

2 Announcements

Recitation session:

- Review of linear algebra
 - Location: Thursday, January 17, from 4:30-5:50 pm in Gates B01

Deadlines next Thu, 11:59 PM:

- HW0, HW1, GHW1, GHW2

Theory of Locality Sensitive Hashing

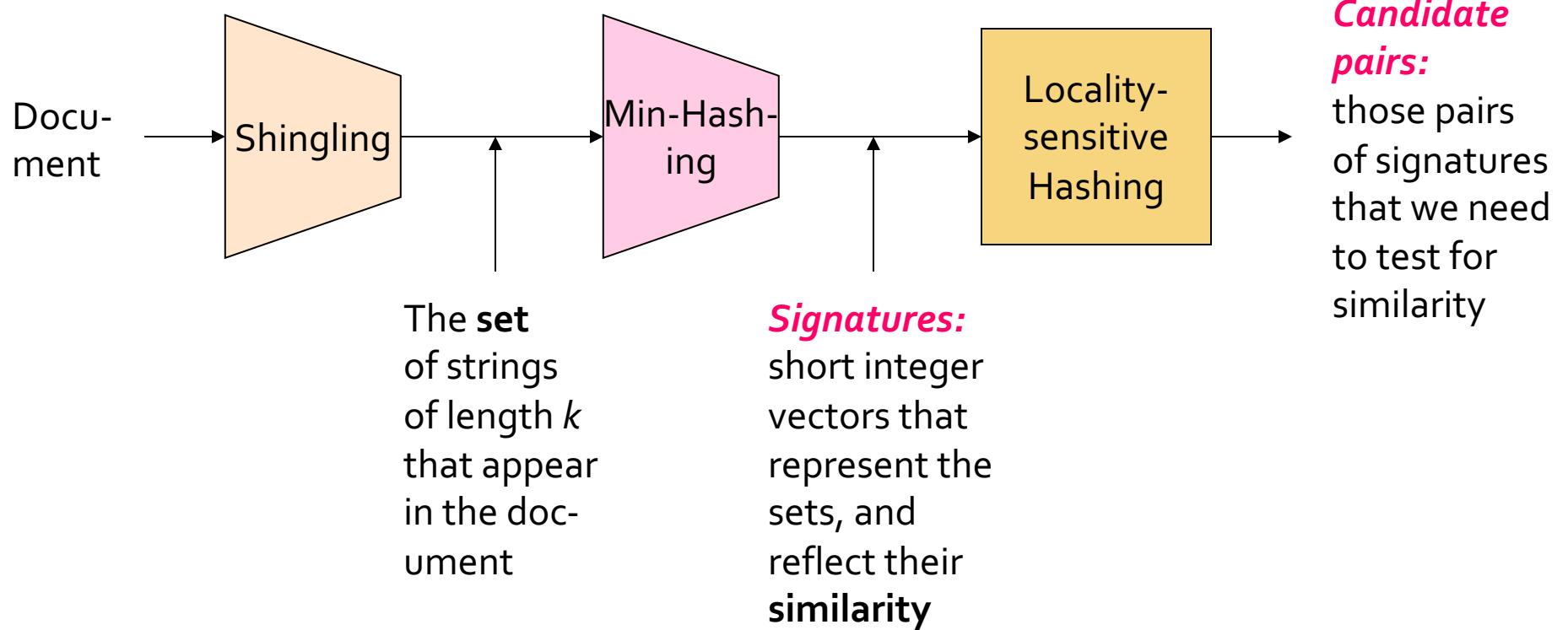
CS246: Mining Massive Datasets
Jure Leskovec, Stanford University
<http://cs246.stanford.edu>



Recap: Finding similar documents

- **Task:** Given a large number (N in the millions or billions) of documents, find “near duplicates”
- **Problem:**
 - Too many documents to compare all pairs
- **Solution:** Hash documents so that similar documents hash into the same bucket
 - Documents in the same bucket are then **candidate pairs** whose similarity is then evaluated

Recap: The Big Picture



Recap: Shingles

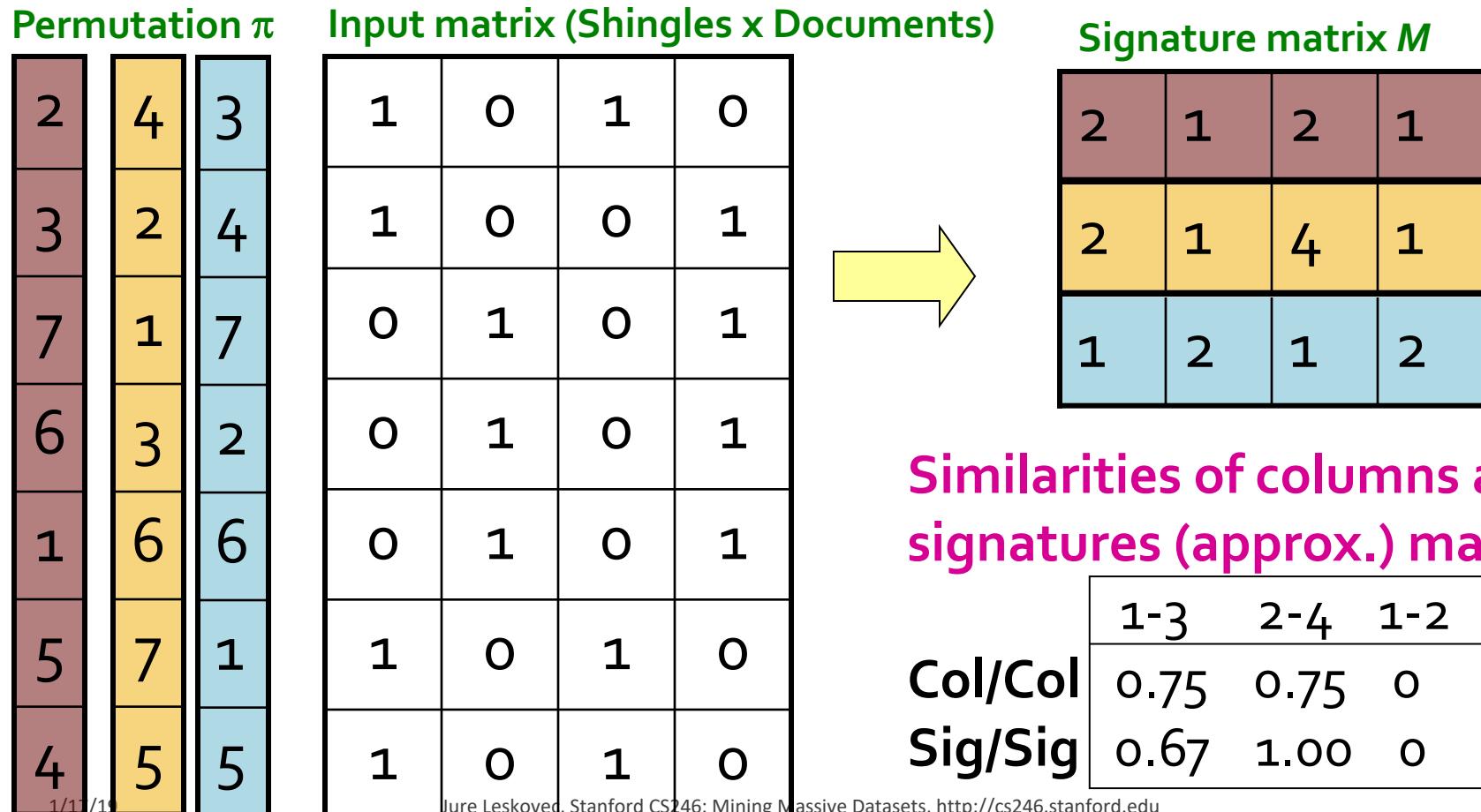
- A *k-shingle* (or *k-gram*) is a sequence of *k* tokens that appears in the document
 - Example: $k=2$; $D_1 = \text{abcab}$
Set of 2-shingles: $C_1 = S(D_1) = \{\text{ab}, \text{bc}, \text{ca}\}$
- Represent a doc by a set of hash values of its *k-shingles*
- A natural **similarity measure** is then the **Jaccard similarity**:

$$\text{sim}(D_1, D_2) = |C_1 \cap C_2| / |C_1 \cup C_2|$$

- Similarity of two documents is the Jaccard similarity of their shingles

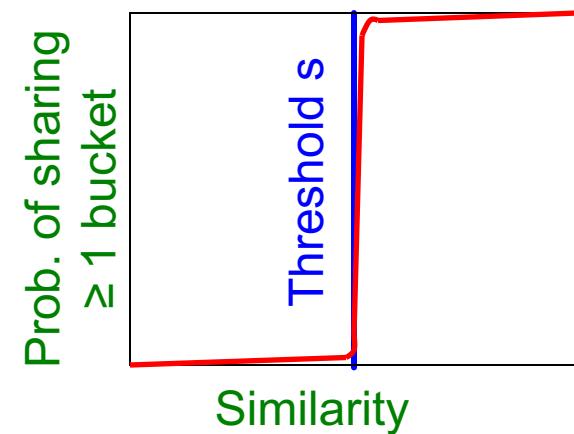
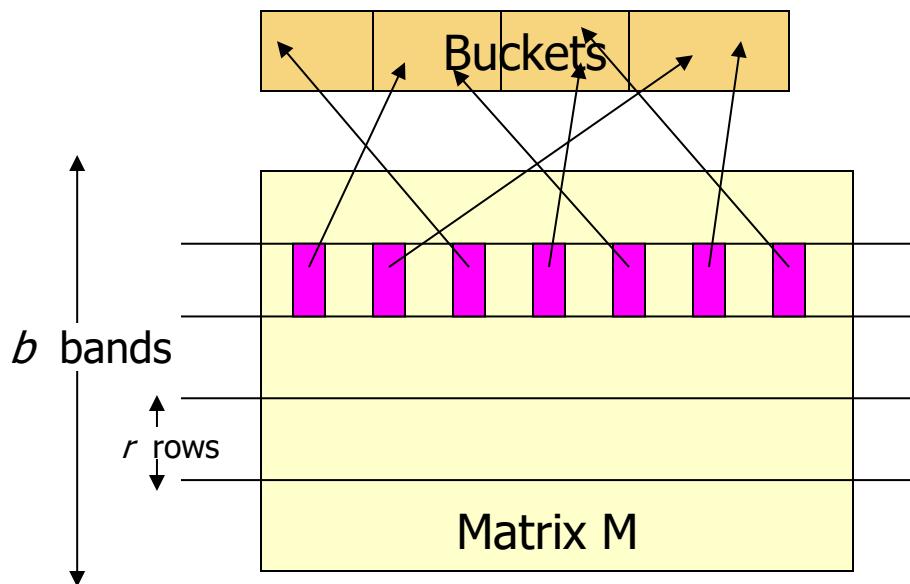
Recap: Minhashing

- **Min-Hashing**: Convert large sets into short signatures, while preserving similarity: $\Pr[h(C_1) = h(C_2)] = \text{sim}(D_1, D_2)$

