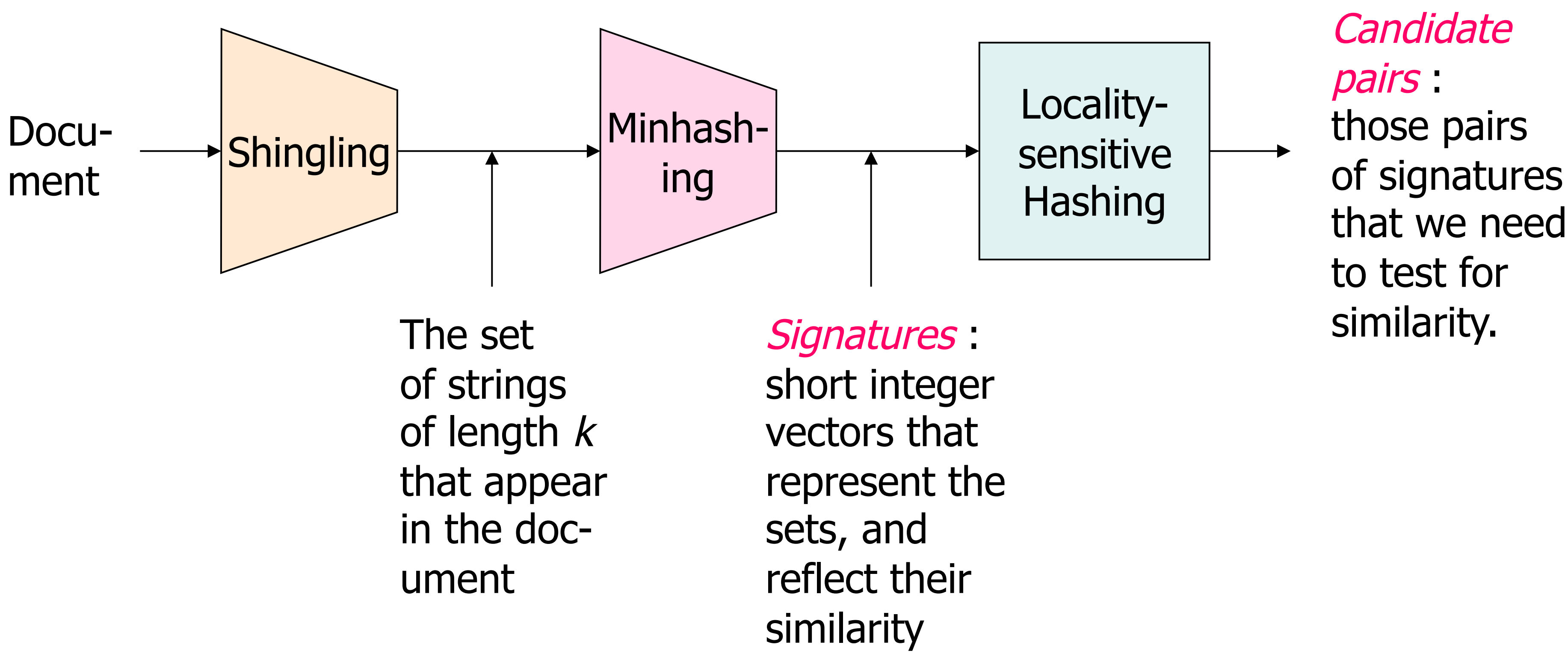
Near Neighbor Search in High Dimensional Data (2)

Locality-Sensitive Hashing (continued)
LS Families and Amplification
LS Families for Common Distance Measures

