More LSH

LS Families of Hash Functions
LSH for Cosine Distance
Special Approaches for High Jaccard Similarity

Jeffrey D. Ullman
Stanford University

Distance Measures

- Generalized LSH is based on some kind of "distance" between points.
  - Similar points are "close."

- Example:
  Jaccard similarity is not a distance
  1 minus Jaccard similarity is.

Axioms of a Distance Measure

- *d* is a *distance measure* if it is a function from pairs of points to real numbers such that:
  1. $d(x,y) \geq 0$.
  2. $d(x,y) = 0$ iff $x = y$.
  3. $d(x,y) = d(y,x)$.
  4. $d(x,y) \leq d(x,z) + d(z,y)$ (*triangle inequality*).

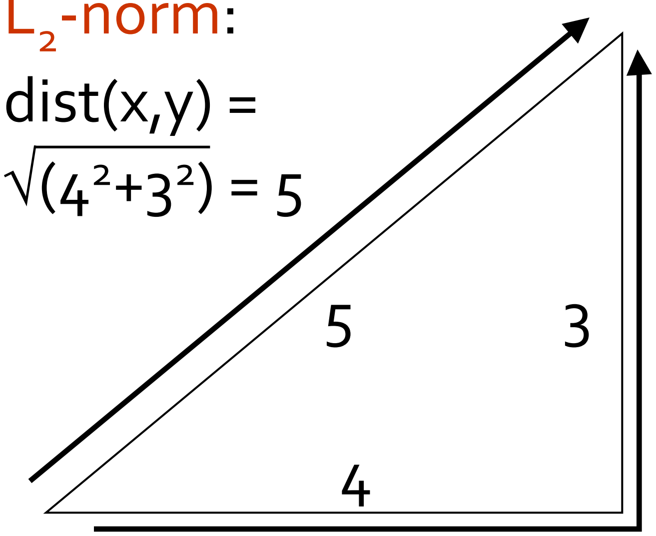
Some Euclidean Distances

- *L_2 norm*: d(x,y) = square root of the sum of the squares of the differences between *x* and *y* in each dimension.
  - The most common notion of "distance."
- *L_1 norm*: sum of the differences in each dimension.
  - *Manhattan distance* = distance if you had to travel along coordinates only.

Examples of Euclidean Distances

L_2-norm:

$dist(x,y) = \sqrt{(4^2+3^2)} = 5$



L_1-norm:

$dist(x,y) = 4+3 = 7$

Some Non-Euclidean Distances

- *Jaccard distance* for sets = 1 minus Jaccard similarity.
- *Cosine distance* for vectors = angle between the vectors.
- *Edit distance* for strings = number of inserts and deletes to change one string into another.

Example: Jaccard Distance

- Consider x = {1,2,3,4} and y = {1,3,5}
- Size of intersection = 2; size of union = 5, Jaccard similarity (not distance) = 2/5.
- d(x,y) = 1 – (Jaccard similarity) = 3/5.

Why J.D. Is a Distance Measure

- $d(x,y) \geq 0$ because $|x \cap y| \leq |x \cup y|$.
- $d(x,x) = 0$ because $x \cap x = x \cup x$.
  - And if $x \neq y$, then the size of $x \cap y$ is strictly less than the size of $x \cup y$.
- $d(x,y) = d(y,x)$ because union and intersection are symmetric.
- $d(x,y) \leq d(x,z) + d(z,y)$ trickier – next slide.