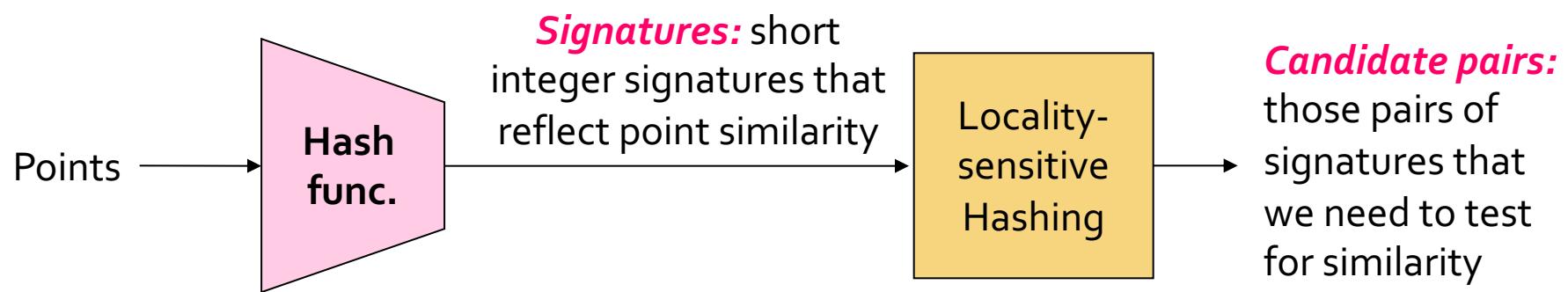


Recap: LSH

- Hash columns of the signature matrix M :
Similar columns likely hash to same bucket
 - Divide matrix M into b bands of r rows ($M=b \cdot r$)
 - *Candidate* column pairs are those that hash to the same bucket for ≥ 1 band



Today: Generalizing Min-hash

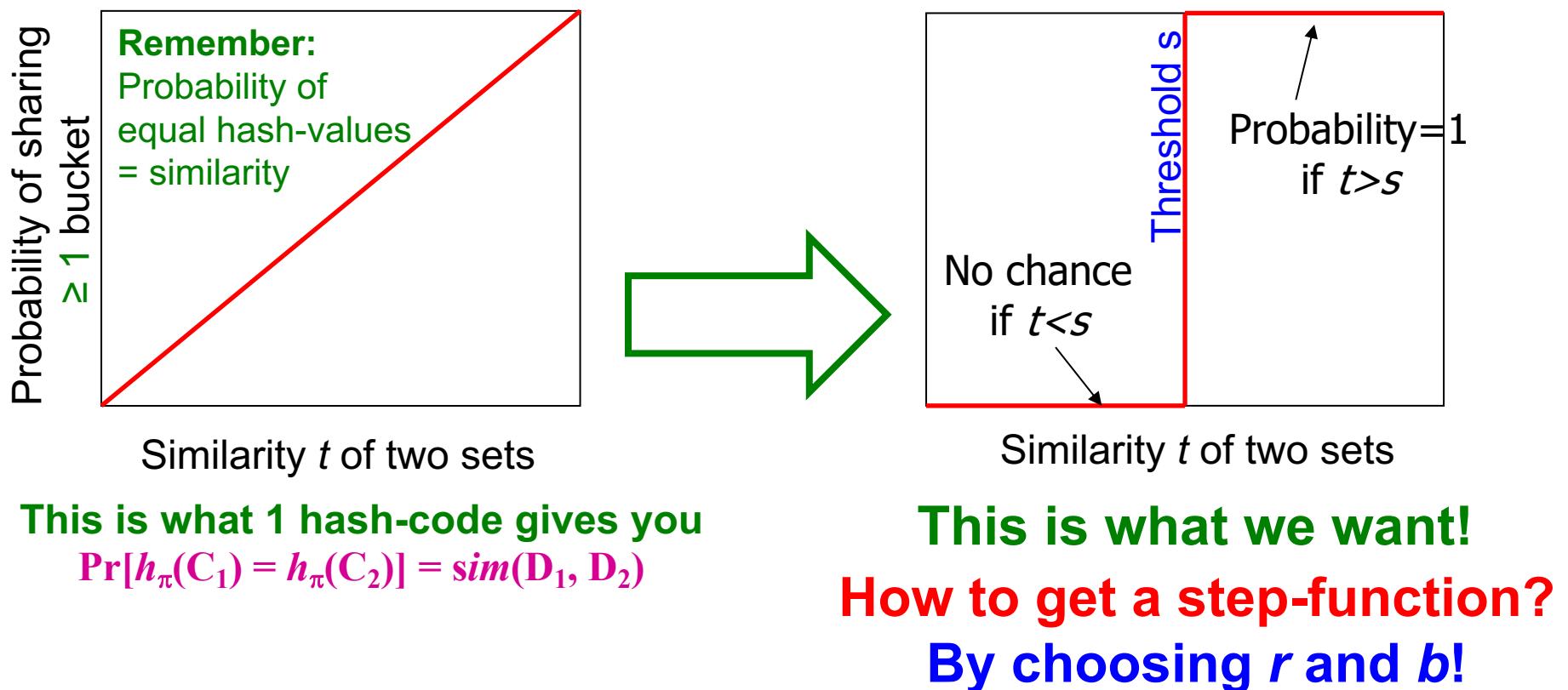


Design a **locality sensitive hash function (for a given distance metric)**

Apply the
“Bands” technique

The S-Curve

- The S-curve is where the “magic” happens



How Do We Make the S-curve?

- Remember: b bands, r rows/band
- Let $\text{sim}(\mathbf{C}_1, \mathbf{C}_2) = s$

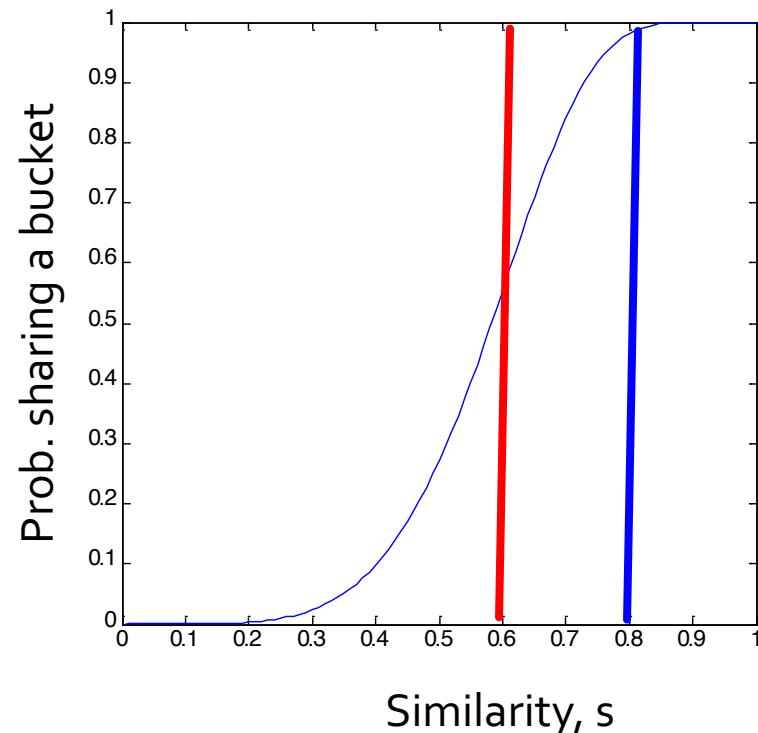
What's the prob. that at least 1 band is equal?

- Pick some band (r rows)
 - Prob. that elements in a single row of columns \mathbf{C}_1 and \mathbf{C}_2 are equal = s
 - Prob. that all rows in a band are equal = s^r
 - Prob. that some row in a band is not equal = $1 - s^r$
- Prob. that all bands are not equal = $(1 - s^r)^b$
- Prob. that at least 1 band is equal = $1 - (1 - s^r)^b$

$P(\mathbf{C}_1, \mathbf{C}_2 \text{ is a candidate pair}) = 1 - (1 - s^r)^b$

Picking r and b : The S-curve

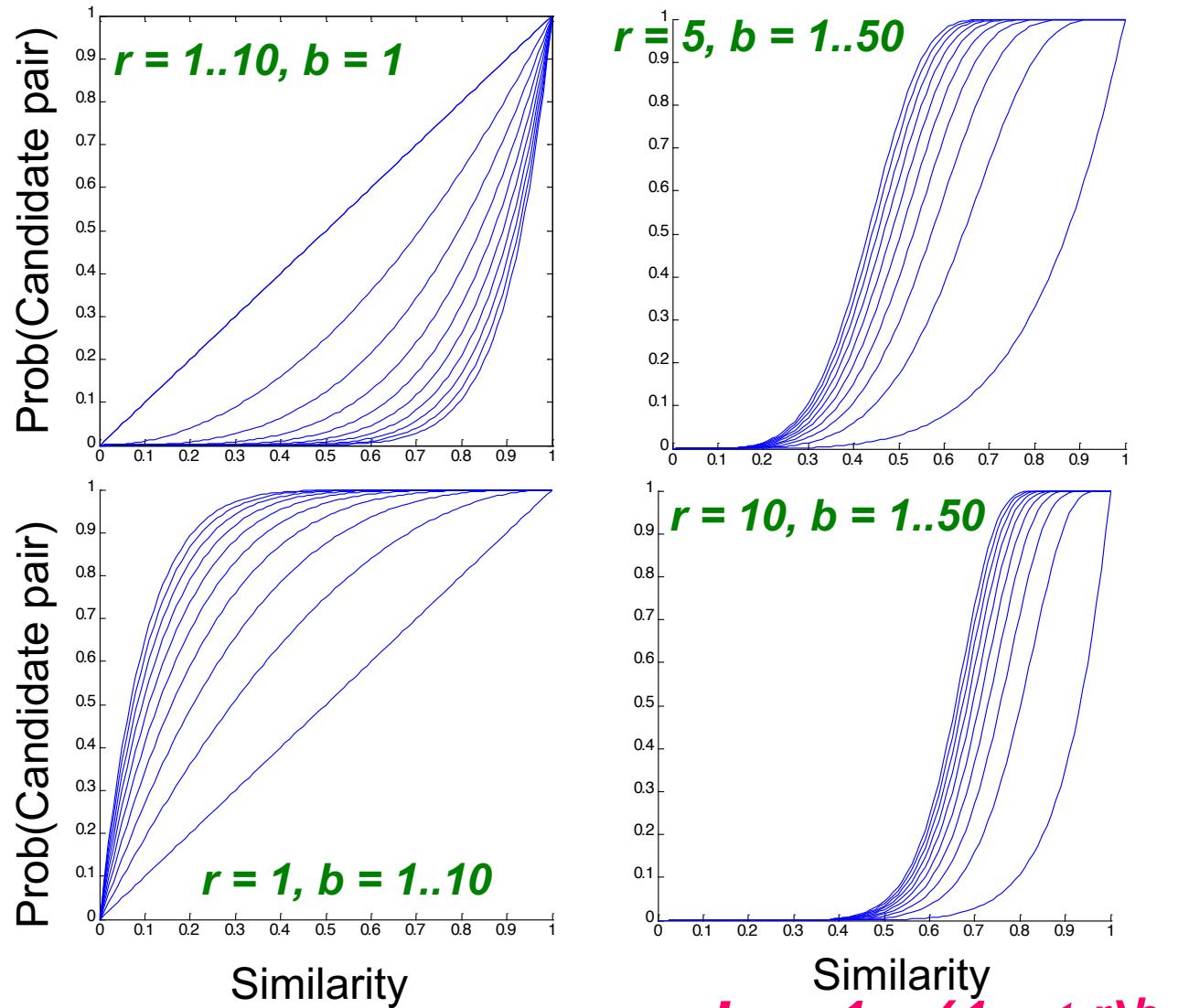
- Picking r and b to get the best S-curve
 - 50 hash-functions ($r=5$, $b=10$)

