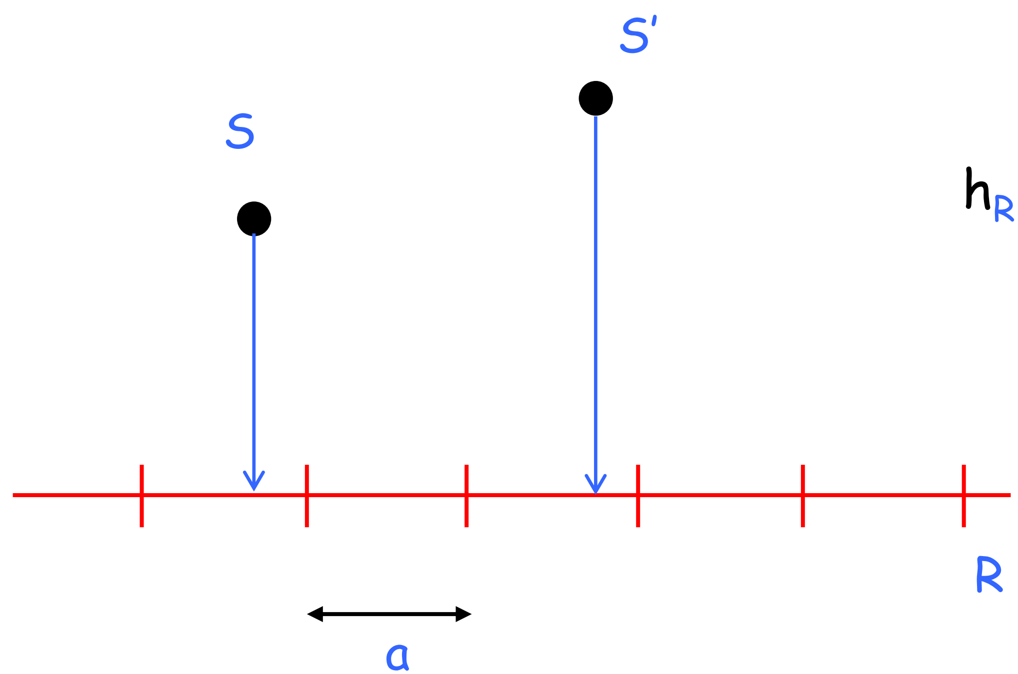


Euclidean distance

- each item is a point in a Euclidean space
- two points are similar if their Euclidean distance is small

pick a random line R, and divide it into segments (buckets) of length a

$h_R(S)$= the bucket in which S falls orthogonally to R



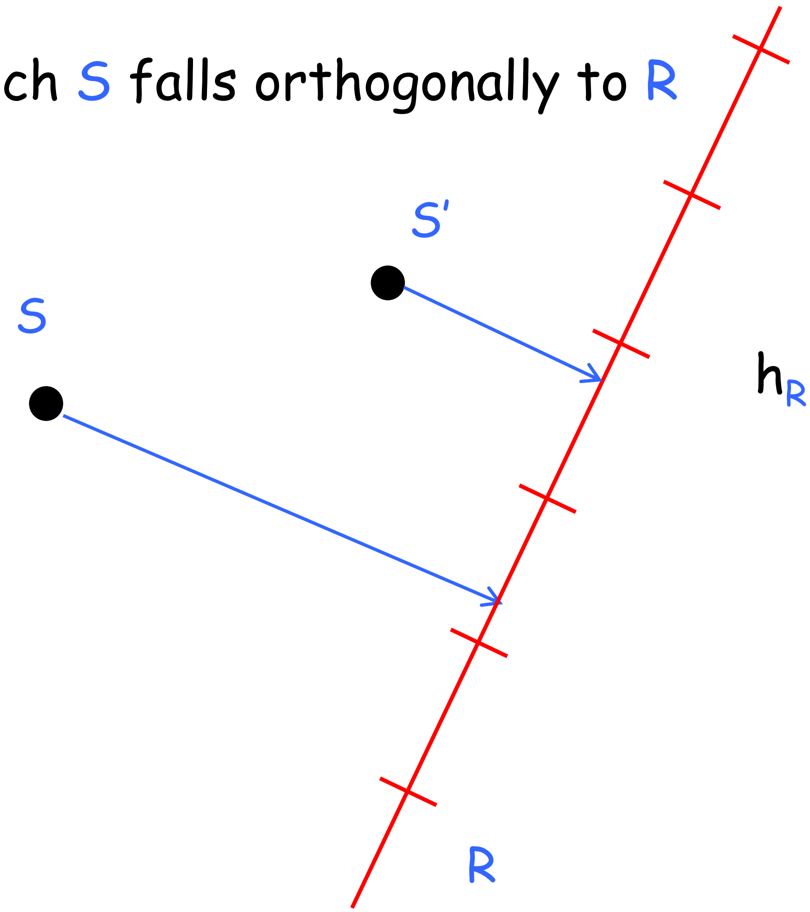
$h_R(S) \neq h_R(S')$

Euclidean distance

- each item is a point in a Euclidean space
- two points are similar if their Euclidean distance is small

pick a random line R, and divide it into segments (buckets) of length a

$h_R(S)$= the bucket in which S falls orthogonally to R



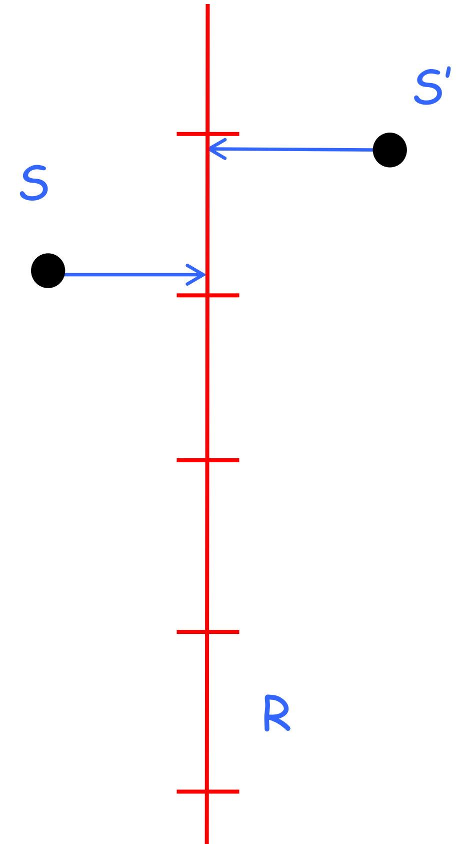
$h_R(S) \neq h_R(S')$

Euclidean distance

- each item is a point in a Euclidean space
- two points are similar if their Euclidean distance is small

pick a random line R, and divide it into segments (buckets) of length a

$h_R(S)$= the bucket in which S falls orthogonally to R



$h_R(S) = h_R(S')$