Locality-Sensitive Hashing for Documents

ANISHA JOSEPH

Locality-Sensitive Hashing

- A fundamental **data-mining problem** is to examine data for “similar” items.
- The **problem of similarity** as one of finding sets with a relatively large intersection.
- **shingling and minhashing** are the methods can be the used to solve similarity problem.
- Another important problem is when we search for similar items of any kind, there may be too many pairs of items to test for their degree of similarity, even if computing the similarity of any one pair can be made very easy.

Locality-Sensitive Hashing

- A technique called "**locality-sensitive hashing**," for focusing search on pairs that are most likely to be similar.
- For "similarity" that are not expressible as intersection of sets, consider the theory of **distance measures** in arbitrary spaces.
- It also motivates a general framework for locality-sensitive hashing that applies for other definitions of "similarity."
- Aspect of **similarity** we are looking at here is **character-level similarity,** not "similar meaning," which requires us to examine the words in the documents and their uses.

Locality-Sensitive Hashing

- Textual similarity also has important uses. Many of these involve finding duplicates or near duplicates.
- Testing whether two documents are exact duplicates is easy; just compare the two documents character-by-character, and if they ever differ then they are not the same.
- Many applications are Plagiarism, Mirror Pages, Articles from the Same Source etc.

Shingling of Documents

- The most effective way to represent documents as set for the purpose of identifying lexically similar documents.
- Construct f the set of short strings that appear within from the document.
- If we do so, then documents that share pieces as short as sentences or even phrases will have many common elements in their sets, even if those sentences appear in different orders in the two documents.

k-Shingles

- A document is a string of characters.
- Define a k-shingle for a document to be any substring of length k found within the document.
- Each document the set of k-shingles that appear one or more times within that document.

Eg:

Suppose our document D is the string abcdabd, and we pick k = 2. Then the set of 2-shingles for D is {ab, bc, cd, da, bd}.

k-Shingles

- Note that the substring ab appears twice within D, but appears only once as a shingle.
- A variation of shingling produces a bag, rather than a set, so each shingle would appear in the result as many times as it appears in the document.
- There are several options regarding how white space (blank, tab, newline, etc.) is treated.
- It probably makes sense to replace any sequence of one or more white-space characters by a single blank.

k-Shingles

Eg: If we use **k = 9,** but eliminate whitespace altogether, then we would see some lexical similarity in the sentences **“The plane was ready for touch down”**. and **“The quarterback scored a touchdown”**.

However, if we retain the blanks, then the first has shingles **touch dow** and **ouch down.**

While the second has **touchdown**. If we eliminated the blanks, then both would have touchdown.

Choosing the Shingle Size

- We can pick k to be any constant we like.
- However, if we pick k too small, then we would expect most sequences of k characters to appear in most documents.
- How large k should be depends on how long typical documents are and how large the set of typical characters is.
- k should be picked large enough that the probability of any given shingle appearing in any given document is low.
- Thus, if our corpus of documents is emails, picking k = 5 should be fine.

Choosing the Shingle Size

- If so, then there would be 27^5 = 14,348,907 possible shingles.
- Since the typical email is much smaller than 14 million characters long, we would expect k = 5 to work well, and indeed it does.
- However, all characters do not appear with equal probability. Common letters and blanks dominate, while ”z” and other letters that have high point-value in Scrabble are rare.
- A good rule of thumb is to imagine that there are only 20 characters and estimate the number of k-shingles as 20^k.

Choosing the Shingle Size

- Thus, even short emails will have many 5-shingles consisting of common letters, and the chances of unrelated emails sharing these common shingles is greater.
- For large documents, such as research articles, choice k = 9 is considered safe.

Hashing Shingles

- Instead of using substrings directly as shingles, we can pick a hash function that maps strings of length k to some number of buckets and treat the resulting bucket number as the shingle.
- we could construct the set of 9-shingles for document and then map each of those 9-shingles to a bucket number in the range 0 to 23^2 − 1.
- Thus, each shingle is represented by four bytes instead of nine.
- Not only has the data been compacted, but we can now manipulate (hashed) shingles by single-word machine operations.

Hashing Shingles

- We can differentiate documents better if we use 9-shingles and hash them down to four bytes than to use 4-shingles, even though the space used to represent a shingle is the same.
- (Reason common letters will be more for 4 shingles instead of 9).
- if we use 9-shingles, there are many more than 232 likely shingles. When we hash them down to four bytes, we can expect almost any sequence of four bytes to be possible.
- (9- shingles -20^9 possible , convert to 4 bytes of 2^32 possible ways)

Shingles Built from Words

An alternative form of shingle has proved effective for the problem of identifying similar news articles.