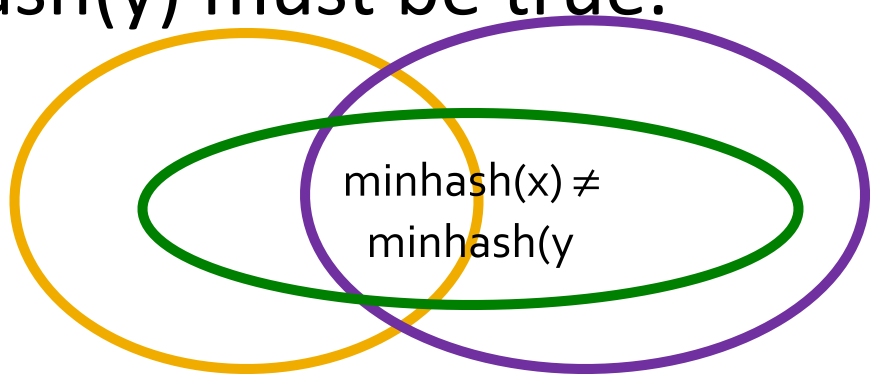
Triangle Inequality for J.D.

$$\underbrace{1 - \frac{|x \cap z|}{|x \cup z|}}_{d(x,z)} + \underbrace{1 - \frac{|y \cap z|}{|y \cup z|}}_{d(z,y)} \geq \underbrace{1 - \frac{|x \cap y|}{|x \cup y|}}_{d(x,y)}$$

- Remember: $|a \cap b|/|a \cup b|$ = probability that minhash(a) = minhash(b).
- Thus, $1 - |a \cap b|/|a \cup b|$ = probability that minhash(a) $\neq$ minhash(b).

Triangle Inequality – (2)

- Claim: prob[minhash(x) ≠ minhash(y)] ≤ prob[minhash(x) ≠ minhash(z)] + prob[minhash(z) ≠ minhash(y)]
- Proof: whenever minhash(x) ≠ minhash(y), at least one of minhash(x) ≠ minhash(z) and minhash(z) ≠ minhash(y) must be true.



minhash(x) ≠ minhash(z)

minhash(z) ≠ minhash(y)

Cosine Distance

- Think of a point as a vector from the origin (0,0,…,0) to its location.
- Two points' vectors make an angle, whose cosine is the normalized dot-product of the vectors: $p_1.p_2/|p_2||p_1|$.
  - Example: p_1 = 00111; p_2 = 10011.
  - $p_1.p_2 = 2$; $|p_1| = |p_2| = \sqrt{3}$.
  - $\cos(\theta) = 2/3$; θ is about 48 degrees.

Edit Distance

- The *edit distance* of two strings is the number of inserts and deletes of characters needed to turn one into the other.
- An equivalent definition:
  $d(x,y) = |x| + |y| - 2|LCS(x,y)|$.
  - LCS = *longest common subsequence* = any longest string obtained both by deleting from *x* and deleting from *y*.

Example: Edit Distance

- *x* = *abcde* ; *y* = *bcduve*.
- Turn *x* into *y* by deleting *a*, then inserting *u* and *v* after *d*.
  - Edit distance = 3.
- Or, computing edit distance through the LCS, note that LCS(x,y) = *bcde*.
- Then: |x| + |y| - 2|LCS(x,y)| = 5 + 6 –2*4 = 3 = edit distance.

