Finding Similar Items: Locality Sensitive Hashing
Goal: Given a large number (N in the millions or billions) of text documents, find pairs that are “near duplicates”

Application:
- Detect mirror and approximate mirror sites/pages:
  - Don’t want to show both in a web search

Problems:
- Many small pieces of one doc can appear out of order in another
- Too many docs to compare all pairs
- Docs are so large or so many that they cannot fit in main memory
3 Essential Steps for Similar Docs

1. **Shingling**: Convert documents to large sets of items

2. **Minhashing**: Convert large sets into short signatures, while preserving similarity

3. **Locality-sensitive hashing**: Focus on pairs of signatures likely to be from similar documents
The set of strings of length $k$ that appear in the document

**Signatures**: short integer vectors that represent the sets, and reflect their similarity

**Candidate pairs**: those pairs of signatures that we need to test for similarity.
A $k$-shingle (or $k$-gram) for a document is a sequence of $k$ tokens that appears in the document.

- Tokens can be characters, words or something else, depending on application.
- Assume tokens = characters for examples.

**Example:** $k=2$; $D_1 =$ abcab

Set of 2-shingles: $S(D_1) = \{ab, bc, ca\}$

- Represent a doc by the set of hash values of its $k$-shingles.
Document $D_1 = \text{set of } k\text{-shingles } C_1 = S(D_1)$

Equivalently, each document is a 0/1 vector in the space of $k$-shingles
- Each unique shingle is a dimension
- Vectors are very sparse

A natural similarity measure is the Jaccard similarity:

$$Sim(D_1, D_2) = \frac{|C_1 \cap C_2|}{|C_1 \cup C_2|}$$
We can encode sets using 0/1 (bit, boolean) vectors

- One dimension per element in the universal set
- Interpret set intersection as bitwise AND, and set union as bitwise OR

Example: \( C_1 = 1100011; C_2 = 0110010 \)
- Size of intersection = 2; size of union = 5, Jaccard similarity (not distance) = 2/5
- \( d(C_1, C_2) = 1 - \text{(Jaccard similarity)} = 3/5 \)
1. Signatures of columns = small summaries of columns

2. Examine pairs of signatures to find similar signatures
   - Essential: Similarities of signatures & columns are related

3. Optional: Check that columns with similar signatures are really similar

   - Warnings:
     1. Comparing all pairs of signatures may take too much time, even if not too much space
        - A job for Locality-Sensitive Hashing
     2. These methods can produce false negatives, and even false positives (if the optional check is not made)
Key idea: “hash” each column $C$ to a small signature $h(C)$, such that:

1. $h(C)$ is small enough that we can fit a signature in main memory for each column
2. $\text{Sim}(C_1, C_2)$ is the same as the “similarity” of $h(C_1)$ and $h(C_2)$

Goal: Find a hash function $h()$ such that:

- if $\text{Sim}(C_1, C_2)$ is high, then with high prob. $h(C_1) = h(C_2)$
- if $\text{Sim}(C_1, C_2)$ is low, then with high prob. $h(C_1) \neq h(C_2)$

Hash docs into buckets, and expect that “most” pairs of near duplicate docs hash into the same bucket
Min-hashing

- Clearly, the hash function depends on the similarity metric
  - Not all similarity metrics have a suitable hash function
- There is a suitable hash function for Jaccard similarity
  - Min-hashing
Min-hashing

- Imagine the rows of the boolean matrix permuted under random permutation $\pi$

- Define a “hash” function $h_\pi(C) = \text{the number of the first (in the permuted order $\pi$) row in which column } C \text{ has 1:}$
  
  $$h_\pi(C) = \min \pi(C)$$

- Use several (e.g., 100) independent hash functions to create a signature
### Minhashing Example

**Permutation** $\pi$

<table>
<thead>
<tr>
<th>1</th>
<th>4</th>
<th>3</th>
</tr>
</thead>
<tbody>
<tr>
<td>3</td>
<td>2</td>
<td>4</td>
</tr>
<tr>
<td>7</td>
<td>1</td>
<td>7</td>
</tr>
<tr>
<td>6</td>
<td>3</td>
<td>6</td>
</tr>
<tr>
<td>2</td>
<td>6</td>
<td>1</td>
</tr>
<tr>
<td>5</td>
<td>7</td>
<td>2</td>
</tr>
<tr>
<td>4</td>
<td>5</td>
<td>5</td>
</tr>
</tbody>
</table>