The news articles are written in a rather different style than are other elements that typically appear on the page with the article.

News articles have a lot of stop words. The most common words such as “and,” “you,” “to,” and so on.

In many applications, we want to ignore stop words, since they don’t tell us anything useful about the article, such as its topic.

Shingles Built from Words

- For the problem of **finding similar news articles**, it was found that defining a shingle to be a **stop word followed by the next two words,** regardless of whether or not they were stop words, **formed a useful set of shingles.**
- Advantage of this approach is that the news article would then contribute more shingles to the set representing the Web page than would the surrounding elements.
- The goal of the exercise is to find pages that had the same articles, regardless of the surrounding elements.

Example

Simple text: **“Buy Sudzo.”**

News Article: “**A spokesperson** *for the* **Sudzo Corporation revealed today** *that* **studies** *have* **shown** *it is* **good** *for* **people** *to* **buy Sudzo products.”**

Italics words are stop words.

Most frequent words that should be considered stop words.

Example

The first three shingles made from a stop word and the next two following are:

A spokesperson for

for the Sudzo

the Sudzo Corporation

There are nine shingles from the sentence. Find it.

Exercise.

1. What are the first ten 3-shinglesm in the below sentence?

The most effective way to represent documents as sets, for the purpose of identifying lexically similar documents is to construct from the document the set of short strings that appear within it.

2. If we use the stop-word-based shingles and we take the stop words to be all the words of three or fewer letters, then what are the shingles?

Similarity-Preserving Summaries of Sets

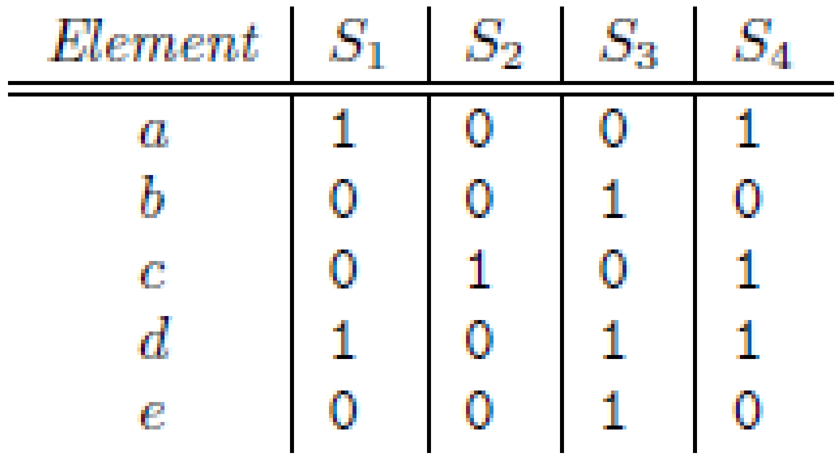
- Sets of shingles are large.
- Even if we hash them to four bytes each, the space needed to store a set is still roughly four times the space taken by the document.
- Our goal in this section is to replace large sets by much smaller representations called “signatures.”
- Compare signatures to find similarity.
- It may not give accurate result.
- If signatures is large.

Matrix Representation of Sets

- Before explaining how it is possible to construct small signatures from large sets, it is helpful to visualize a collection of sets as their characteristic matrix.
- Columns of the matrix correspond to the sets, and the rows correspond to elements of the universal set from which elements of the sets are drawn.
- There is a 1 in row r and column c if the element for row r is a member of the set for column c. Otherwise the value in position (r, c) is 0.

Matrix Representation of Sets

Matrix representing sets chosen from the universal set {a, b, c, d, e}. Here, S1 = {a, d}, S2 = {c}, S3 = {b, d, e}, and S4 = {a, c, d}.



Element	S_1	S_2	S_3	S_4
a	1	0	0	1
b	0	0	1	0
c	0	1	0	1
d	1	0	1	1
e	0	0	1	0

Figure 3.2: A matrix representing four sets

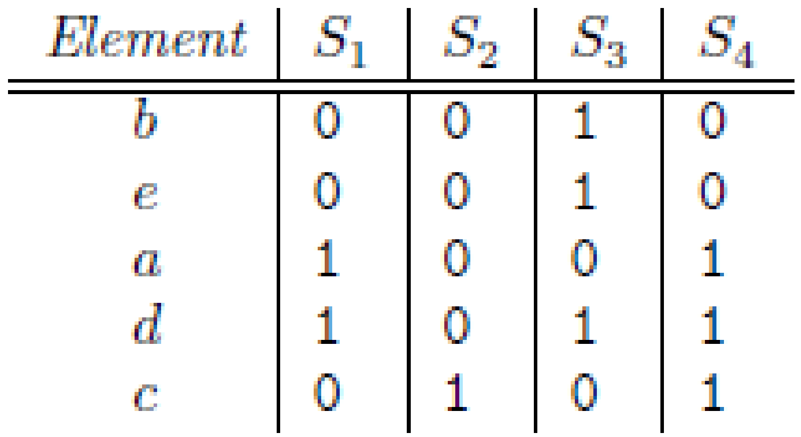
Matrix Representation of Sets

The top row and leftmost columns are not part of the matrix, but are present only to remind us what the rows and columns represent.