Anand Rajaraman

The Big Picture



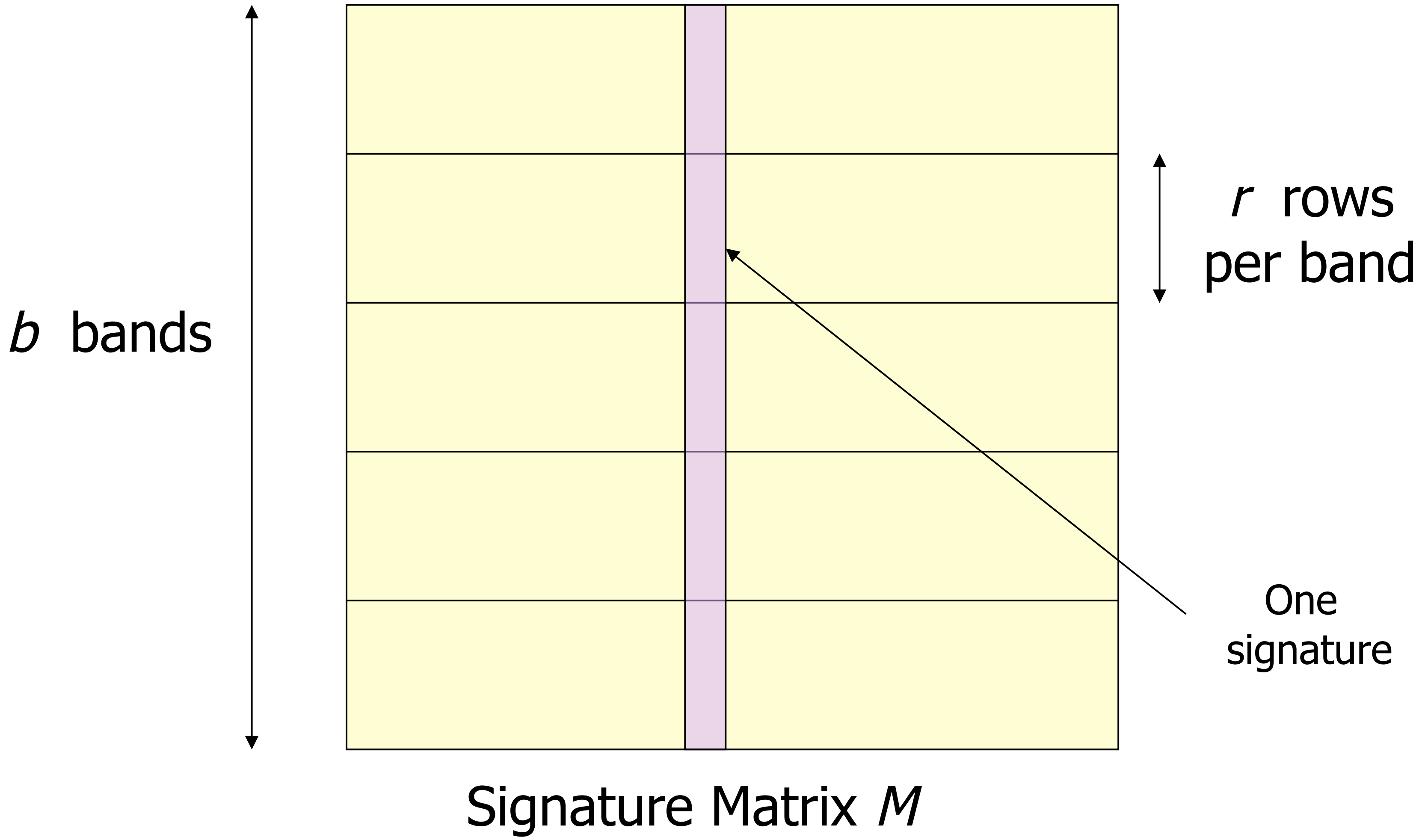
Candidate Pairs

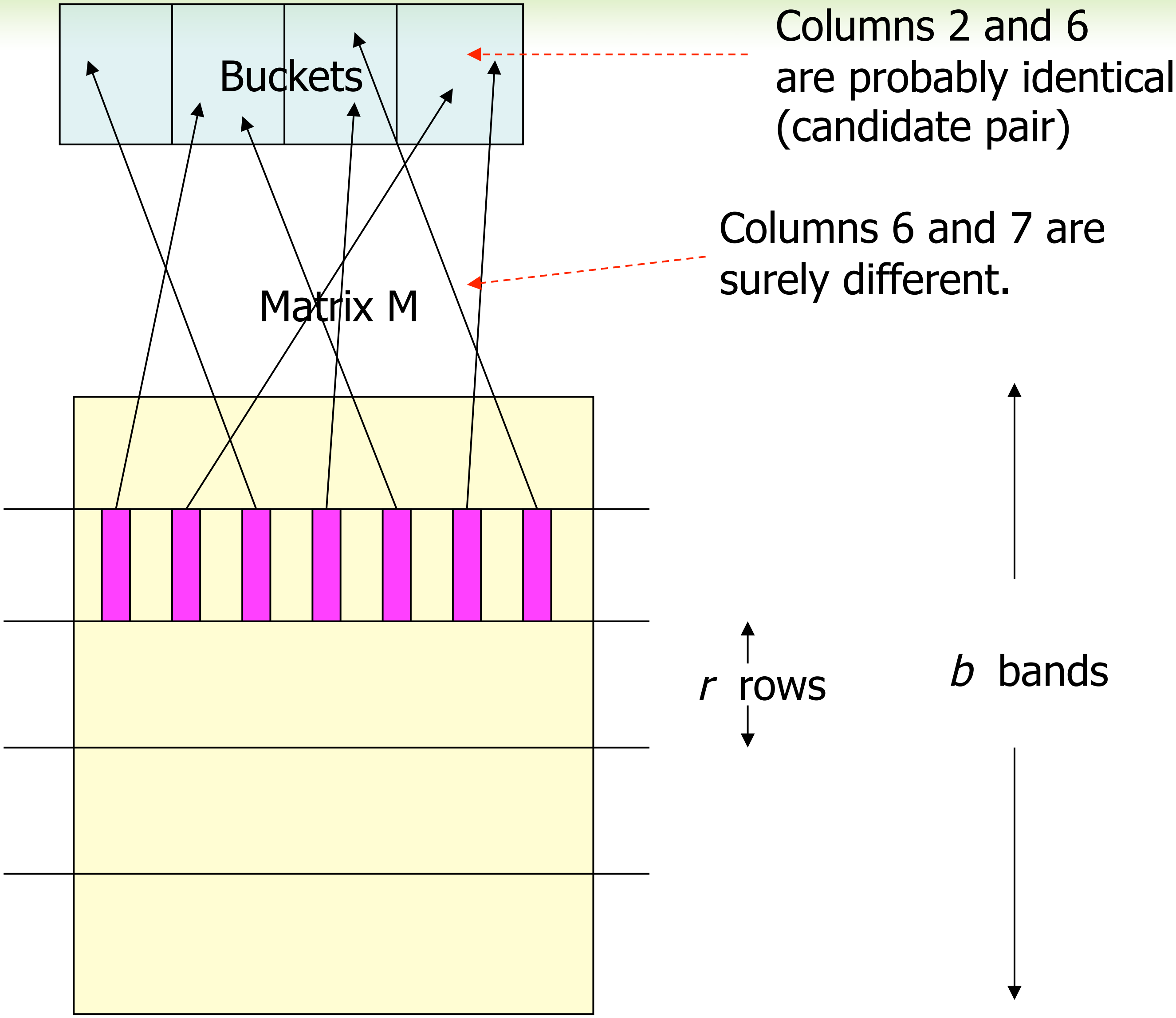
- Pick a similarity threshold *s*
  - e.g., *s* = 0.8.
  - Goal: Find documents with Jaccard similarity at least *s*.
- Columns *i* and *j* are a candidate pair if their signatures agree in at least a fraction *s* of their rows
- We expect documents *i* and *j* to have the same similarity as their signatures.

LSH for Minhash Signatures

- Big idea: hash columns of signature matrix *M* several times.
- Arrange that (only) similar columns are likely to hash to the same bucket, with high probability
- Candidate pairs are those that hash to the same bucket

Partition Into Bands




Buckets
Columns 2 and 6 are probably identical (candidate pair)
Columns 6 and 7 are surely different.
Matrix M
r rows
b bands

Partition into Bands – (2)

- Divide matrix *M* into *b* bands of *r* rows.
  - Create one hash table per band
- For each band, hash its portion of each column to its hash table
- *Candidate pairs* are columns that hash to the same bucket for $\geq$ 1 band.
- Tune *b* and *r* to catch most similar pairs, but few nonsimilar pairs.

Simplifying Assumption

- There are enough buckets that columns are unlikely to hash to the same bucket unless they are identical in a particular band.
- Hereafter, we assume that "same bucket" means "identical in that band."
- Assumption needed only to simplify analysis, not for correctness of algorithm.

Example of bands

- 100 min-hash signatures/document
- Let's choose choose b = 20, r = 5
  - 20 bands, 5 signatures per band
- Goal: find pairs of documents that are at least 80% similar.