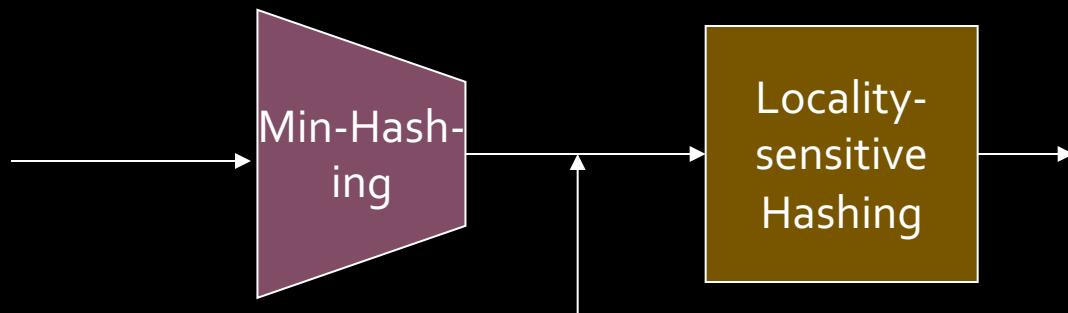


S-curves as a func. of b and r

Given a fixed threshold s .

We want choose r and b such that the **$P(\text{Candidate pair})$** has a “step” right around s .





Signatures:

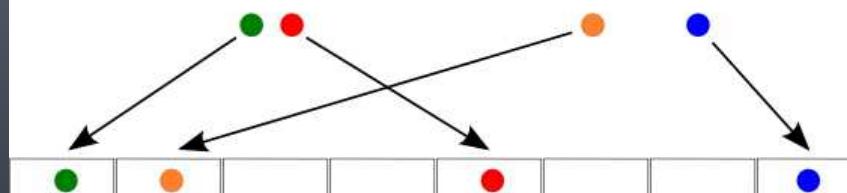
short vectors
that represent
the sets, and
reflect their
similarity

Candidate pairs:

those pairs
of signatures
that we need
to test for
similarity

Theory of LSH

general hashing



locality-sensitive hashing



Theory of LSH

- **We have used LSH to find similar documents**
 - More generally, we found similar columns in large sparse matrices with high Jaccard similarity
- **Can we use LSH for other distance measures?**
 - e.g., Euclidean distances, Cosine distance
 - **Let's generalize what we've learned!**

Distance Measures

- $d()$ is a **distance measure** if it is a function from pairs of points \mathbf{x}, \mathbf{y} to real numbers such that:
 - $d(\mathbf{x}, \mathbf{y}) \geq 0$
 - $d(\mathbf{x}, \mathbf{y}) = 0$ iff $\mathbf{x} = \mathbf{y}$
 - $d(\mathbf{x}, \mathbf{y}) = d(\mathbf{y}, \mathbf{x})$
 - $d(\mathbf{x}, \mathbf{y}) \leq d(\mathbf{x}, \mathbf{z}) + d(\mathbf{z}, \mathbf{y})$ (triangle inequality)
- Jaccard distance for sets = 1 - Jaccard similarity
- Cosine distance for vectors = angle between the vectors
- Euclidean distances:
 - L_2 norm: $d(\mathbf{x}, \mathbf{y})$ = square root of the sum of the squares of the differences between \mathbf{x} and \mathbf{y} in each dimension
 - The most common notion of “distance”
 - L_1 norm: sum of absolute value of the differences in each dimension
 - *Manhattan distance* = distance if you travel along coordinates only

Families of Hash Functions

- For Min-Hashing signatures, we got a **Min-Hash function for each permutation of rows**
- A “hash function” is any function that allows us to say whether two elements are “equal”
 - **Shorthand:** $h(x) = h(y)$ means “*h says x and y are equal*”
- A **family** of hash functions is any set of hash functions from which we can ***pick one at random efficiently***
 - **Example:** The set of Min-Hash functions generated from permutations of rows

Locality-Sensitive (LS) Families

- Suppose we have a space S of points with a distance measure $d(x,y)$

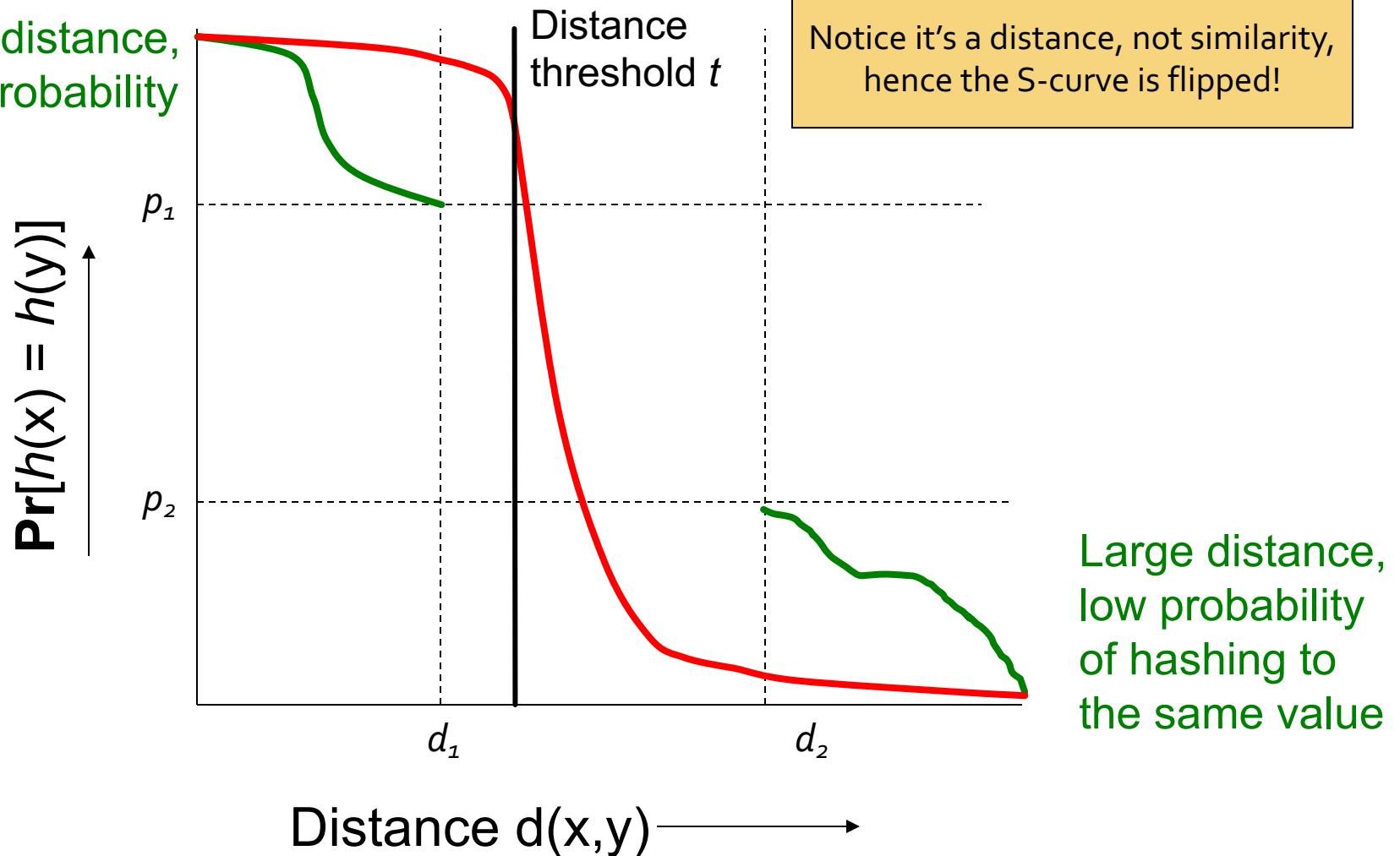
Critical assumption

- 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 :
 - If $d(x, y) \leq d_1$, then the probability over all $h \in H$, that $h(x) = h(y)$ is at least p_1
 - If $d(x, y) \geq d_2$, then the probability over all $h \in H$, that $h(x) = h(y)$ is at most p_2

With a LS Family we can do LSH!

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

Small distance,
high probability



Example of LS Family: Min-Hash

- Let:

- S = space of all sets,
- d = Jaccard distance,
- H is family of Min-Hash functions for all permutations of rows
- Then for any hash function $h \in H$:

$$\Pr[h(x) = h(y)] = 1 - d(x, y)$$

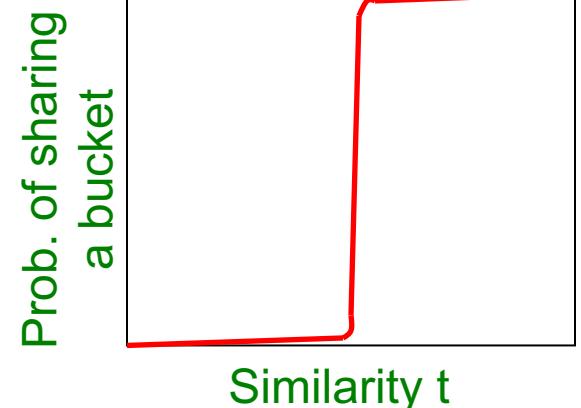
- Simply restates theorem about Min-Hashing in terms of distances rather than similarities

Example: LS Family – (2)

- **Claim:** Min-hash H is a $(1/3, 2/3, 2/3, 1/3)$ -sensitive family for S and d .
 - If distance $\leq 1/3$
(so similarity $\geq 2/3$)
 - Then probability
that Min-Hash values
agree is $\geq 2/3$
- For Jaccard similarity, Min-Hashing gives a $(d_1, d_2, (1-d_1), (1-d_2))$ -sensitive family for any $d_1 < d_2$

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
- Can do LSH with any (d_1, d_2, p_1, p_2) -sensitive family!
- Two constructions:
 - AND construction like “rows in a band”
 - OR construction like “many bands”



Amplifying Hash Functions: AND and OR

AND of Hash Functions

- Given family H , construct family H' consisting of r functions from H
- For $h = [h_1, \dots, h_r]$ in H' , we say $h(x) = h(y)$ if and only if $h_i(x) = h_i(y)$ for all i $1 \leq i \leq r$
 - Note this corresponds to creating a band of size r

- 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 the fact that h_i 's are independent

Also lowers probability
for small distances (**Bad**)

Lowers probability for
large distances (**Good**)