Minhashing

- To minhash a set represented by a column of the characteristic matrix, pick a permutation of the rows.
- The minhash value of any column is the number of the first row, in the permuted order, in which the column has a 1.
- Eg: Let us suppose we pick the order of rows beadc for the matrix.
- This permutation defines a minhash function h that maps sets to rows.
- Let us compute the minhash value of set S1 according to h.
- The first column, which is the column for set S1, has 0 in row b, so we proceed to row e, the second in the permuted order.

Minhashing



Element	S_1	S_2	S_3	S_4
b	0	0	1	0
e	0	0	1	0
a	1	0	0	1
d	1	0	1	1
c	0	1	0	1

Figure 3.3: A permutation of the rows of Fig. 3.2

In this matrix, we can read off the values of h by scanning from the top until we come to a 1. Thus, we see that h(S2) = c, h(S3) = b, and h(S4) = a.

Jacard Similarity

To the columns for sets S1 and S2, then rows can be divided into three classes,

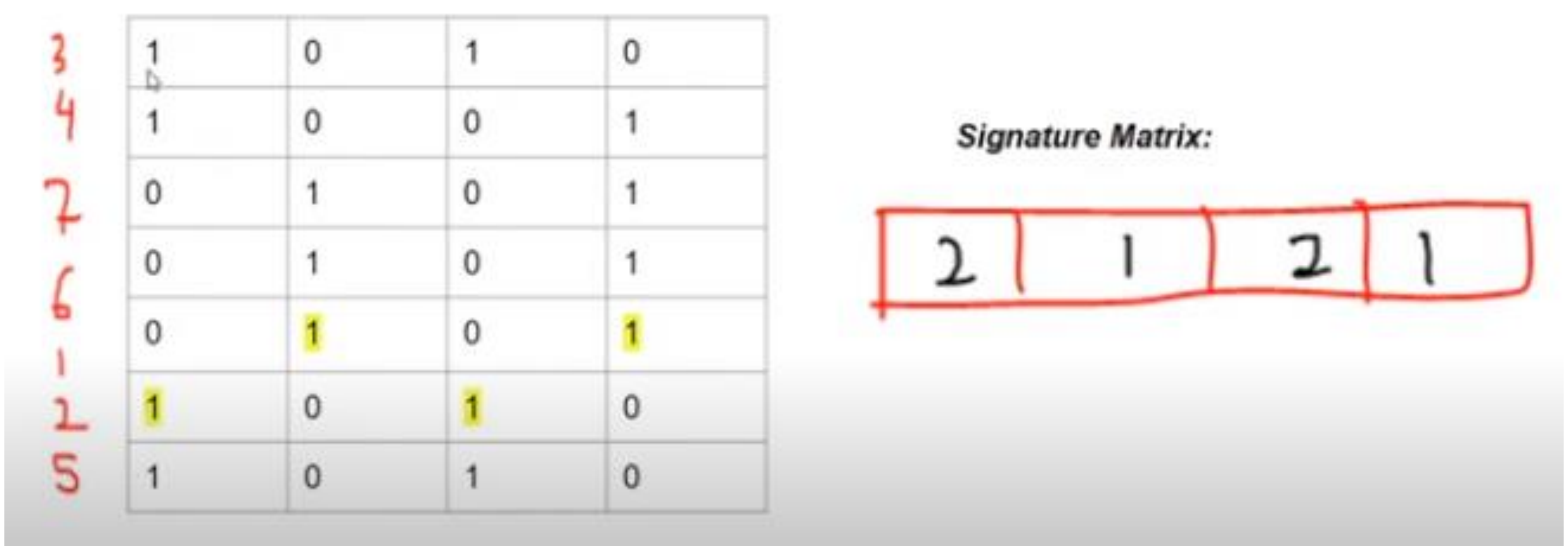
1. Type X rows have 1 in both columns.
2. Type Y rows have 1 in one of the columns and 0 in the other.
3. Type Z rows have 0 in both columns.