LSH Families of Hash Functions

Definition
Combining hash functions
Making steep S-Curves

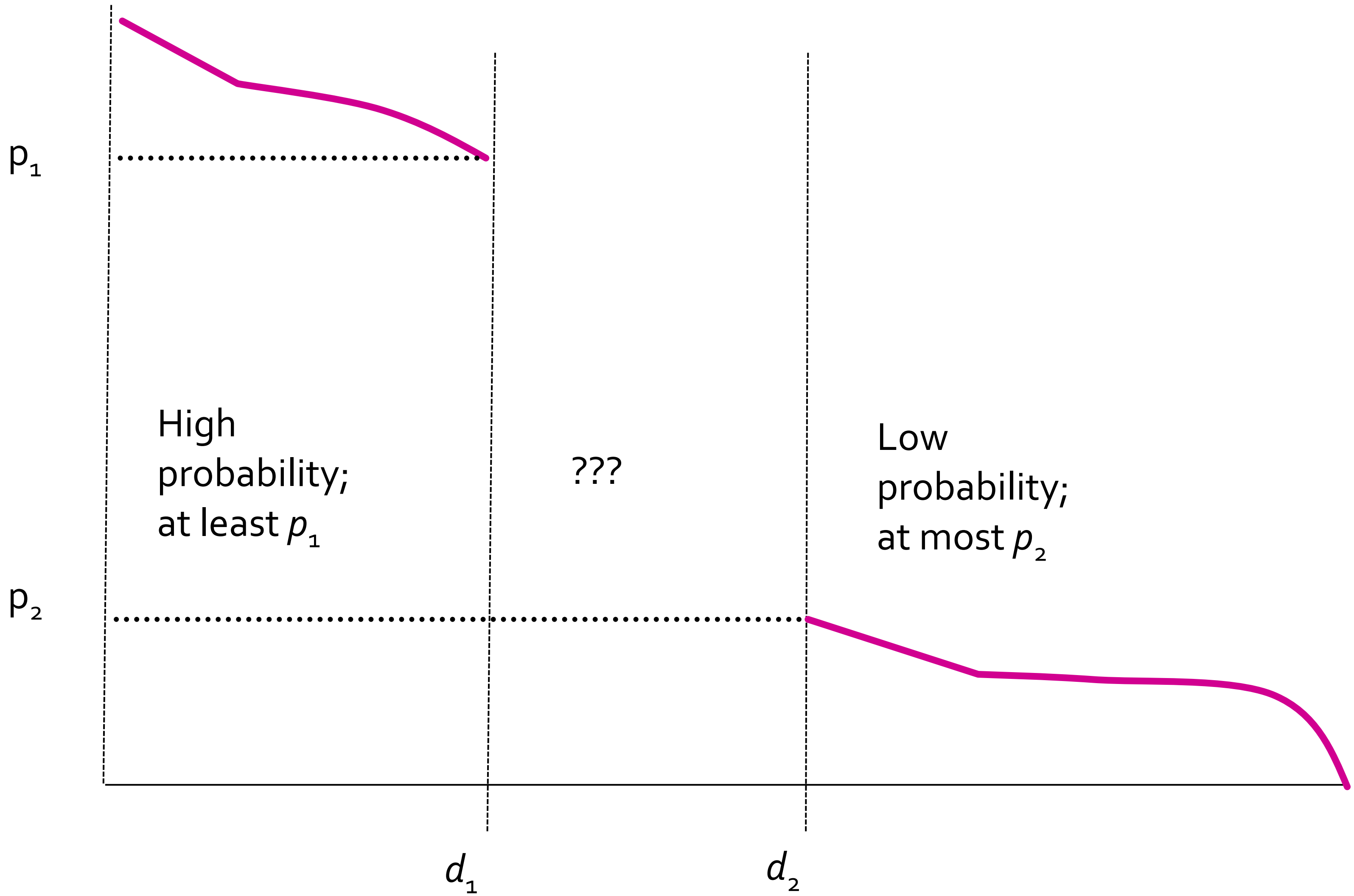
Hash Functions Decide Equality

- There is a subtlety about what a "hash function" is, in the context of LSH families.
- A hash function h really takes two elements x and y, and returns a decision whether x and y are candidates for comparison.
- Example: the family of minhash functions computes minhash values and says "yes" iff they are the same.
- Shorthand: "h(x) = h(y)" means h says "yes" for pair of elements x and y.

LSH Families Defined

- 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 the probability over all *h* in **H**, that $h(x) = h(y)$ is at least p_1.
  2. 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.

LS Families: Illustration



Example: LS Family

- Let:
  - S = subsets of some universal set,
  - d = Jaccard distance,
  - **H** formed from the minhash functions for all permutations of the universal set.
- Then Prob[h(x)=h(y)] = 1-d(x,y).
  - Restates theorem about Jaccard similarity and minhashing in terms of Jaccard distance.

Example: LS Family – (2)

- Claim: **H** is a $(\boxed{1/3}, \boxed{3/4}, \boxed{2/3}, \boxed{1/4})$-sensitive family for *S* and *d*.