Subtlety Regarding Independence

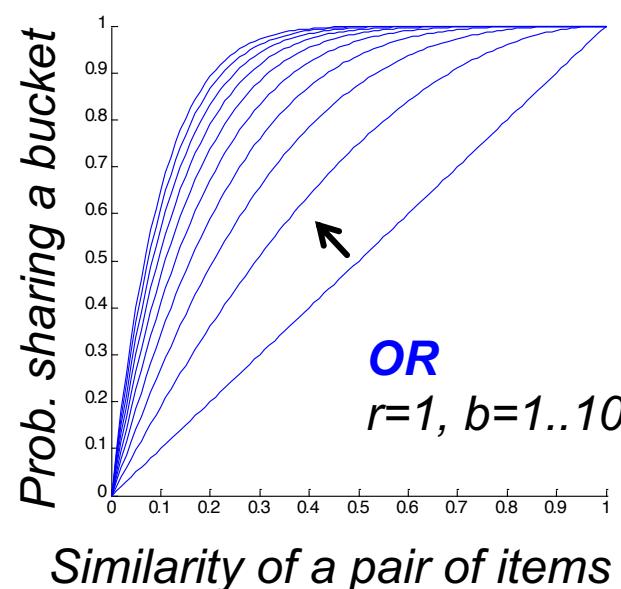
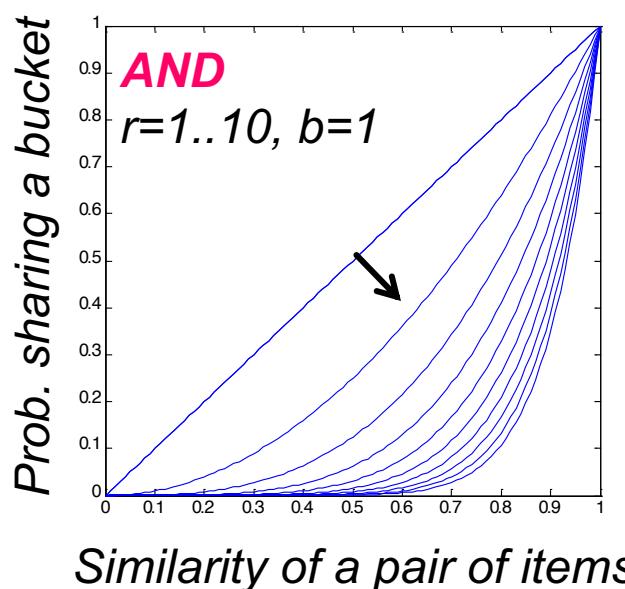
- Independence of hash functions (HFs) really means that the prob. of two HFs saying “yes” is the product of each saying “yes”
 - But two particular hash functions could be highly correlated
 - For example, in Min-Hash if their permutations agree in the first one million entries
 - However, the probabilities in definition of a LSH-family are over all possible members of H, H' (i.e., average case and not the worst case)

OR of Hash Functions

- Given family H , construct family H' consisting of b functions from H
- For $h = [h_1, \dots, h_b]$ in H' ,
 $h(x) = h(y)$ if and only if $h_i(x) = h_i(y)$ for at least 1 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
- **Proof:** Use the fact that h_i 's are independent
 - Raises probability for small distances (Good)
 - Raises probability for large distances (Bad)

Effect of AND and OR Constructions

- **AND** makes all probs. **shrink**, but by choosing r correctly, we can make the lower prob. approach 0 while the higher does not
- **OR** makes all probs. **grow**, but by choosing b correctly, we can make the higher prob. approach 1 while the lower does not



Combine AND and OR Constructions

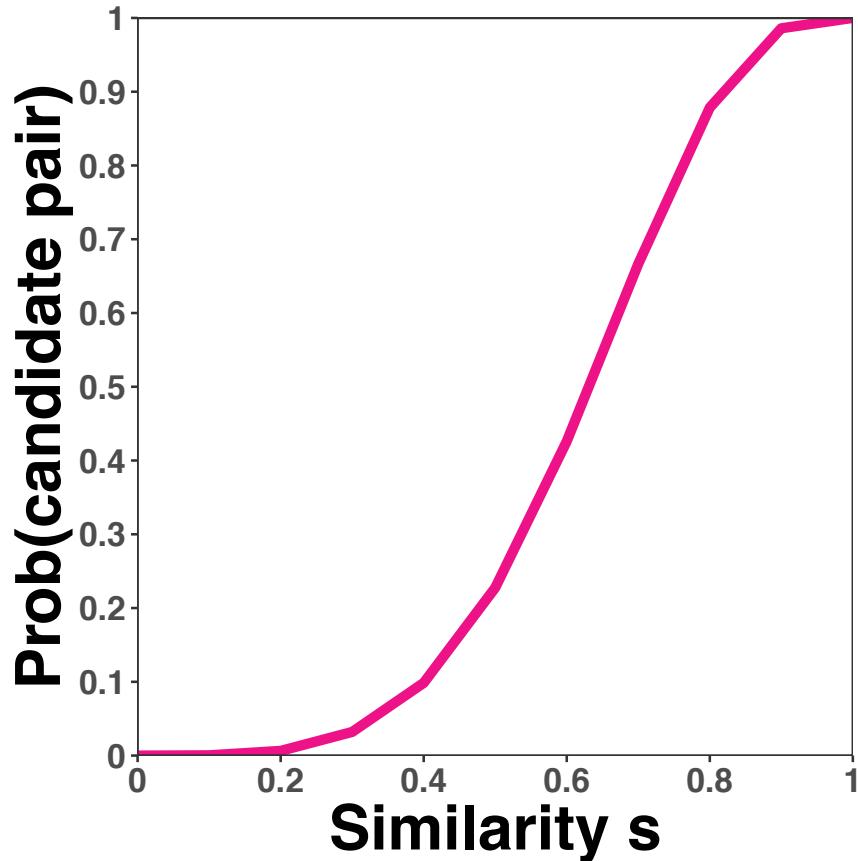
- By choosing b and r correctly, we can make the lower probability approach 0 while the higher approaches 1
- As for the signature matrix, we can use the AND construction followed by the OR construction
 - Or vice-versa
 - Or any sequence of AND's and OR's alternating

Composing Constructions

- ***r*-way AND followed by *b*-way OR construction**
 - Exactly what we did with Min-Hashing
 - AND: If bands match in all r values hash to same bucket
 - OR: Cols that have ≥ 1 common bucket \rightarrow Candidate
- Take points x and y s.t. $Pr[h(x) = h(y)] = s$
 - H will make (x,y) a candidate pair with prob. s
- Construction makes (x,y) a candidate pair with probability **$1-(1-s^r)^b$** **The S-Curve!**
 - 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 $1-(1-s^4)^4$

s	p=1-(1-s⁴)⁴
.2	.0064
.3	.0320
.4	.0985
.5	.2275
.6	.4260
.7	.6666
.8	.8785
.9	.9860

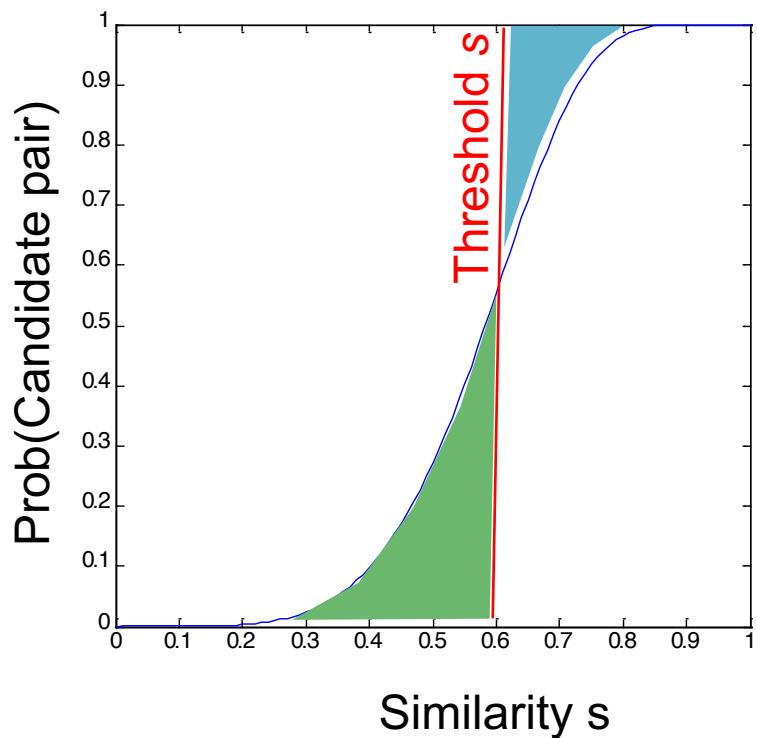


$r = 4, b = 4$ transforms a (.2,.8,.8,.2)-sensitive family into a (.2,.8,.8785,.0064)-sensitive family.

How to choose r and b

Picking r and b : The S-curve

- Picking r and b to get desired performance
 - 50 hash-functions ($r = 5, b = 10$)



Blue area X: False Negative rate

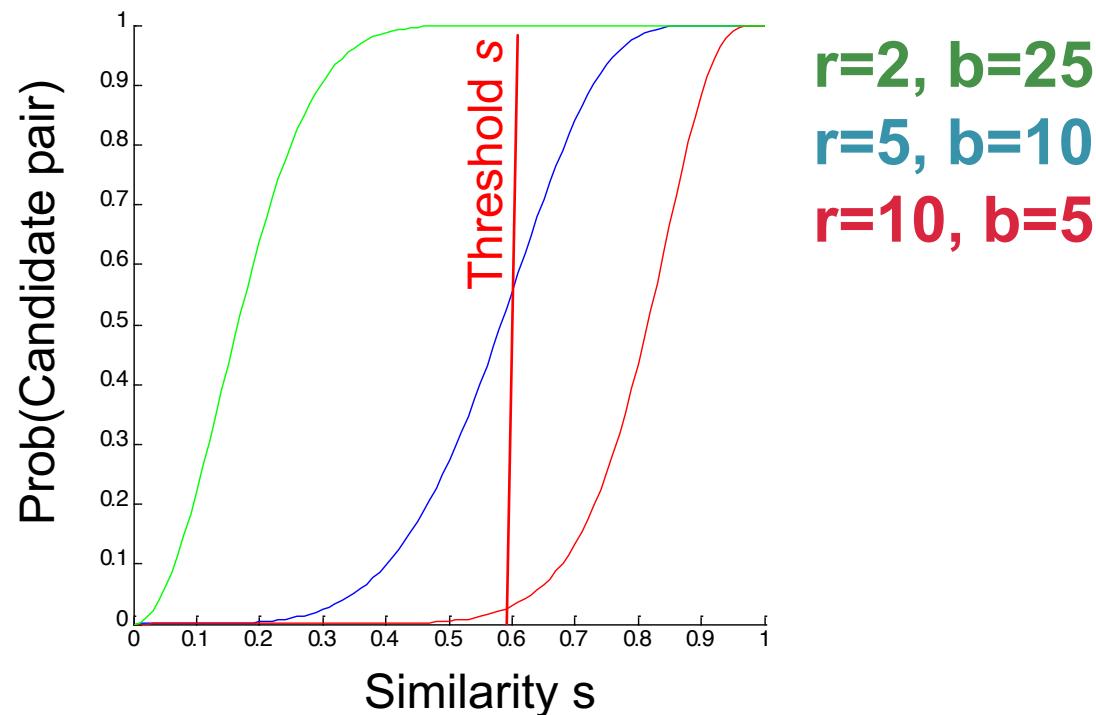
These are pairs with $\text{sim} > s$ but the **X** fraction won't share a band and then will **never become candidates**. This means we will never consider these pairs for (slow/exact) similarity calculation!