Suppose C_1, C_2 are 80% Similar

- Probability C_1, C_2 identical in one particular band: $(0.8)^5 = 0.328$.
- Probability C_1, C_2 are *not* similar in any of the 20 bands: $(1-0.328)^{20} = .00035$.
  - i.e., about 1/3000th of the 80%-similar column pairs are false negatives
  - We would find 99.965% pairs of truly similar documents

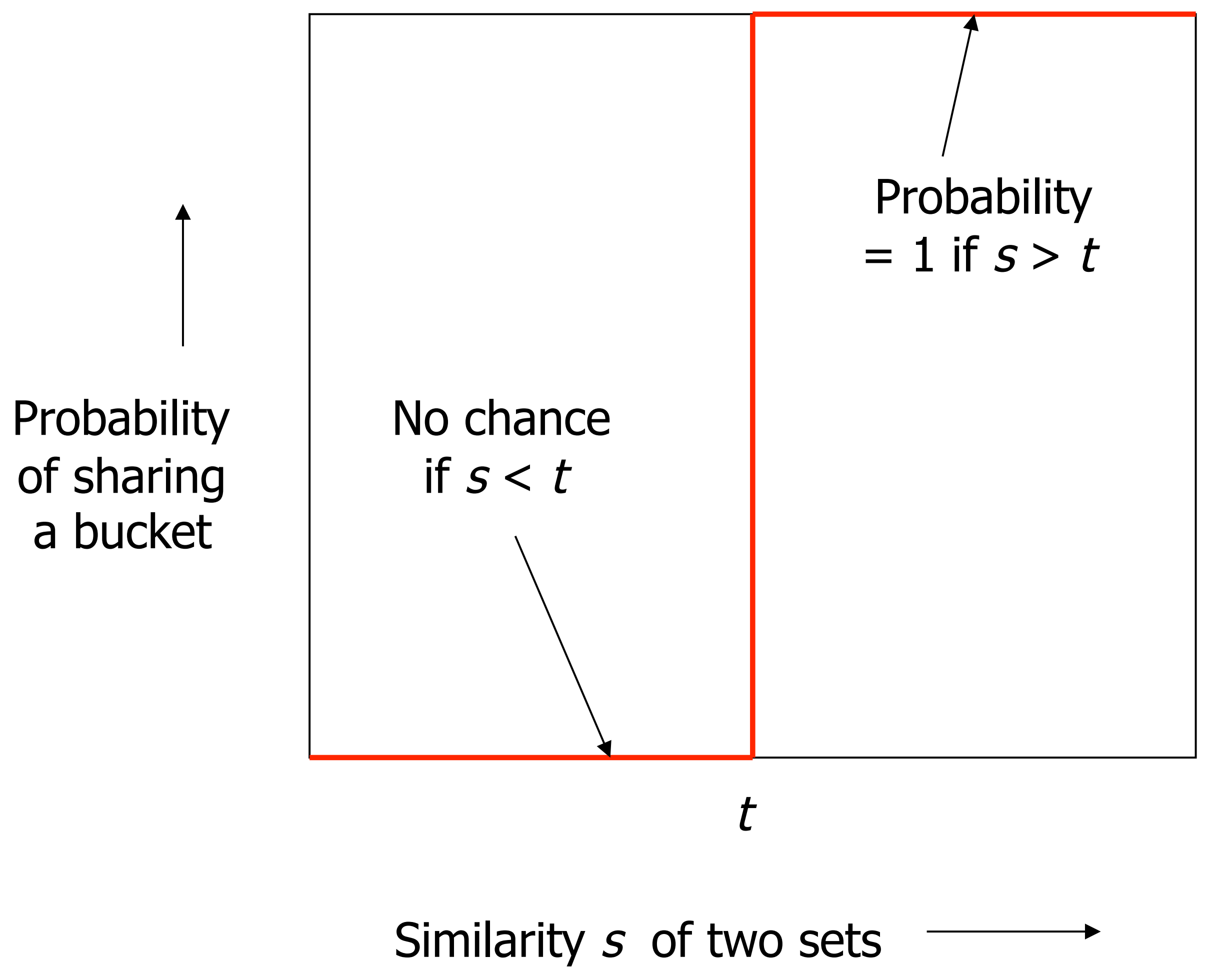
Suppose C_1, C_2 Only 30% Similar

- Probability C_1, C_2 identical in any one particular band: $(0.2)^5 = 0.00243$
- Probability C_1, C_2 identical in $\geq$ 1 of 20 bands: $20 * 0.00243 = 0.0486$
- In other words, approximately 4.86% pairs of docs with similarity 30% end up becoming candidate pairs
  - False positives