**Input matrix**

<table>
<thead>
<tr>
<th></th>
<th>1</th>
<th>0</th>
<th>1</th>
<th>0</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td></td>
</tr>
</tbody>
</table>

**Signature matrix** $M$

<table>
<thead>
<tr>
<th></th>
<th>2</th>
<th>1</th>
<th>2</th>
<th>1</th>
</tr>
</thead>
<tbody>
<tr>
<td>2</td>
<td>1</td>
<td>4</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>2</td>
<td>1</td>
<td>2</td>
<td></td>
</tr>
</tbody>
</table>
Choose a random permutation $\pi$

Prob. that $h_\pi(C_1) = h_\pi(C_2)$ is the same as $Sim(C_1, C_2)$:

$$Pr[h_\pi(C_1) = h_\pi(C_2)] = Sim(C_1, C_2)$$

Why?

- Let $X$ be a set of shingles, $X \subseteq [2^{64}]$, $x \in X$
- Then: $Pr[\pi(x) = \min(\pi(X))] = 1/|X|$  
  - It is equally likely that any $x \in X$ is mapped to the min element
- Let $x$ be s.t. $\pi(x) = \min(\pi(C_1 \cup C_2))$
- Then either: $\pi(x) = \min(\pi(C_1))$ if $x \in C_1$, or
  $\pi(x) = \min(\pi(C_2))$ if $x \in C_2$
- So the prob. that both are true is the prob. $x \in C_1 \cap C_2$
- $Pr[\min(\pi(C_1))=\min(\pi(C_2))] = |C_1 \cap C_2|/|C_1 \cup C_2| = Sim(C_1, C_2)$
Four Types of Rows

- Given cols $C_1$ and $C_2$, rows may be classified as:

<table>
<thead>
<tr>
<th></th>
<th>$C_1$</th>
<th>$C_2$</th>
</tr>
</thead>
<tbody>
<tr>
<td>a</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>b</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>c</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>d</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>

Also, $a = \#$ rows of type a, etc.

**Note:** $\text{Sim}(C_1, C_2) = \frac{a}{a + b + c}$

**Then:** $\Pr[h(C_1) = h(C_2)] = \text{Sim}(C_1, C_2)$

- Look down the cols $C_1$ and $C_2$ until we see a 1
- If it’s a type-$a$ row, then $h(C_1) = h(C_2)$
- If a type-$b$ or type-$c$ row, then not
The *similarity of two signatures* is the fraction of the hash functions in which they agree.

Note: Because of the minhash property, the similarity of columns is the same as the expected similarity of their signatures.
Min Hashing – Example

Input matrix

<table>
<thead>
<tr>
<th></th>
<th>1</th>
<th>0</th>
<th>1</th>
<th>0</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>

Signature matrix $M$

<table>
<thead>
<tr>
<th></th>
<th>2</th>
<th>1</th>
<th>2</th>
<th>1</th>
</tr>
</thead>
<tbody>
<tr>
<td>2</td>
<td>1</td>
<td>4</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>2</td>
<td>1</td>
<td>2</td>
<td></td>
</tr>
</tbody>
</table>

Similarities:

<table>
<thead>
<tr>
<th></th>
<th>Col/Col</th>
<th>0.75</th>
<th>0.75</th>
<th>0</th>
<th>0</th>
</tr>
</thead>
<tbody>
<tr>
<td>Sig/Sig</td>
<td>0.67</td>
<td>1.00</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>
Minhash Signatures

- Pick (say) 100 random permutations of the rows
- Think of Sig(C) as a column vector
- Let Sig(C)[i] = according to the i-th permutation, the index of the first row that has a 1 in column C
- Note: We store the sketch of document C in ~100 bytes: 
  \[ \text{Sig}(C)[i] = \min(\pi_i(C)) \]
Suppose the matrix has 1 billion rows

Hard to pick a random permutation from 1...billion

Representing a random permutation requires 1 billion entries

Accessing rows in permuted order leads to thrashing
A good approximation to permuting rows: pick 100 (?) hash functions

- $h_1, h_2, \ldots$
- For rows $r$ and $s$, if $h_i(r) < h_i(s)$, then $r$ appears before $s$ in permutation $i$.