Green area Y: False Positive rate

These are pairs with $\text{sim} < s$ but we will consider them as candidates. This is not too bad, we will consider them for (slow/exact) similarity computation and discard them.

Picking r and b : The S-curve

- Picking r and b to get desired performance
 - 50 hash-functions ($r * b = 50$)

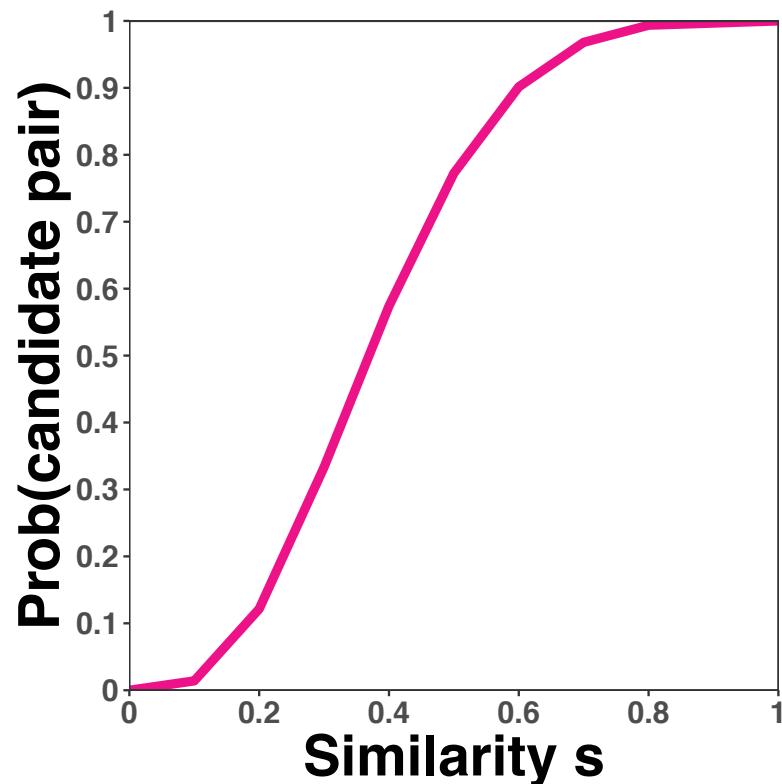


OR-AND Composition

- Apply a **b** -way **OR** construction followed by an **r** -way **AND** construction
- Transforms similarity s (probability p) into $(1-(1-s)^b)^r$
 - **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 $(1-(1-s)^4)^4$

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



The example transforms a (.2,.8,.8,.2)-sensitive family into a (.2,.8,.9936,.1215)-sensitive family

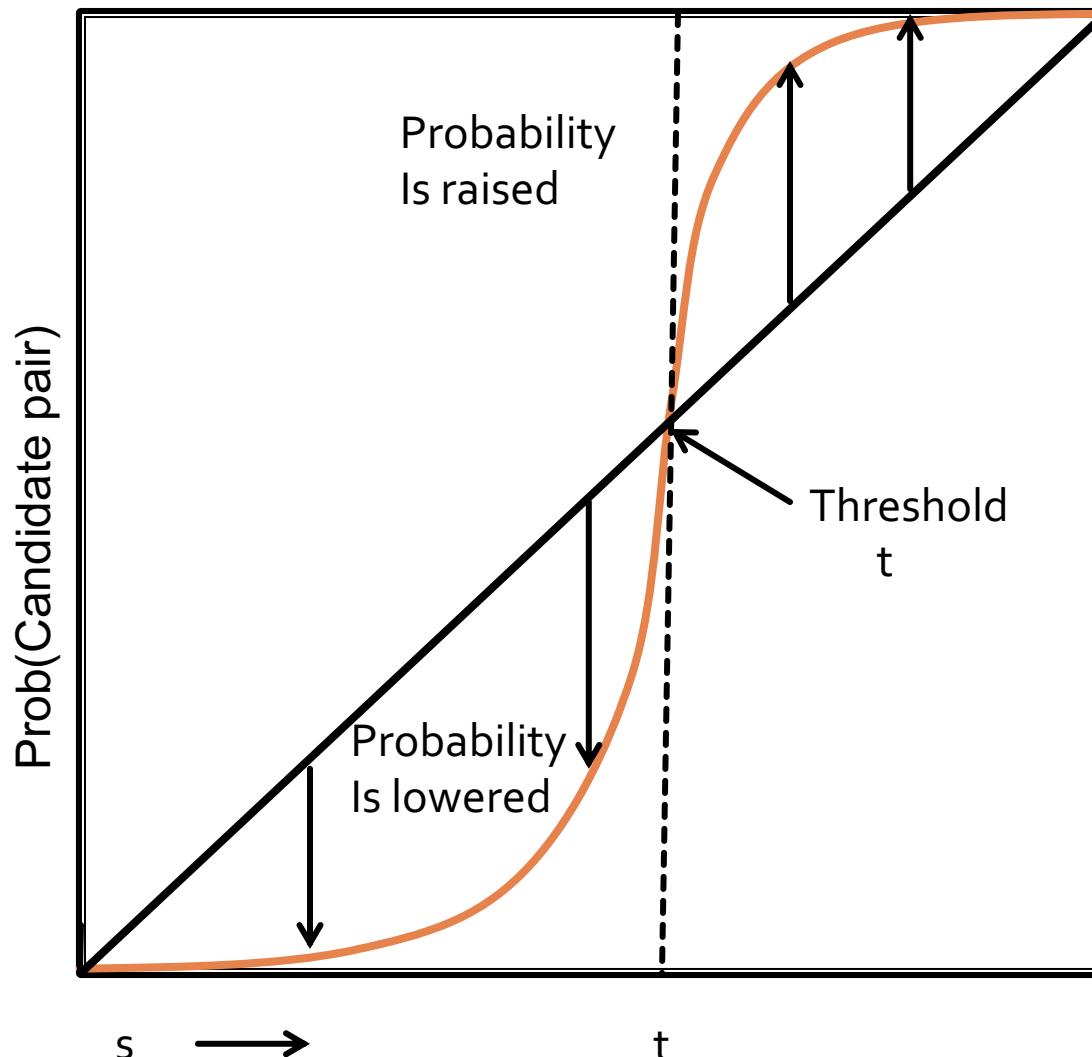
Cascading Constructions

- **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 (=4*4*4*4) of the original hash functions

General Use of S-Curves

- For each AND-OR S-curve $1-(1-s^r)^b$, there is a *threshold* t , for which $1-(1-t^r)^b = t$
- Above t , high probabilities are increased; below t , low probabilities are decreased
- You improve the sensitivity as long as the low probability is less than t , and the high probability is greater than t
 - Iterate as you like
- Similar observation for the OR-AND type of S-curve: $(1-(1-s)^b)^r$

Visualization of Threshold



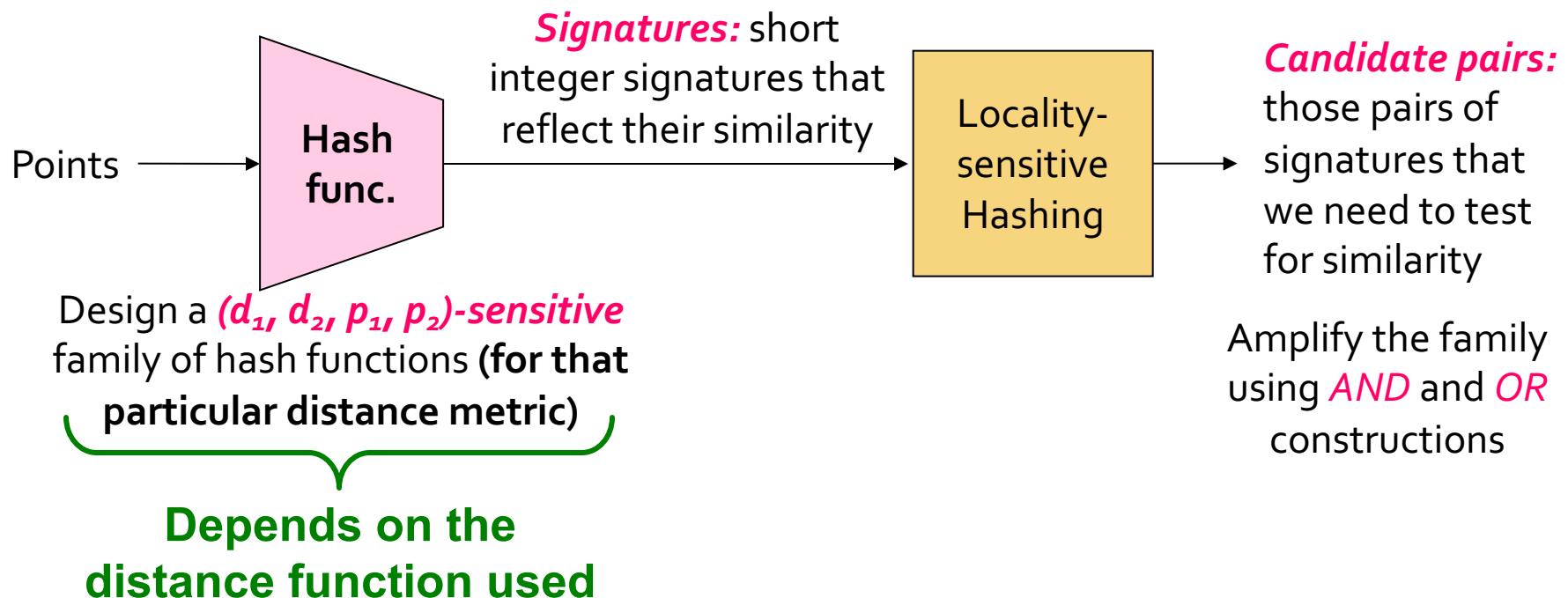
Summary

- Pick any two distances $d_1 < d_2$
- Start with a $(d_1, d_2, (1-d_1), (1-d_2))$ -sensitive family
- Apply constructions to **amplify** (d_1, d_2, p_1, p_2) -sensitive family, where p_1 is almost 1 and p_2 is almost 0
- **The closer to 0 and 1 we get, the more hash functions must be used!**