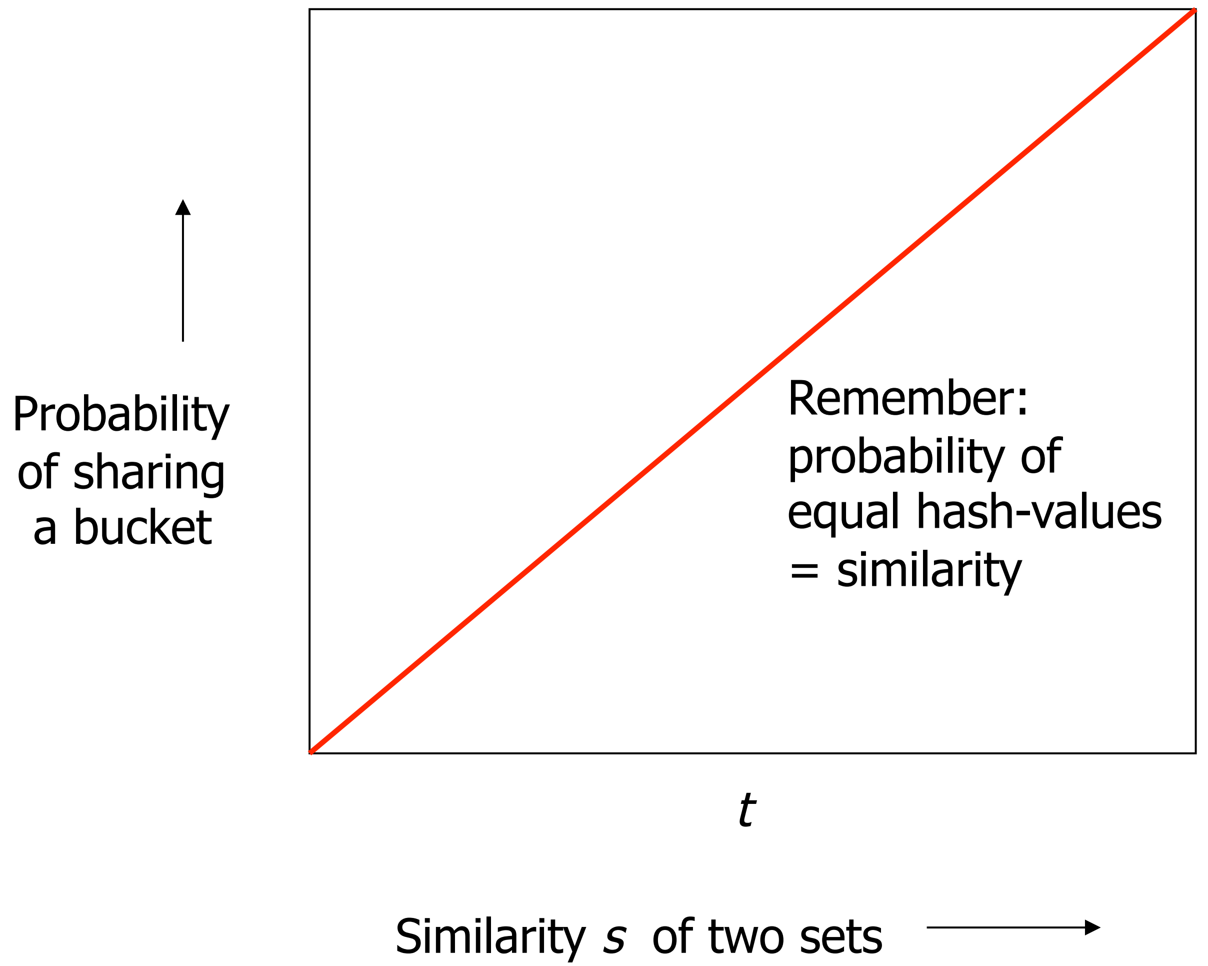
LSH Involves a Tradeoff

- Pick the number of minhashes, the number of bands, and the number of rows per band to balance false positives/ negatives.
- Example: if we had only 15 bands of 5 rows, the number of false positives would go down, but the number of false negatives would go up.

Analysis of LSH – What We Want



What One Band of One Row Gives You

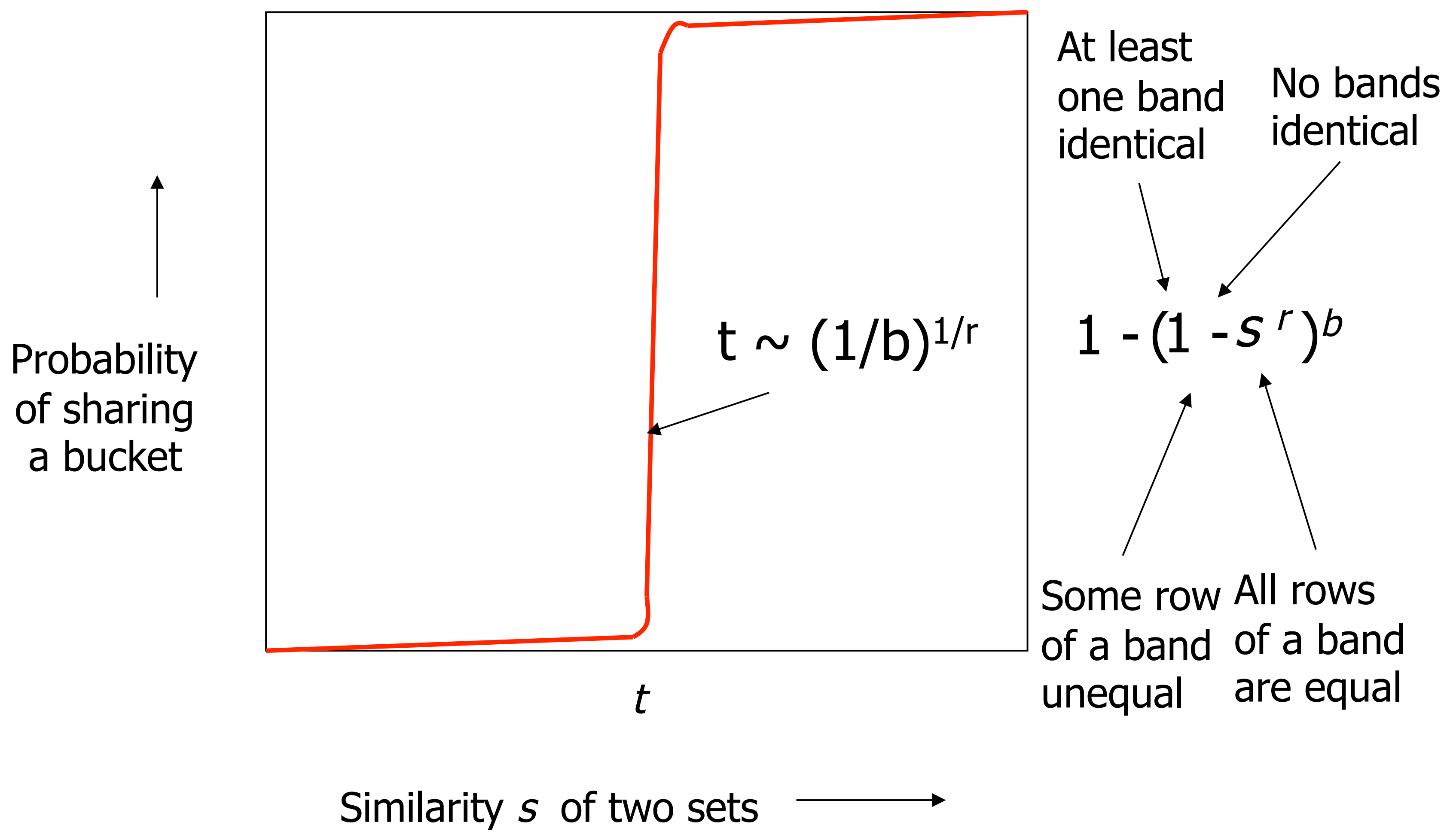


b bands, r rows/band

- Columns C and D have similarity s
- Pick any band (r rows)
  - Prob. that all rows in band equal = s^r
  - Prob. that some row in band unequal = $1 - s^r$
- Prob. that no band identical = $(1 - s^r)^b$
- Prob. that at least 1 band identical =