If distance $\leq 1/3$
(so similarity $\geq 2/3$)

If distance $\geq 3/4$
(so similarity $\leq 1/4$)

Then probability
that minhash values
agree is $\geq 2/3$

Then probability
that minhash values
agree is $\leq 1/4$

For Jaccard similarity, minhashing gives us a $(d_1,d_2,(1-d_1),(1-d_2))$-sensitive family for any $d_1 < d_2$.

Amplifying an LSH-Family

- The "bands" technique we learned for signature matrices carries over to this more general setting.
  - Goal: the "S-curve" effect seen there.
- AND construction like "rows in a band."
- OR construction like "many bands."

AND of Hash Functions

- Given family **H**, construct family **H'** whose members each consist 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'** whose members each consist 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.

Effect of AND and OR Constructions

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

Composing Constructions

- 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.

AND-OR Composition

- Each of the two probabilities *p* is transformed into $1-(1-p^r)^b$.
  - The "S-curve" studied before.
- 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

- Each of the two probabilities *p* is transformed 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.

Cascading Constructions

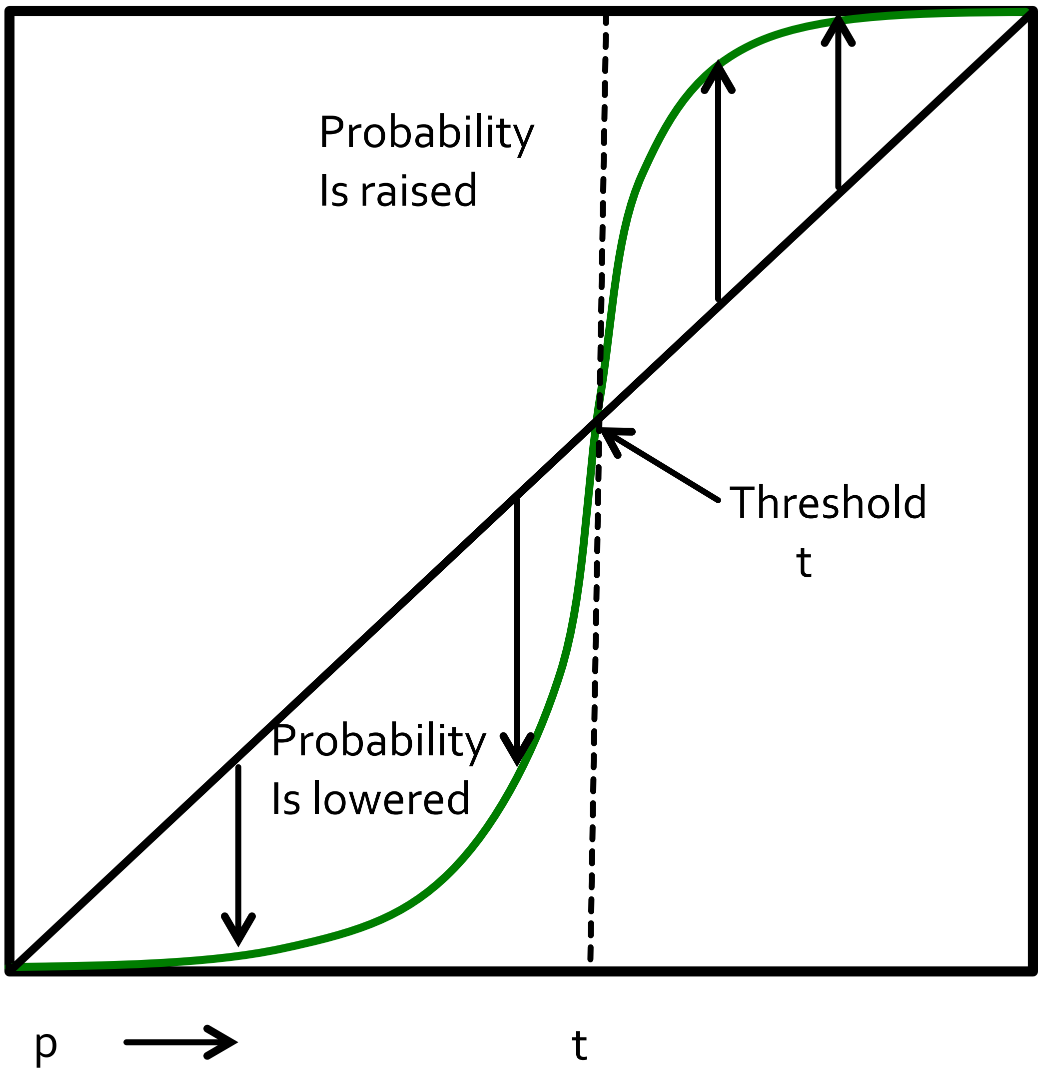
- Example: Apply the (4,4) OR-AND construction followed by the (4,4) AND-OR construction.
- Transforms a (.2,.8,.8,.2)-sensitive family into a (.2,.8,.9999996,.0008715)-sensitive family.