For each column $c$ and each hash function $h_i$, keep a “slot” $M(i, c)$

**Intent:** $M(i, c)$ will become the smallest value of $h_i(r)$ for which column $c$ has 1 in row $r$

- i.e., $h_i(r)$ gives order of rows for $i$-th permutation
Example

<table>
<thead>
<tr>
<th>Row</th>
<th>C1</th>
<th>C2</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>2</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>3</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>4</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>5</td>
<td>0</td>
<td>1</td>
</tr>
</tbody>
</table>

\[ h(x) = x \mod 5 \]
\[ h(1)=1, \ h(2)=2, \ h(3)=3, \ h(4)=4, \ h(5)=0 \]
\[ h(C1) = 1 \]
\[ h(C2) = 0 \]

\[ g(x) = 2x+1 \mod 5 \]
\[ g(1)=3, \ g(2)=0, \ g(3)=2, \ g(4)=4, \ g(5)=1 \]
\[ g(C1) = 2 \]
\[ g(C2) = 0 \]

\[ \text{Sig}(C1) = [1,2] \]
\[ \text{Sig}(C2) = [0,0] \]
Sort the input matrix so it is ordered by rows
  So can iterate by reading rows sequentially from disk

\[
\text{for each row } r \\
  \text{for each column } c \\
    \text{if } c \text{ has 1 in row } r \\
    \text{for each hash function } h_i \text{ do} \\
      \text{if } h_i(r) < M(i, c) \text{ then} \\
      \quad M(i, c) := h_i(r)
\]
### Example

<table>
<thead>
<tr>
<th>Row</th>
<th>C1</th>
<th>C2</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>2</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>3</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>4</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>5</td>
<td>0</td>
<td>1</td>
</tr>
</tbody>
</table>

$h(x) = x \text{ mod } 5$

$g(x) = 2x + 1 \text{ mod } 5$

<table>
<thead>
<tr>
<th>$h(x)$</th>
<th>$g(x)$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$h(1) = 1$</td>
<td>$g(1) = 3$</td>
</tr>
<tr>
<td>$h(2) = 2$</td>
<td>$g(2) = 0$</td>
</tr>
<tr>
<td>$h(3) = 3$</td>
<td>$g(3) = 2$</td>
</tr>
<tr>
<td>$h(4) = 4$</td>
<td>$g(4) = 4$</td>
</tr>
<tr>
<td>$h(5) = 0$</td>
<td>$g(5) = 1$</td>
</tr>
</tbody>
</table>

$M(i, c)$
The set of strings of length $k$ that appear in the document

Signatures: short integer vectors that represent the sets, and reflect their similarity

Candidate pairs: those pairs of signatures that we need to test for similarity.

Locality Sensitive Hashing
Goal: Pick a similarity threshold $s$, e.g., $s = 0.8$
Find documents with Jaccard similarity at least $s$

**LSH – General idea:** Use a function $f(x, y)$ that tells whether or not $x$ and $y$ is a *candidate pair*: a pair of elements whose similarity must be evaluated

- For minhash matrices: Hash columns to many buckets, and make elements of the same bucket candidate pairs
- Each pair of documents that hashes into the same bucket is a *candidate pair*
Candidates from Minhash signatures

- Pick a similarity threshold $s$, a fraction $< 1$

- Columns $x$ and $y$ are a candidate pair if their signatures agree on at least fraction $s$ of their rows:
  \[ M(i, x) = M(i, y) \] for at least frac. $s$ values of $i$

- We expect documents $x$ and $y$ 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

Matrix $M$

$b$ bands

$r$ rows per band

One signature
Divide matrix $M$ into $b$ bands of $r$ rows.

For each band, hash its portion of each column to a hash table with $k$ buckets.

- Make $k$ as large as possible.

*Candidate* column pairs are those that hash to the same bucket for $\geq 1$ band.