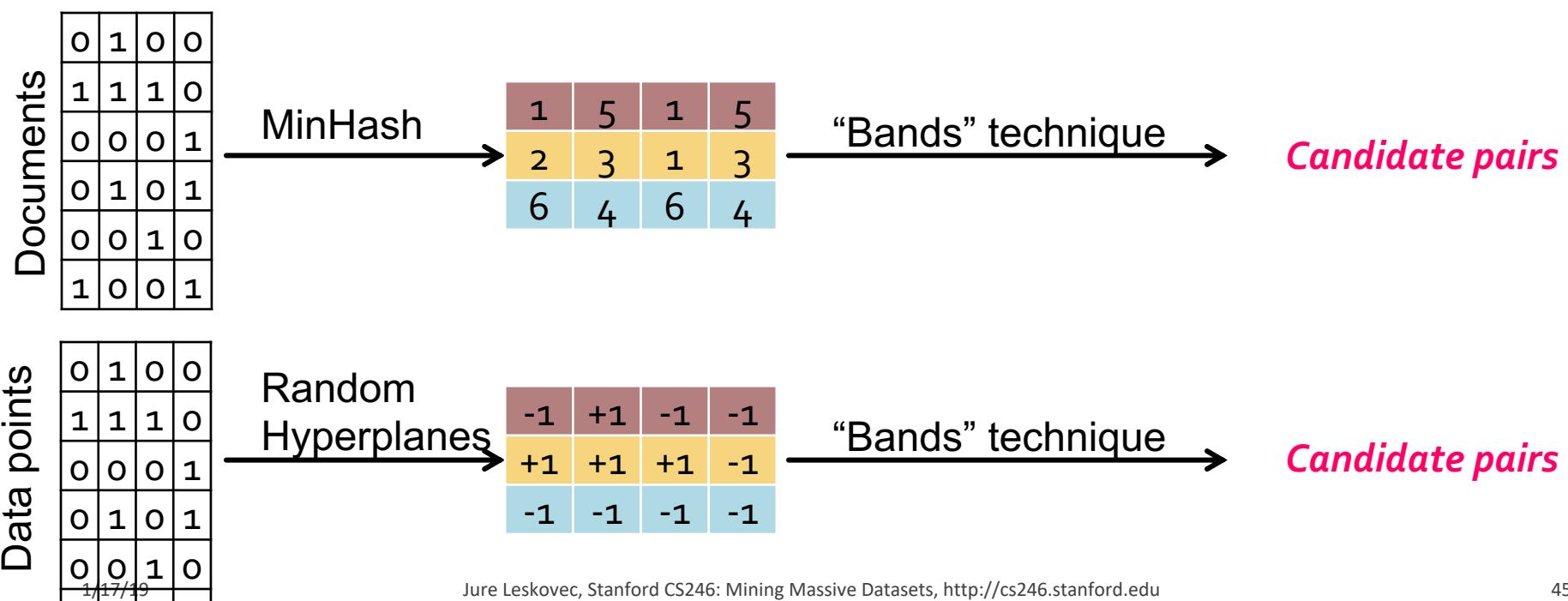
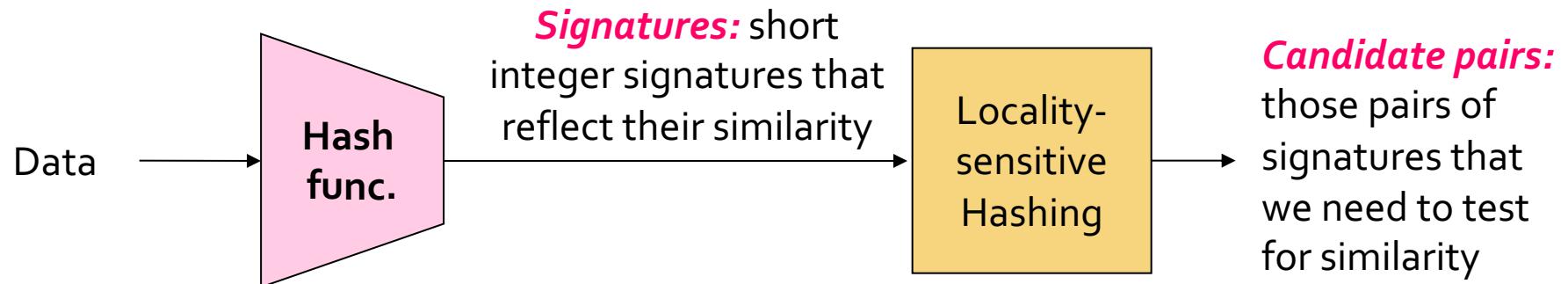
LSH for other distance metrics

LSH for other Distance Metrics

- LSH methods for other distance metrics:
 - Cosine distance: Random hyperplanes
 - Euclidean distance: Project on lines



Summary of what we will learn

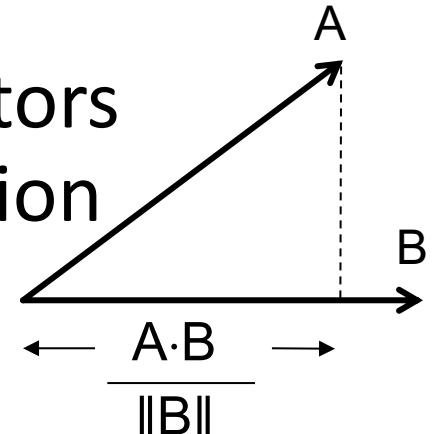


Cosine Distance

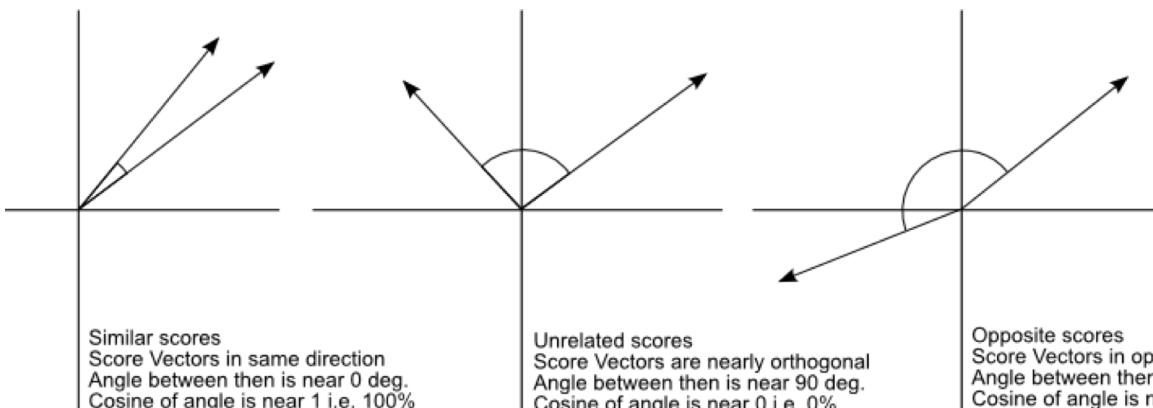
- **Cosine distance** = angle between vectors from the origin to the points in question

$$d(A, B) = \theta = \arccos(A \cdot B / \|A\| \cdot \|B\|)$$

- Has range $[0, \pi]$ (equivalently $[0, 180^\circ]$)
- Can divide θ by π to have distance in range $[0, 1]$
- **Cosine similarity** = $1 - d(A, B)$



- But often defined as **cosine sim**: $\cos(\theta) = \frac{A \cdot B}{\|A\| \|B\|}$



- Has range $-1 \dots 1$ for general vectors
- Range $0..1$ for non-negative vectors (angles up to 90°)

LSH for Cosine Distance

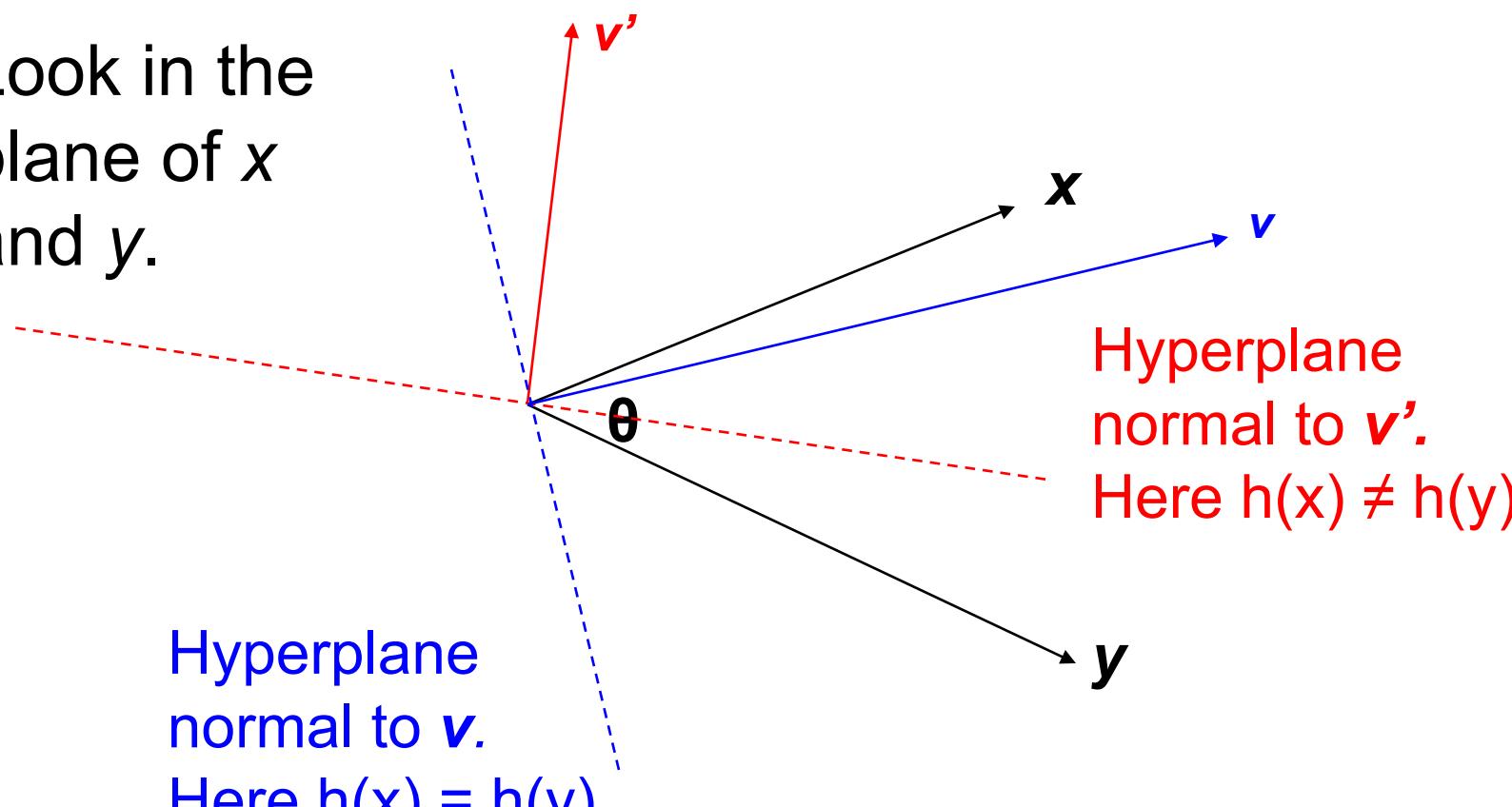
- For **cosine distance**, there is a technique called **Random Hyperplanes**
 - Technique similar to Min-Hashing
- **Random Hyperplanes** method is a $(d_1, d_2, (1-d_1/\pi), (1-d_2/\pi))$ -*sensitive* family for any d_1 and d_2
- **Reminder:** (d_1, d_2, p_1, p_2) -*sensitive*
 1. If $d(x,y) \leq d_1$, then prob. that $h(x) = h(y)$ is at least p_1
 2. If $d(x,y) \geq d_2$, then prob. that $h(x) = h(y)$ is at most p_2