$$1 - (1 - s^r)^b$$

What *b* Bands of *r* Rows Gives You



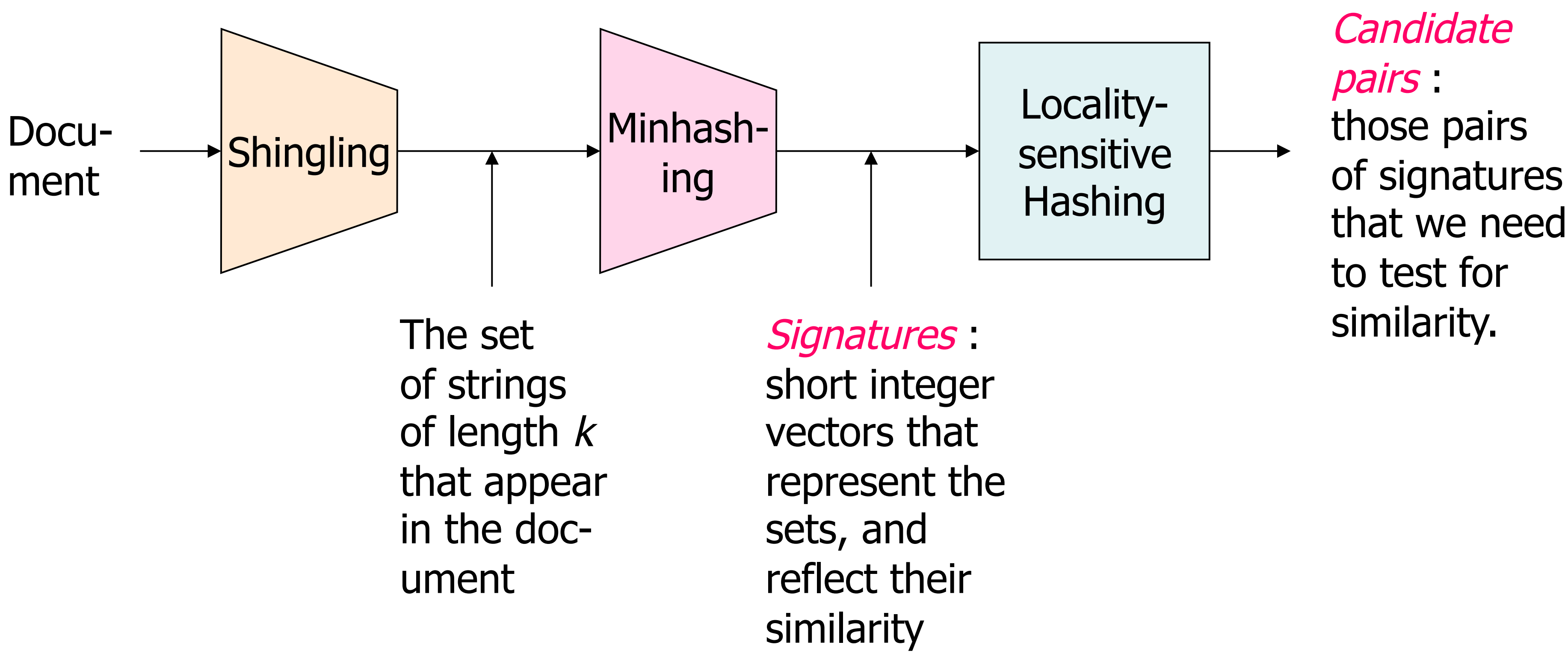
Example: $b = 20$; $r = 5$

s	$1-(1-s^r)^b$
.2	.006
.3	.047
.4	.186
.5	.470
.6	.802
.7	.975
.8	.9996

LSH Summary

- Tune to get almost all pairs with similar signatures, but eliminate most pairs that do not have similar signatures.
- Check in main memory that candidate pairs really do have similar signatures.
- Optional: In another pass through data, check that the remaining candidate pairs really represent similar documents.

The Big Picture



Theory of LSH

- We have used LSH to find similar documents
  - In reality, columns in large sparse matrices with high Jaccard similarity
  - e.g., customer/item purchase histories
- Can we use LSH for other distance measures?
  - e.g., Euclidean distances, Cosine distance
  - Let's generalize what we've learned!