Tune $b$ and $r$ to catch most similar pairs, but few nonsimilar pairs.
Matrix $M$ has $r$ rows and $b$ bands.

Columns 2 and 6 are probably identical (candidate pair).

Columns 6 and 7 are surely different.
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

- Suppose 100,000 columns
- Signatures of 100 integers.
- Therefore, signatures take 40Mb.
- Choose 20 bands of 5 integers/band.

- Goal: find pairs of documents that are at least 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: $(1-0.328)^{20} = 0.00035$.

- 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 $C_1$, $C_2$ are only 30% similar

- Probability $C_1$, $C_2$ identical in any one particular band: $(0.3)^5 = 0.00243$
- Probability $C_1$, $C_2$ identical in $\geq 1$ of 20 bands: $\leq 20 \times 0.00243 = 0.0486$
- 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

Probability of sharing a bucket

No chance if $s < t$

Probability
$= 1$ if $s > t$

Similarity $s$ of two sets
Probability of sharing a bucket

Remember: probability of equal hash-values = similarity

Similarity $s$ of two sets

$t$
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

All rows of a band are equal

At least one band identical

No bands identical

Some row of a band unequal

All rows of a band are equal

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

$t \sim (1/b)^{1/r}$

Probability of sharing a bucket

Similarity $s$ of two sets
Example: $b = 20; r = 5$

<table>
<thead>
<tr>
<th>s</th>
<th>$1-(1-s^r)^b$</th>
</tr>
</thead>
<tbody>
<tr>
<td>.2</td>
<td>.006</td>
</tr>
<tr>
<td>.3</td>
<td>.047</td>
</tr>
<tr>
<td>.4</td>
<td>.186</td>
</tr>
<tr>
<td>.5</td>
<td>.470</td>
</tr>
<tr>
<td>.6</td>
<td>.802</td>
</tr>
<tr>
<td>.7</td>
<td>.975</td>
</tr>
<tr>
<td>.8</td>
<td>.9996</td>
</tr>
</tbody>
</table>
**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 set of strings of length $k$ that appear in the document

Signatures: short integer 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
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!
For min-hash signatures, we got a min-hash function for each permutation of rows

An example of a family of hash functions

- A “hash function” is any function that takes two elements and says whether or not they are “equal” (really, are candidates for similarity checking).
  - **Shorthand**: \( h(x) = h(y) \) means “\( h \) says \( x \) and \( y \) are equal.”

- A **family** of hash functions is any set of hash functions
  - A set of related hash functions generated by some mechanism

- We should be able to efficiently pick a hash function at random from such a family
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 $d(x, y) > d_2$, 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

High probability; at least \(p_1\)

Low probability; at most \(p_2\)

\[ h(x) = h(y) \]

\[ d(x, y) \]
Let $S = \text{sets}$, $d = \text{Jaccard distance}$, $\mathbf{H}$ is family of 
minhash functions for all permutations of 
rows

Then for any hash function $h$ in $\mathbf{H}$,

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

Simply restates theorem about min-hashing in 
terms of distances rather than similarities
Claim: $H$ 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 minhash values agree is $\geq 2/3$

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

Theory leaves unknown what happens to pairs that are at distance between $d_1$ and $d_2$

Consequence: no guarantees about fraction of false positives in that range
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 $\mathcal{H}$, construct family $\mathcal{H}'$ consisting of $r$ functions from $\mathcal{H}$.
- For $h = [h_1, \ldots, h_r]$ in $\mathcal{H}'$, $h(x) = h(y)$ if and only if $h_i(x) = h_i(y)$ for all $i$.

**Theorem:** If $\mathcal{H}$ is $(d_1, d_2, p_1, p_2)$-sensitive, then $\mathcal{H}'$ is $(d_1, d_2, (p_1)^r, (p_2)^r)$-sensitive.

**Proof:** Use fact that $h_i$’s are independent.
Given family $\mathbf{H}$, construct family $\mathbf{H}'$ consisting of $b$ functions from $\mathbf{H}$.

For $h = [h_1, \ldots, h_b]$ in $\mathbf{H}'$, $h(x) = h(y)$ if and only if $h_i(x) = h_i(y)$ for some $i$.