Random Hyperplanes

- Each vector v determines a hash function h_v , with two buckets
- $h_v(x) = +1$ if $v \cdot x \geq 0$; $= -1$ if $v \cdot x < 0$
- LS-family H = set of all functions derived from any vector
- **Claim:** For points x and y ,
$$\Pr[h(x) = h(y)] = 1 - d(x,y) / \pi$$

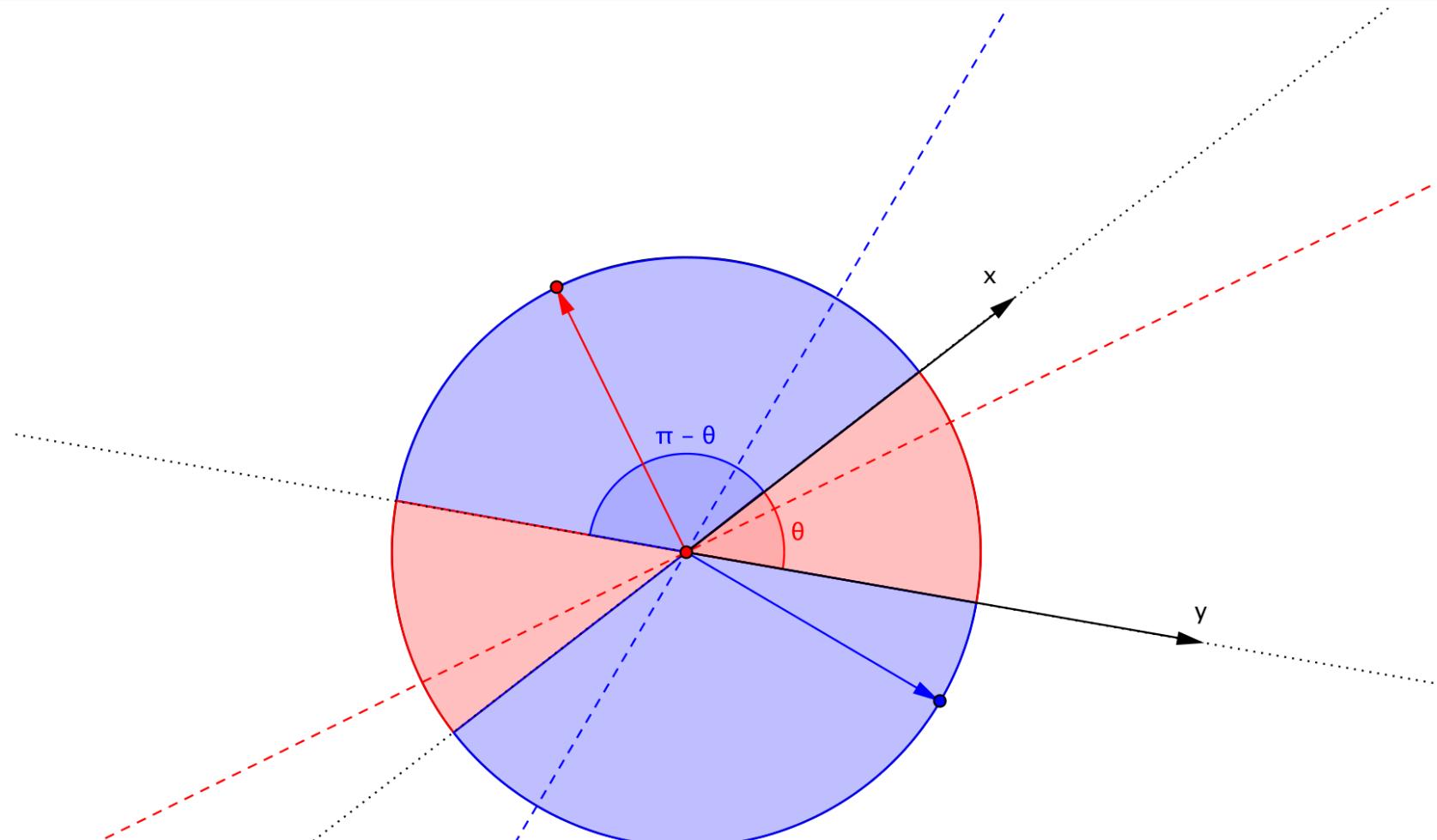
Proof of Claim

Look in the plane of x and y .



Note: what is important is that hyperplane is outside the angle, not that the vector is inside.

Proof of Claim



So: Prob[Red case] = θ / π

So: $P[h(x)=h(y)] = 1 - \theta/\pi = 1 - d(x,y)/\pi$

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 we used the Min-Hash signatures for Jaccard distance
- Amplify using **AND/OR** constructions

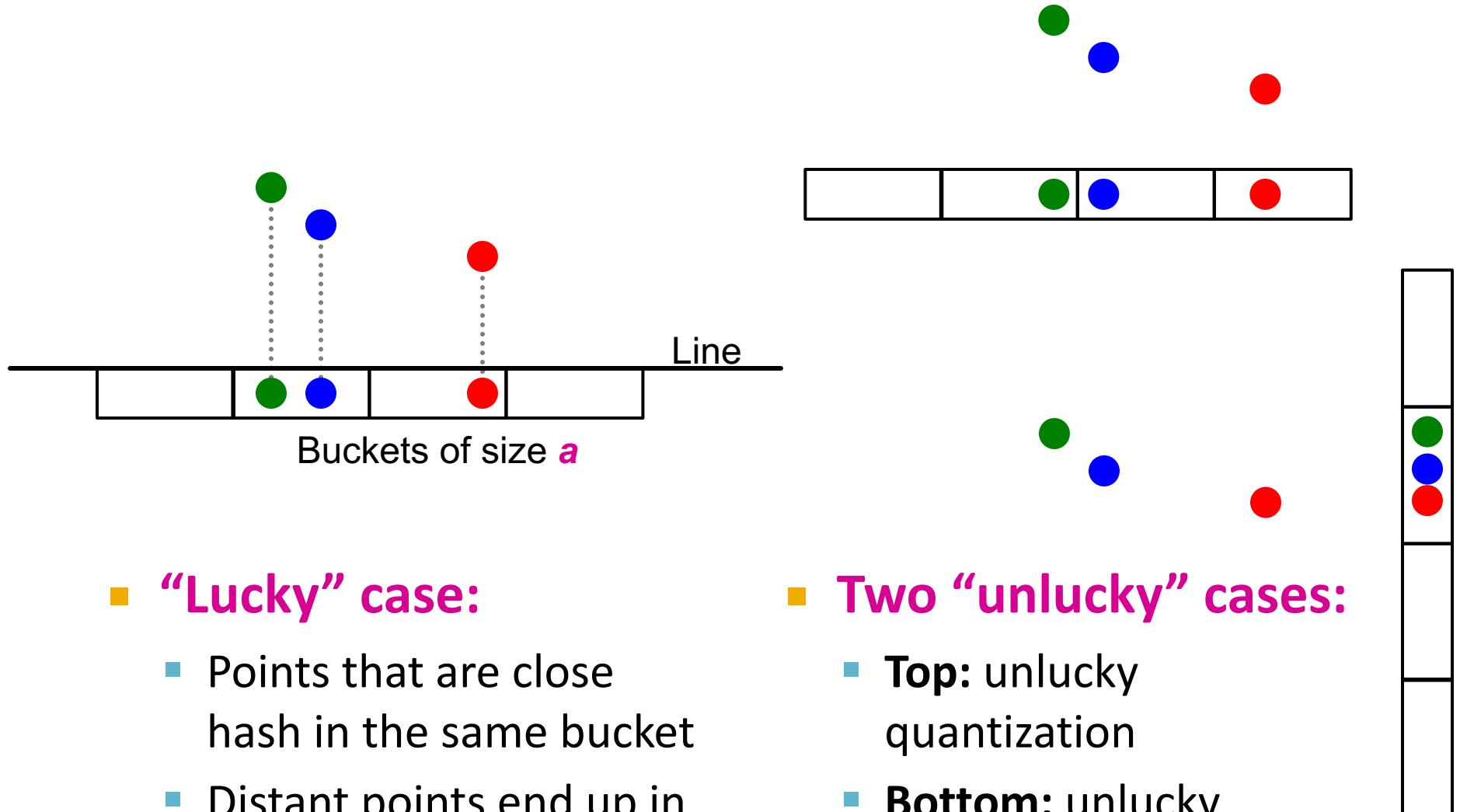
How to pick random vectors?

- Expensive to pick a random vector in M dimensions for large M
 - Would have to generate M random numbers
- **A more efficient approach**
 - It suffices to consider only vectors v consisting of +1 and -1 components
 - **Why?** Assuming data is random, then vectors of +/-1 cover the entire space evenly (and does not bias in any way)

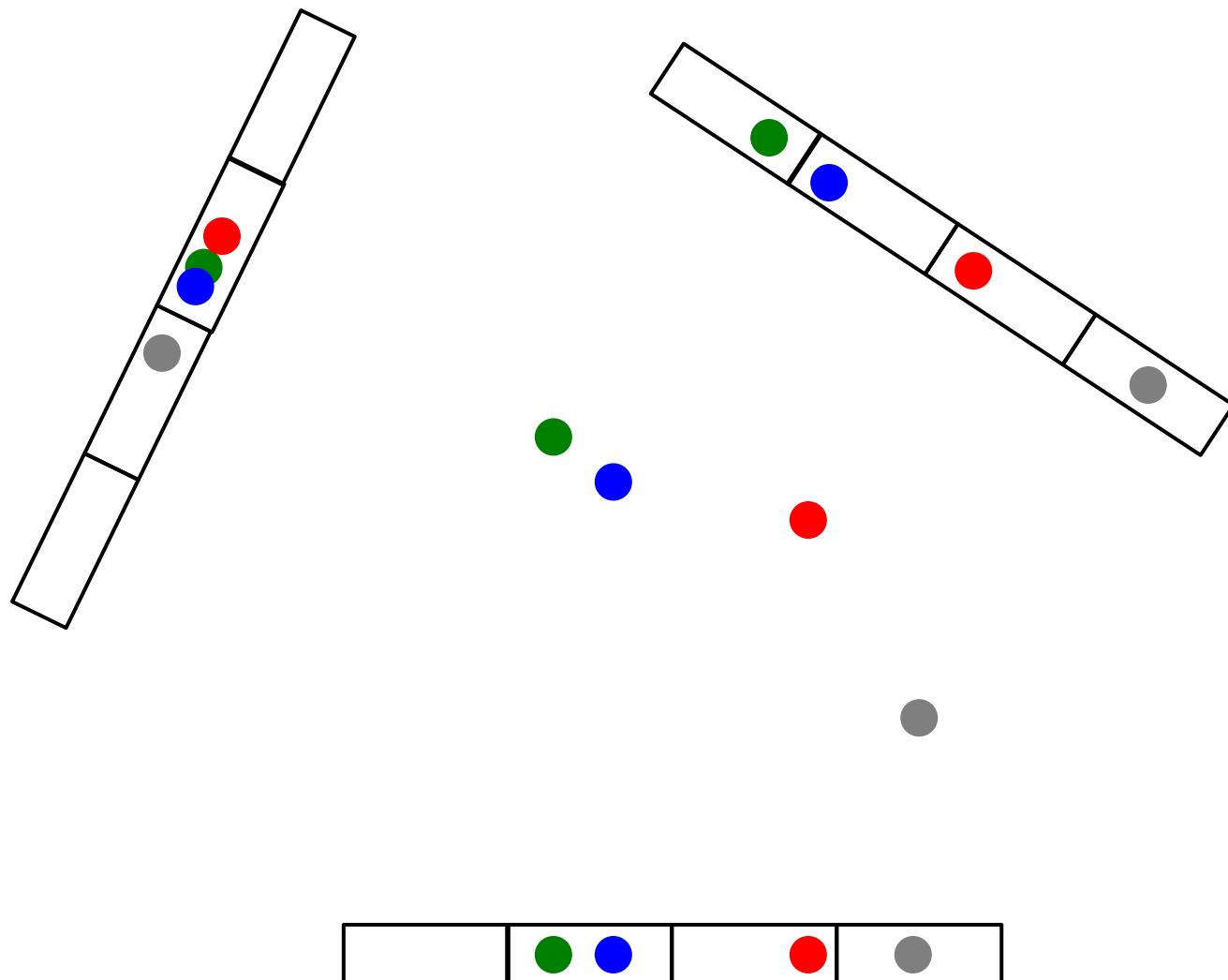
LSH for Euclidean Distance

- **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
 - An element of the “Signature” is a bucket id for that given projection line
- **Nearby points are always close;**
distant points are rarely in same bucket