Then $SIM(S1, S2) = x/(x + y)$.

Minhash Signatures

- To represent sets, we pick at random some number n of permutations of the rows of M.
- Call the minhash functions determined by these permutations h1, h2, . . . , hn.
- From the column representing set S, construct the minhash signature for S, the vector [h1(S), h2(S), . . . , hn(S)].
- We normally represent this list of hash-values as a column.
- we can form from matrix M a **signature matrix**, in which the ith column of M is replaced by the minhash signature for (the set of) the ith column.

Example – Signature Matrix



3	1	0	1	0
4	1	0	0	1
7	0	1	0	1
6	0	1	0	1
1	0	1	0	1
2	1	0	1	0
5	1	0	1	0

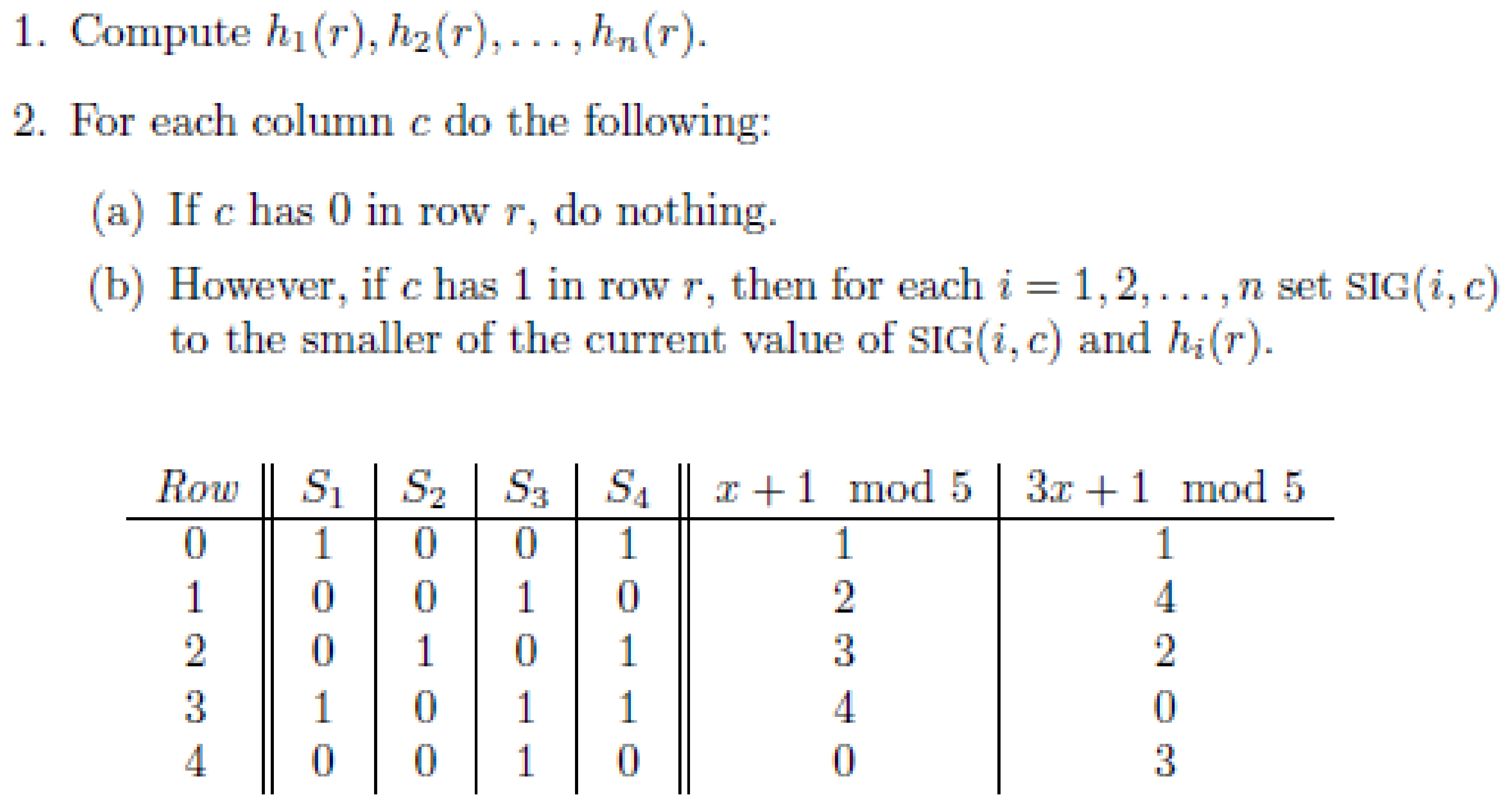
Signature Matrix:

2	1	2	1

Computing Minhash Signatures

- It is not feasible to permute a large characteristic matrix explicitly.
- Even picking a random permutation of millions or billions of rows is time-consuming, and the necessary sorting of the rows would take even more time.
- it is possible to simulate the effect of a random permutation by a random hash function that maps row numbers to as many buckets as there are rows.