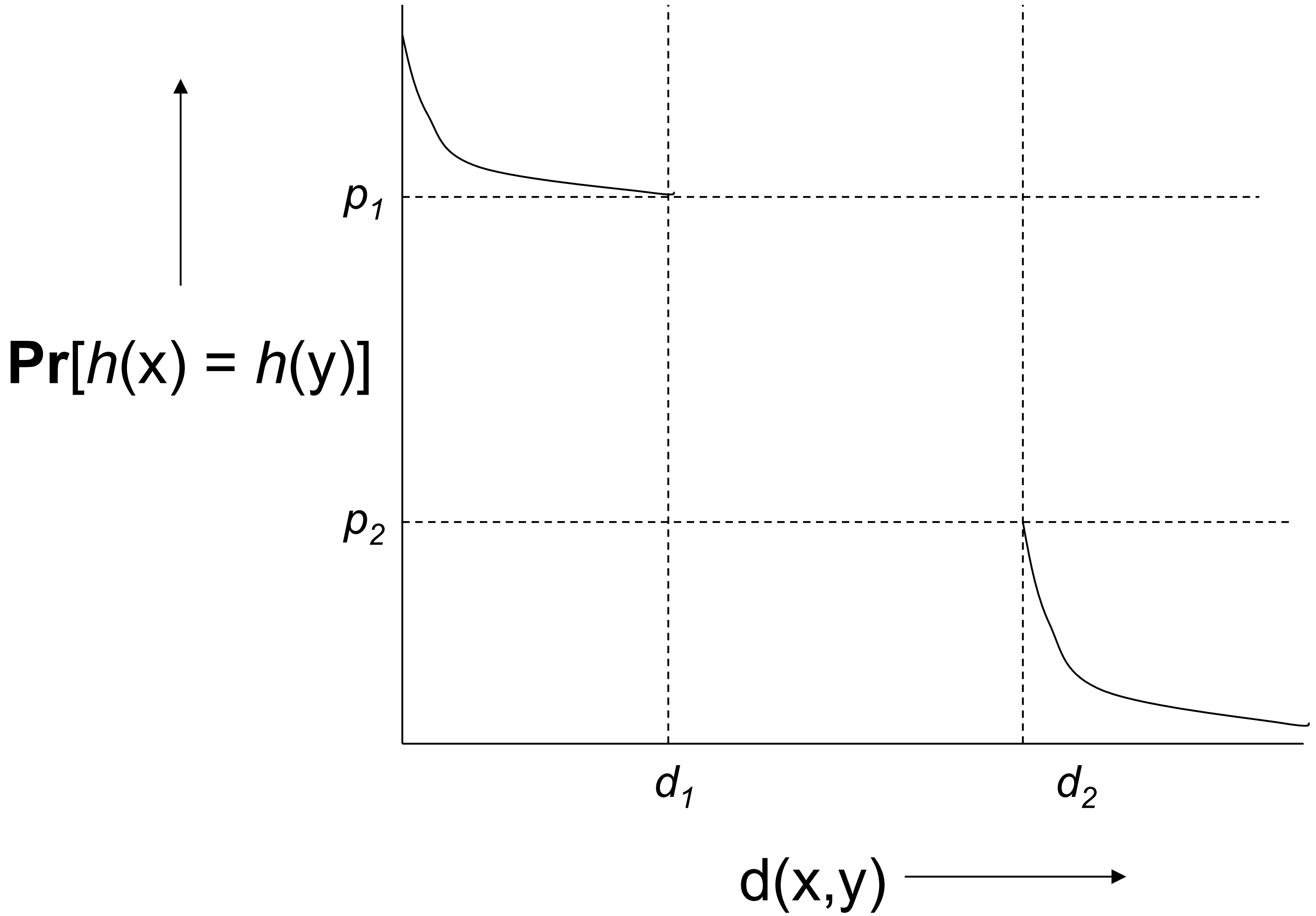
Families of Hash Functions

- For min-hash signatures, we got a min-hash function for each permutation of rows
- An example of a family of hash functions
  - A (large) set of related hash functions generated by some mechanism
  - We should be able to effciently pick a hash function at random from such a family

Locality-Sensitive (LS) Families

- Suppose we have a space *S* of points with a distance measure *d*.
- A family **H** of hash functions is said to be *(d_1,d_2,p_1,p_2)-sensitive* if for any *x* and *y* in *S* :
  1. If $d(x,y) \leq d_1$, then prob. over all *h* in **H**, that $h(x) = h(y)$ is at least p_1.
  2. If $d(x,y) \geq d_2$, then prob. over all *h* in **H**, that $h(x) = h(y)$ is at most p_2.

A (d_1,d_2,p_1,p_2)-sensitive function



Example: LS Family

- Let *S* = sets, *d* = Jaccard distance, **H** is family of minhash functions for all permutations of rows
- Then for any hash function h in **H**,

  Pr[h(x)=h(y)] = 1-d(x,y)
- Simply restates theorem about min-hashing in terms of distances rather than similarities

Example: LS Family – (2)

- Claim: **H** is a (1/3, 2/3, 2/3, 1/3)-sensitive family for *S* and *d*.

If distance ≤ 1/3
(so similarity ≥ 2/3)

Then probability
that minhash values
agree is ≥ 2/3

- For Jaccard similarity, minhashing gives us a (d1,d2,(1-d1),(1-d2))-sensitive family for any d1 < d2.

Amplifying a LS-Family

- Can we reproduce the "S-curve" effect we saw before for any LS family?
- The "bands" technique we learned for signature matrices carries over to this more general setting.
- Two constructions:
  - AND construction like "rows in a band."
  - OR construction like "many bands."

AND of Hash Functions

- Given family **H**, construct family **H'** consisting of *r* functions from **H**.
- For $h = [h_1, \ldots, h_r]$ in **H'**, h(x)=h(y) if and only if $h_i(x)=h_i(y)$ for all *i*.
- Theorem: If **H** is (d_1, d_2, p_1, p_2)-sensitive, then **H'** is $(d_1, d_2, (p_1)^r, (p_2)^r)$-sensitive.
- Proof: Use fact that h_i's are independent.

OR of Hash Functions

- Given family **H**, construct family **H'** consisting of *b* functions from **H**.
- For $h = [h_1,\ldots,h_b]$ in **H'**, h(x)=h(y) if and only if $h_i(x)=h_i(y)$ for some *i*.
- Theorem: If **H** is (d_1,d_2,p_1,p_2)-sensitive, then **H'** is $(d_1,d_2,1-(1-p_1)^b,1-(1-p_2)^b)$-sensitive.

Composing Constructions

- *r*-way AND construction followed by *b*-way OR construction
  - Exactly what we did with minhashing
- Take points x and y s.t. **Pr**[*h*(x) = *h*(y)] = p
  - **H** will make (x,y) a candidate pair with prob. p
- This construction will make (x,y) a candidate pair with probability $1-(1-p^r)^b$
  - The S-Curve!

AND-OR Composition

- Example: Take **H** and construct **H'** by the AND construction with $r = 4$. Then, from **H'**, construct **H''** by the OR construction with $b = 4$.

Table for Function $1-(1-p^4)^4$

p	**$1-(1-p^4)^4$**
.2	.0064
.3	.0320
.4	.0985
.5	.2275
.6	.4260
.7	.6666
.8	.8785
.9	.9860

Example: Transforms a (.2,.8,.8,.2)-sensitive family into a (.2,.8,.8785,.0064)-sensitive family.

OR-AND Composition

- Apply a b-way OR construction followed by an r-way AND construction
- Tranforms probability *p* into $(1-(1-p)^b)^r$.
  - The same S-curve, mirrored horizontally and vertically.
- Example: Take **H** and construct **H'** by the OR construction with *b* = 4. Then, from **H'**, construct **H''** by the AND construction with *r* = 4.

Table for Function $(1-(1-p)^4)^4$

p	$(1-(1-p)^4)^4$
.1	.0140
.2	.1215
.3	.3334
.4	.5740
.5	.7725
.6	.9015
.7	.9680
.8	.9936

Example: Transforms a (.2,.8,.8,.2)-sensitive family into a (.2,.8,.9936,.1215)-sensitive family.

Cascading Constructions

- Example: Apply the (4,4) OR-AND construction followed by the (4,4) AND-OR construction.
- Transforms a (.2,.8,.8,.2)-sensitive family into a (.2,.8,.9999996,.0008715)-sensitive family.
- Note this family uses 256 of the original hash functions.

Summary

- Pick any two distances $x < y$
- Start with a (x, y, (1-x), (1-y))-sensitive family
- Apply constructions to produce (x, y, p, q)-sensitive family, where *p* is almost 1 and *q* is almost 0.
- The closer to 0 and 1 we get, the more hash functions must be used.