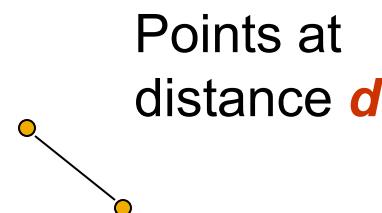
Projection of Points



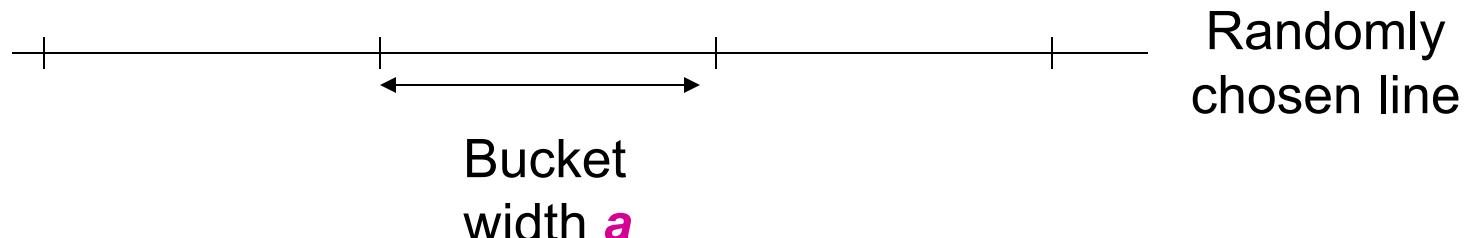
Multiple Projections



Projection of Points

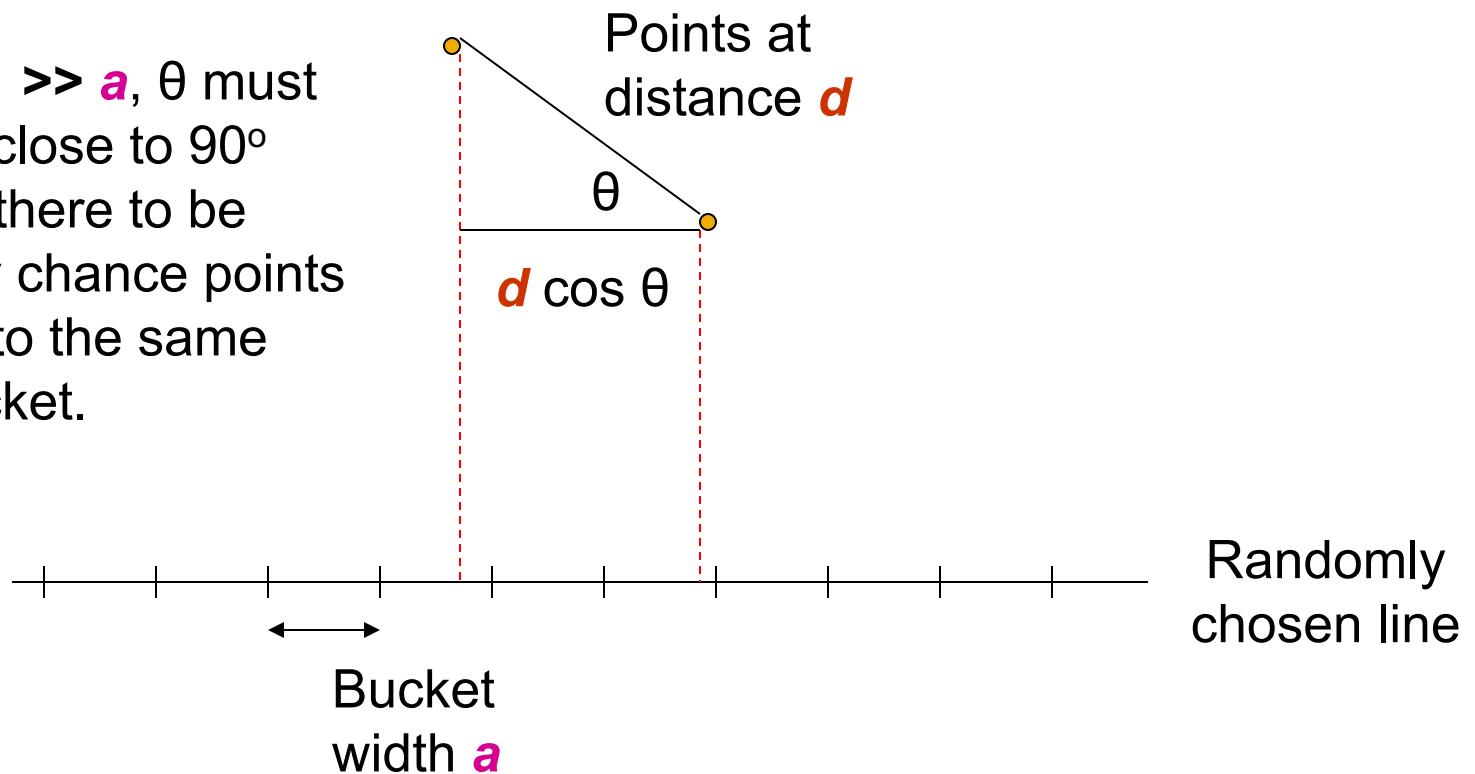


If $d \ll a$, then the chance the points are in the same bucket is at least $1 - d/a$.



Projection of Points

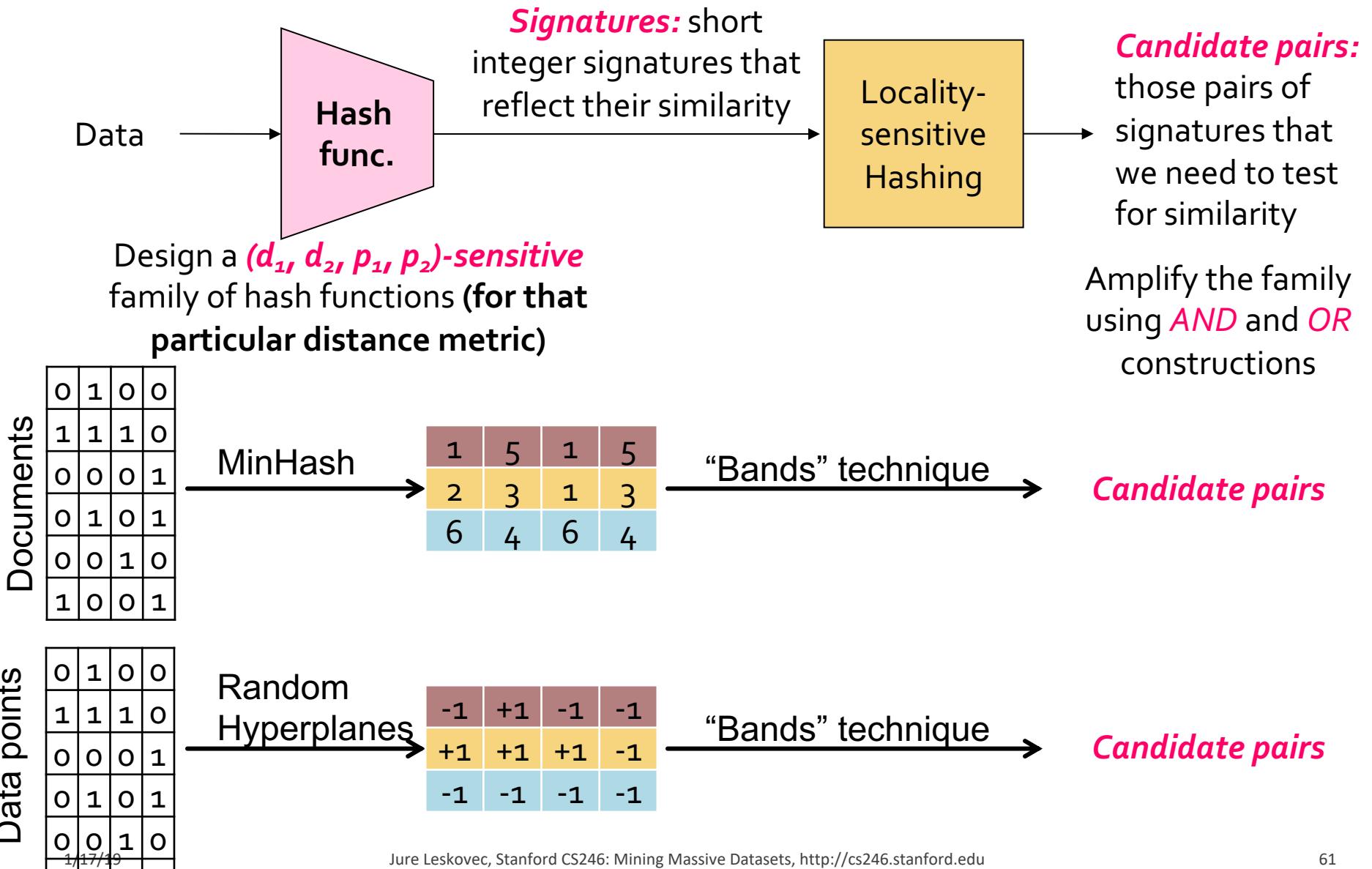
If $d \gg a$, θ must be close to 90° for there to be any chance points go to the same bucket.



A LS-Family for Euclidean Distance

- If points are distance $d \leq a/2$, prob. they are in same bucket $\geq 1 - d/a = 1/2$
- If points are distance $d \geq 2a$ apart, then they can be in the same bucket only if $d \cos \theta \leq a$
 - $\cos \theta \leq 1/2$
 - $60^\circ \leq \theta \leq 90^\circ$, 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**

Summary



Two Important Points

- Property $P(h(C_1)=h(C_2))=\text{sim}(C_1, C_2)$ of hash function h is the essential part of LSH, without which we can't do anything
- LS-hash functions transform data to signatures so that the bands technique (AND, OR constructions) can then be applied