- Check remaining pairs are similar documents:
  - With 100,000 documents and 100 true pairs, we get 187 candidates with 87 false positives!

General Use of S-Curves

- For each AND-OR S-curve $1-(1-p^r)^b$, 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: $(1-(1-p)^b)^r$.

Visualization of Threshold



An LSH Family for Cosine Distance

Random Hyperplanes
Sketches (Signatures)

Random Hyperplanes – (1)

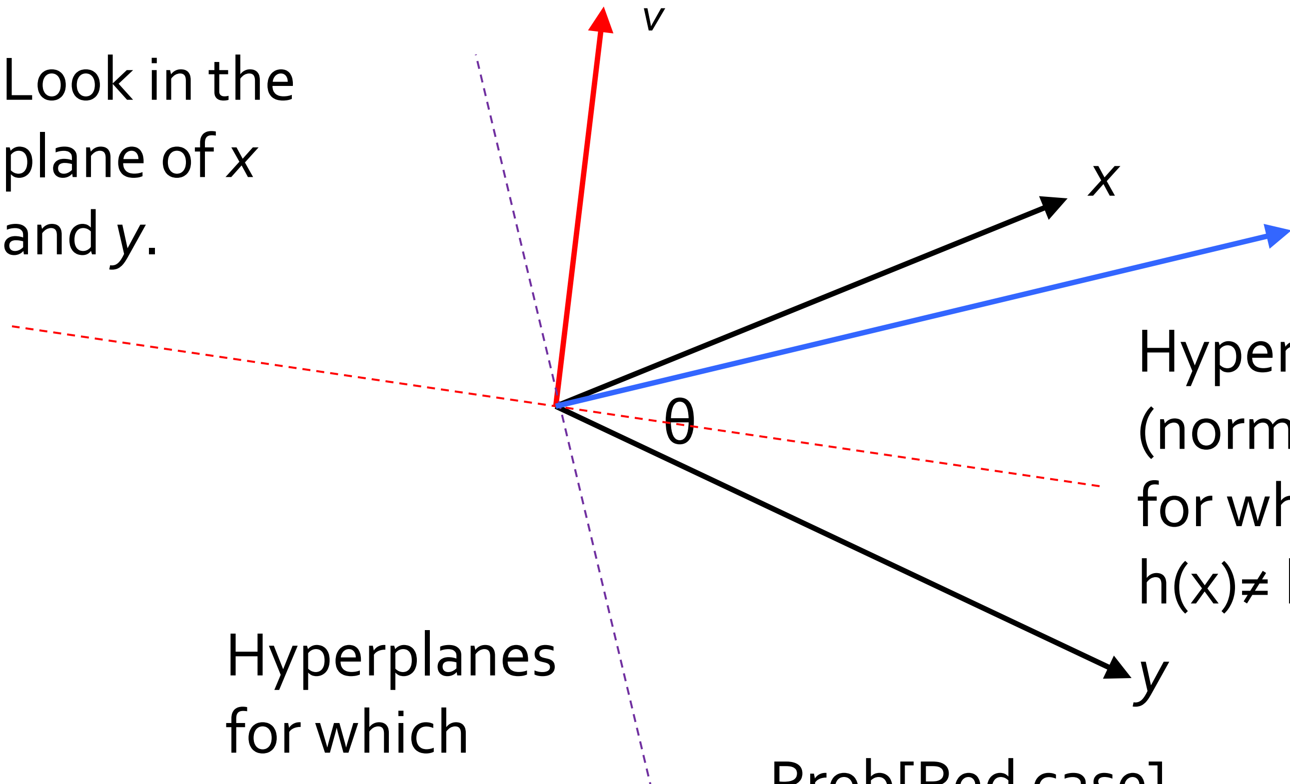
- For cosine distance, there is a technique analogous to minhashing for generating a $(d_1,d_2,(1\text{-}d_1/180),(1\text{-}d_2/180))$-sensitive family for any d_1 and d_2.
- Called *random hyperplanes*.