**Theorem:** If $\mathbf{H}$ is $(d_1, d_2, p_1, p_2)$-sensitive, then $\mathbf{H}'$ 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

- *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
  - Construction makes (x,y) a candidate pair with probability \( 1 - (1-p^r)^b \)
    - 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 $1-(1-p^4)^4$**

<table>
<thead>
<tr>
<th>$p$</th>
<th>$1-(1-p^4)^4$</th>
</tr>
</thead>
<tbody>
<tr>
<td>.2</td>
<td>.0064</td>
</tr>
<tr>
<td>.3</td>
<td>.0320</td>
</tr>
<tr>
<td>.4</td>
<td>.0985</td>
</tr>
<tr>
<td>.5</td>
<td>.2275</td>
</tr>
<tr>
<td>.6</td>
<td>.4260</td>
</tr>
<tr>
<td>.7</td>
<td>.6666</td>
</tr>
<tr>
<td>.8</td>
<td>.8785</td>
</tr>
<tr>
<td>.9</td>
<td>.9860</td>
</tr>
</tbody>
</table>

**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
- Transforms 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 \((1-(1-p)^4)^4\)

<table>
<thead>
<tr>
<th>p</th>
<th>((1-(1-p)^4)^4)</th>
</tr>
</thead>
<tbody>
<tr>
<td>.1</td>
<td>.0140</td>
</tr>
<tr>
<td>.2</td>
<td>.1215</td>
</tr>
<tr>
<td>.3</td>
<td>.3334</td>
</tr>
<tr>
<td>.4</td>
<td>.5740</td>
</tr>
<tr>
<td>.5</td>
<td>.7725</td>
</tr>
<tr>
<td>.6</td>
<td>.9015</td>
</tr>
<tr>
<td>.7</td>
<td>.9680</td>
</tr>
<tr>
<td>.8</td>
<td>.9936</td>
</tr>
</tbody>
</table>

**Example:** Transforms a \((.2,.8,.8,.2)\)-sensitive family into a \((.2,.8,.9936,.1215)\)-sensitive family.
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.
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.
For cosine distance, there is a technique called Random Hyperplanes.

- Technique similar to minhashing.
- A \( \left( d_1, d_2, \left(1 - d_1 / 180\right), \left(1 - d_2 / 180\right) \right) \)-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 $\mathcal{H} = \text{set of all functions derived from any vector}$.
- Claim: For points $x$ and $y$,
  \[
  \Pr[h(x) = h(y)] = 1 - \frac{d(x,y)}{180}
  \]
Proof of Claim

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

Hyperplane normal to $\nu$

$h(x) = h(y)$

Hyperplane normal to $\nu$

$h(x) \neq h(y)$

$\text{Prob[Red case]} = \frac{\theta}{180}$
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?
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.
If $d \ll a$, then the chance the points are in the same bucket is at least $1 - \frac{d}{a}$. 

Points at distance $d$

Bucket width $a$
Projection of Points

If $d >> a$, $\theta$ must be close to $90^\circ$ for there to be any chance points go to the same bucket.
An LS-Family for Euclidean Distance

- If points are distance $d \leq \frac{a}{2}$, prob. they are in same bucket $\geq 1 - \frac{d}{a} = \frac{1}{2}$
- If points are distance $\geq 2a$ apart, then they can be in the same bucket only if $d \cos \theta \leq a$
  - $\cos \theta \leq \frac{1}{2}$
  - $60 \leq \theta < 90$
  - i.e., at most $1/3$ probability.
- Yields a $(\frac{a}{2}, 2a, 1/2, 1/3)$-sensitive family of hash functions for any $a$.
- Amplify using AND-OR cascades
Fixup: Euclidean Distance

- For previous distance measures, we could start with an \((x, y, p, q)\)-sensitive family for any \(x < y\), and drive \(p\) and \(q\) to 1 and 0 by AND/OR constructions.
- Here, we seem to need \(y \geq 4x\).
But as long as $x < y$, the probability of points at distance $x$ falling in the same bucket is greater than the probability of points at distance $y$ doing so.

Thus, the hash family formed by projecting onto lines is an $(x, y, p, q)$-sensitive family for some $p > q$.

Then, amplify by AND/OR constructions.