LSH for Cosine Distance

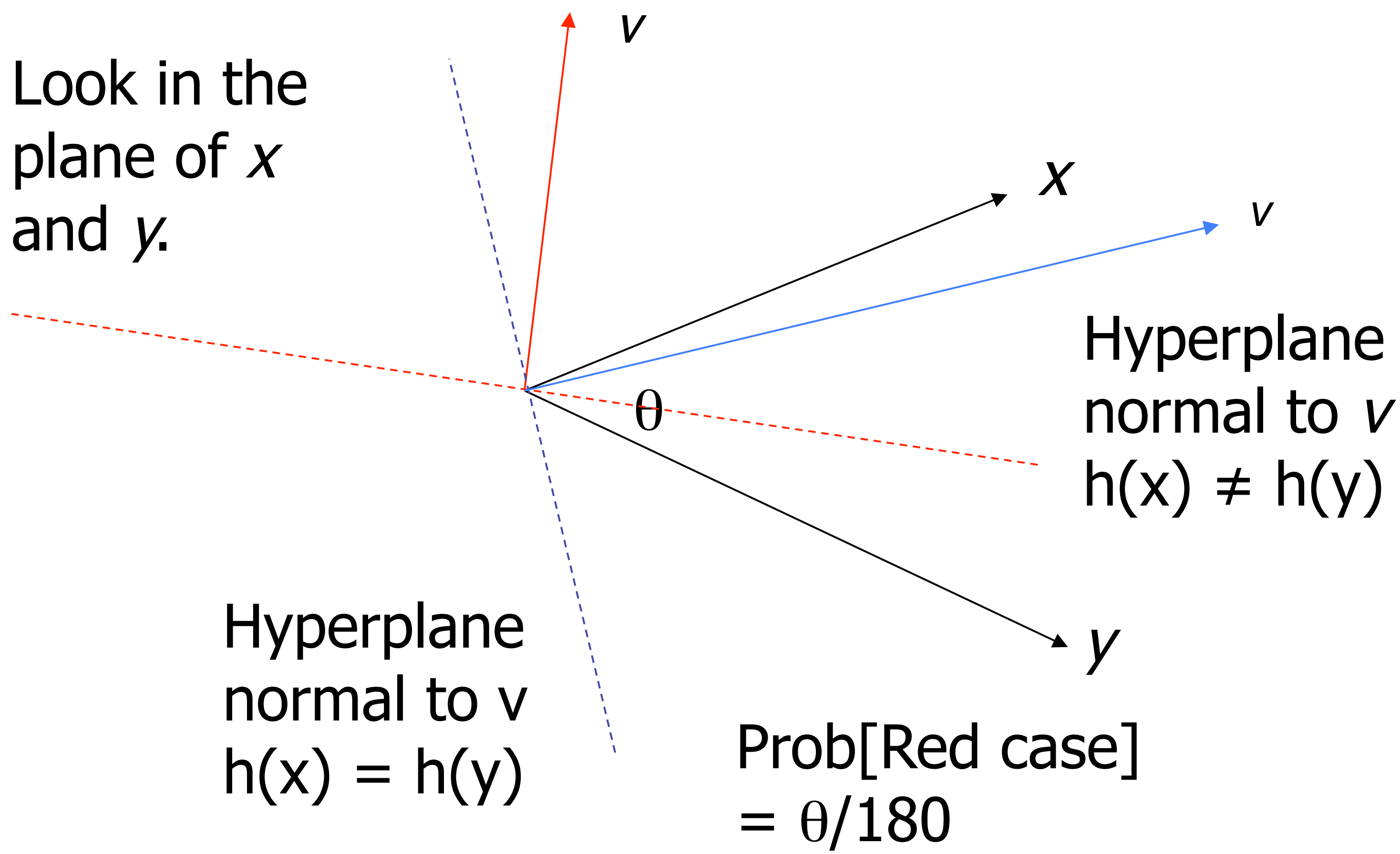
- Random Hypeplanes
  - Technique similar to minhashing
- A $(d_1, d_2, (1-d_1/180), (1-d_2/180))$-sensitive family for any d_1 and d_2.

Random Hyperplanes

- Pick a random vector *v*, which determines a hash function h_v with two buckets.
- $h_v(x) = +1$ if $v.x > 0$; $= -1$ if $v.x < 0$.
- LS-family **H** = set of all functions derived from any vector.
- Claim: For points x and y,

  Pr[h(x)=h(y)] = 1 – d(x,y)/180

Proof of Claim



Signatures for Cosine Distance

- Pick some number of random vectors, and hash your data for each vector.
- The result is a signature (*sketch*) of +1's and –1's for each data point
- Can be used for LSH like the minhash signatures for Jaccard distance.
- Amplified using AND and OR constructions