Random Hyperplanes – (2)

- Each vector v determines a hash function h_v with two buckets.
- $h_v(x) = +1$ if $v * x > 0$,
  $h_v(x) = -1$ if $v * x < 0$.
- LS-family **H** = set of all functions derived from any vector v.
- Claim: Prob[h(x)=h(y)] = 1 – (angle between x and y divided by 180).

Proof of Claim

Look in the plane of *x* and *y*.



Hyperplanes (normal to *v*) for which $h(x) \neq h(y)$

Hyperplanes for which $h(x) = h(y)$

Prob[Red case] $= \theta/180$

Signatures for Cosine Distance

- Pick some number of vectors, and hash your data for each vector.
- The result is a signature (*sketch*) of +1's and –1's that can be used for LSH like the minhash signatures for Jaccard distance.
- But you don't have to think this way.
- The existence of the LSH-family is sufficient for amplification by AND/OR.

Simplification

- We need not pick from among all possible vectors *v* to form a component of a sketch.
- It suffices to consider only vectors *v* consisting of +1 and –1 components.

Methods for High Degrees of Jaccard Similarity

Sets Represented by Sorted Strings
Use of String Length
Exploiting Prefixes

Setting: Sets as Strings

- We'll again talk about Jaccard similarity and distance of sets.
- However, now represent sets by strings (lists of symbols):
  1. Order the universal set.
  2. Represent a set by the string of its elements in sorted order.

Example: Shingles

- If the universal set is k-shingles, there is a natural lexicographic order.
- Think of each shingle as a single symbol.
- Then the 2-shingling of abcad, which is the set {ab, bc, ca, ad}, is represented by the list (*string*) [ab, ad, bc, ca] of length 4.

Example: Words

- If we treat a document as a set of words, we could order the words lexicographically.
- Better: Order words lowest-frequency-first.
- Why? We shall bucketize documents based on the early words in their lists.
  - Documents spread over more buckets.

Jaccard and Edit Distances

- Suppose two sets have Jaccard distance J and are represented by strings s_1 and s_2. Let the LCS of s_1 and s_2 have length C and the (insert/delete) edit distance of s_1 and s_2 be E.

  Then:

  - 1-J = Jaccard similarity = C/(C+E).
  - J = E/(C+E).

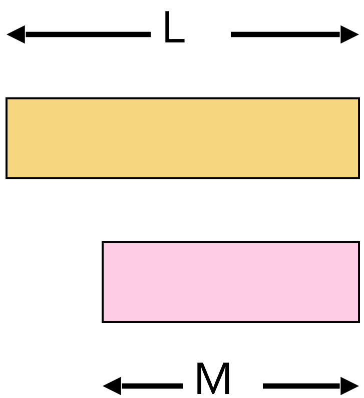
Works because these strings never repeat a symbol, and symbols appear in the same order.