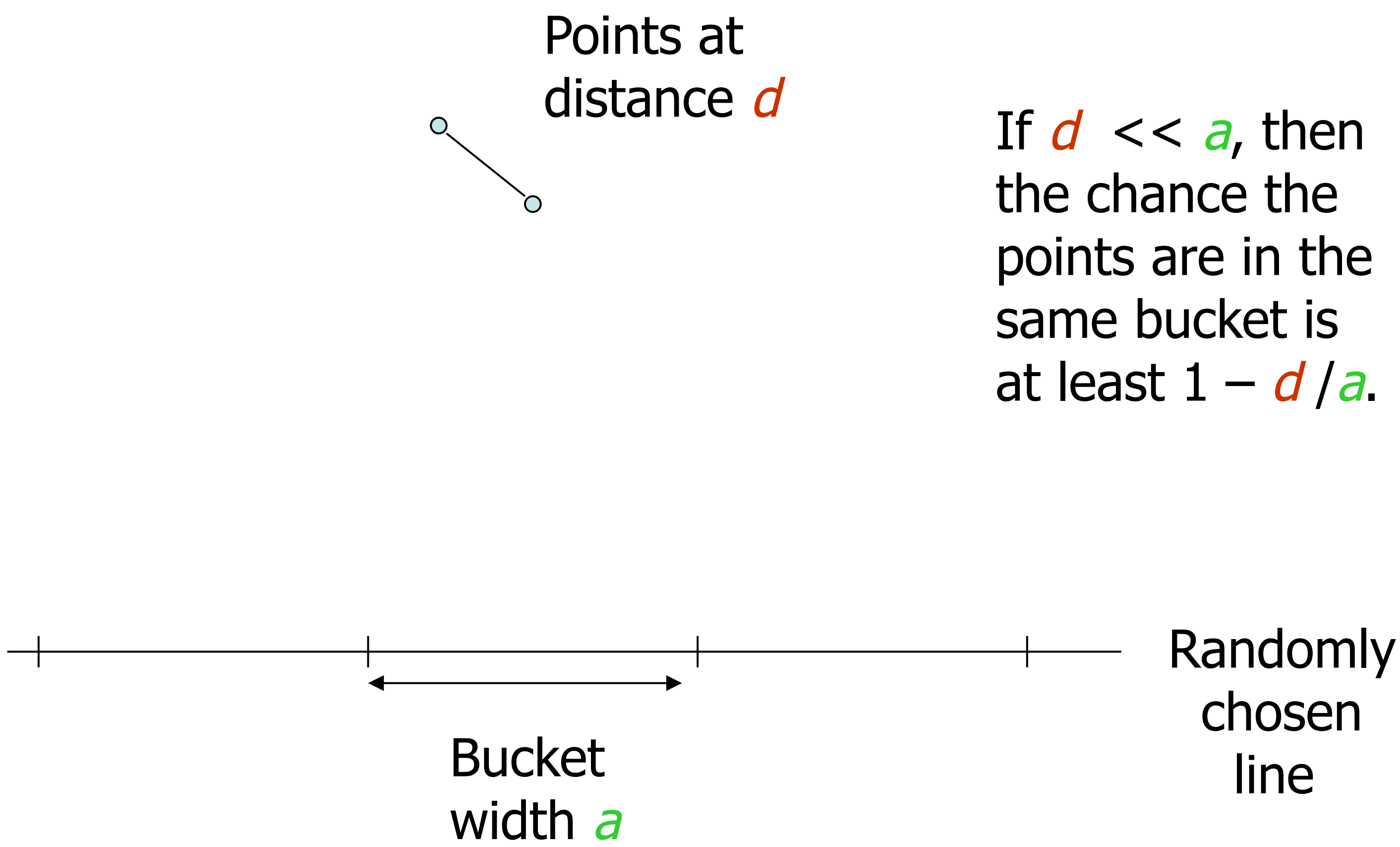
How to pick random vectors

- Expensive to pick a random vector in M dimensions for large M
  - M random numbers
- A more efficient approach
  - It suffices to consider only vectors *v* consisting of +1 and –1 components.
  - Why is this more efficient?

LSH for Euclidean Distance

- Simple idea: hash functions correspond to lines.
- Partition the line into buckets of size *a*.
- Hash each point to the bucket containing its projection onto the line.
- Nearby points are always close; distant points are rarely in same bucket.

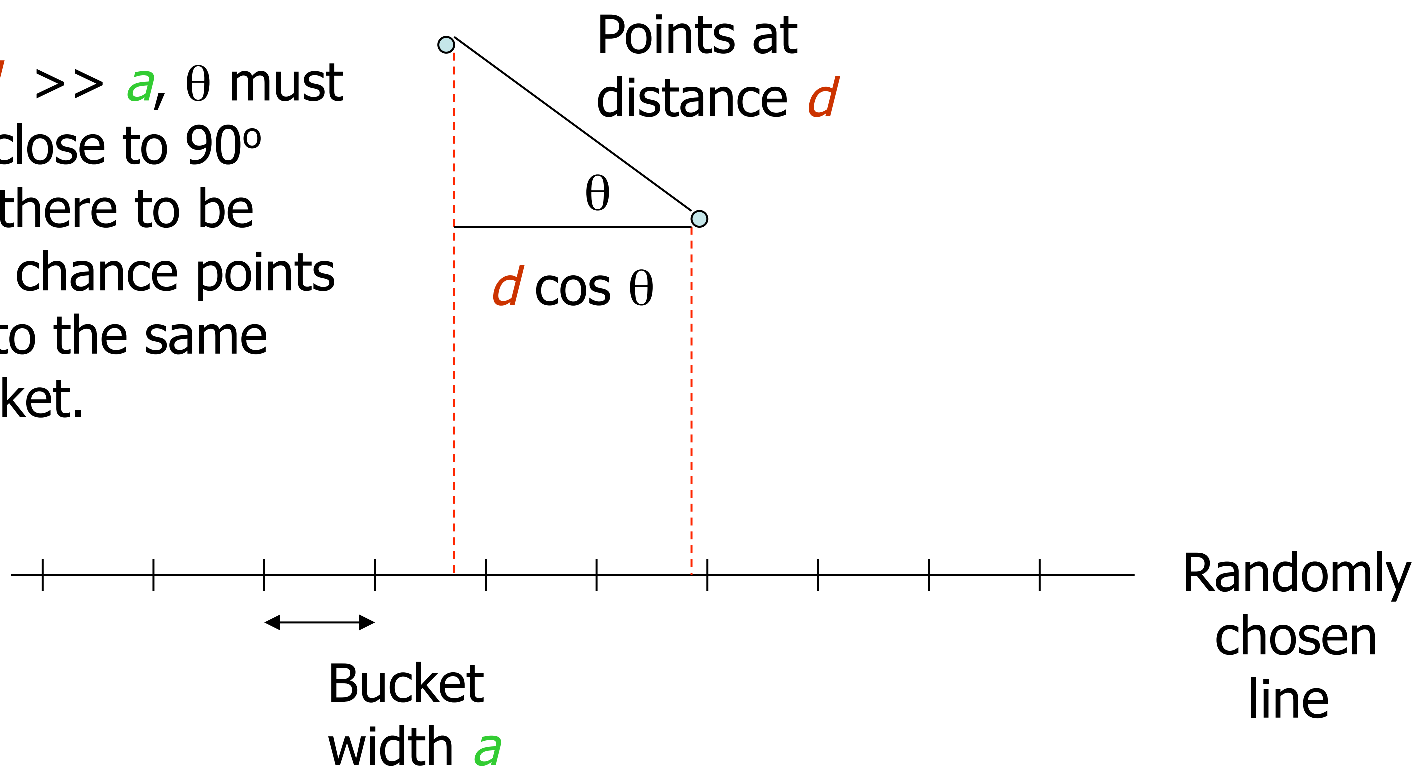
Projection of Points



If *d* << *a*, then the chance the points are in the same bucket is at least 1 – *d* / *a*.

Projection of Points

If d >> a, θ must be close to 90º for there to be any chance points go to the same bucket.



An LS-Family for Euclidean Distance

- If points are distance $d \leq a/2$, prob. they are in same bucket ≥ 1- d/a = 1/2
- If points are distance $\geq 2a$ apart, then they can be in the same bucket only if d cos θ ≤ a
  - cos θ ≤ ½
  - 60 ≤ θ ≤ 90
  - I.e., at most 1/3 probability.
- Yields a (a/2, 2a, 1/2, 1/3)-sensitive family of hash functions for any a.
- Amplify using AND-OR cascades