Example: s_1 = acefh; s_2 = bcdegh.
LCS = ceh; C = 3; E = 5; 1-J = 3/8.

Length-Based Indexes

- The simplest thing to do is create an index on the length of strings.
- A set whose string has length L can be Jaccard distance J from a set whose string has length M only if $L \times (1\text{-}J) \leq M \leq L/(1\text{-}J)$.
- Example: if 1-J = 90% (Jaccard similarity), then M is between 90% and 111% of L.

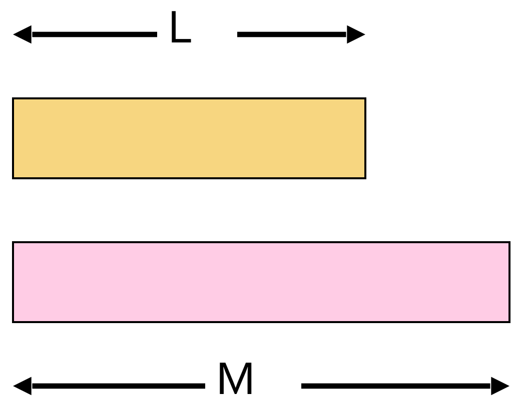
Why the Limit on Lengths?



$1\text{-}J \leq M/L$

$M \geq L \times (1\text{-}J)$

A shortest candidate



$1\text{-}J \leq L/M$

$M \leq L/(1\text{-}J)$

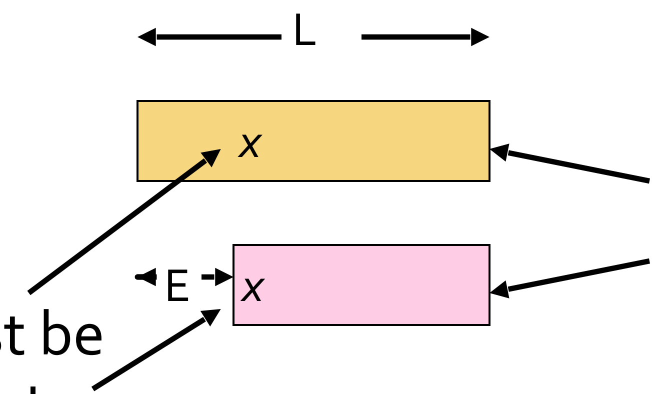
A longest candidate

Prefix-Based Indexing

- Example: If two strings are 90% similar, they must share some symbol in their *prefixes*.
  - These prefixes are of length just above 10% of the length of each string.
- In general: we can base an index on symbols in just the first $\lfloor JL+1 \rfloor$ positions of a string of length L.

Why the Limit on Prefixes?

Suppose a string of length L has E symbols
Before the first match with a second string.



Must be
Equal

Extreme case: second string has none of the first E symbols of the first string, but they agree thereafter.

If two strings do not share any of the first E symbols of the first string, then J ≥ E/L.

Thus, E = JL is possible, but any larger E is impossible. Index E+1 positions.

Indexing Prefixes

- Think of a bucket for each possible symbol.
- Each string of length L is placed in the bucket for the symbols in each of its first $\lfloor JL+1 \rfloor$ positions.

Lookup

- Given a *probe* string *s* of length L, with J the limit on Jaccard distance:

```
for (each symbol a among the
 first ⌊JL+1⌋ positions of s)
   look for other strings in
    the bucket for a;
```

Example: Indexing Prefixes

- Let J = 0.2.
- String abcdef is indexed under *a* and *b*.
  - $\lfloor (0.2)*6 + 1 \rfloor = 2.$
- String acdfg is indexed under *a* and *c*.
  - $\lfloor (0.2)*5 + 1 \rfloor = 2.$
- String bcde is indexed only under *b*.
  - $\lfloor (0.2)*4 + 1 \rfloor = 1.$
- If we search for strings similar to cdef, we need look only in the bucket for *c*.