Computing Minhash Signatures



1. Compute $h_1(r), h_2(r), \ldots, h_n(r)$.
2. For each column c do the following:
   (a) If c has 0 in row r, do nothing.
   (b) However, if c has 1 in row r, then for each $i = 1, 2, \ldots, n$ set $\text{SIG}(i, c)$ to the smaller of the current value of $\text{SIG}(i, c)$ and $h_i(r)$.

Row	S_1	S_2	S_3	S_4	$x + 1 \mod 5$	$3x + 1 \mod 5$
0	1	0	0	1	1	1
1	0	0	1	0	2	4
2	0	1	0	1	3	2
3	1	0	1	1	4	0
4	0	0	1	0	0	3

Figure 3.4: Hash functions computed for the matrix of Fig. 3.2

Computing Minhash Signatures

Initially, this matrix consists of all ∞'s:

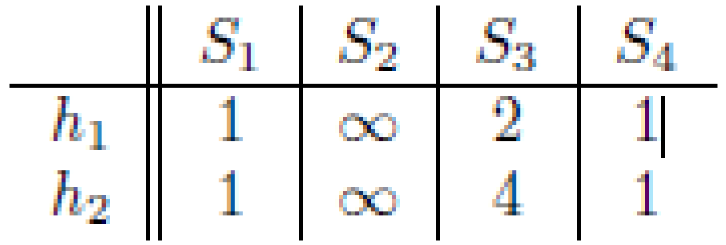
	S_1	S_2	S_3	S_4
h_1	∞	∞	∞	∞
h_2	∞	∞	∞	∞

First, we consider row 0,

	S_1	S_2	S_3	S_4
h_1	1	∞	∞	1
h_2	1	∞	∞	1

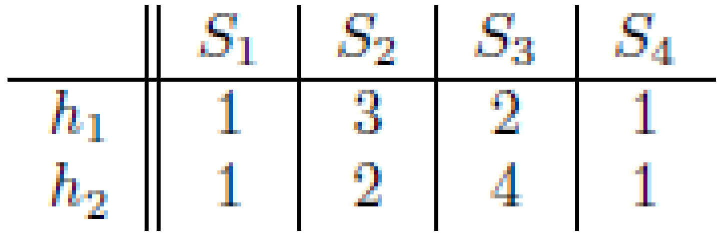
Computing Minhash Signatures

Now, we move to the row numbered 1,



	S_1	S_2	S_3	S_4
h_1	1	∞	2	1
h_2	1	∞	4	1

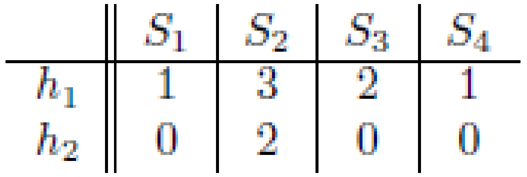
row numbered 2,



	S_1	S_2	S_3	S_4
h_1	1	3	2	1
h_2	1	2	4	1

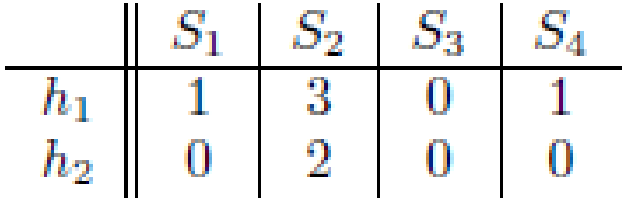
Computing Minhash Signatures

row numbered 3,



	S_1	S_2	S_3	S_4
h_1	1	3	2	1
h_2	0	2	0	0

row numbered 4,



	S_1	S_2	S_3	S_4
h_1	1	3	0	1
h_2	0	2	0	0

Computing Minhash Signatures

- We can estimate the Jaccard similarities of the underlying sets from this signature matrix.
- Notice that columns 1 and 4 are identical, so we guess that SIM(S1, S4) = 1.0.
- we see that the true Jaccard similarity of S1 and S4 is 2/3.

  (2 rows with 1's, and 3 rows with 0 and 1)
- Remember that the fraction of rows that agree in the signature matrix is only an estimate of the true Jaccard similarity.

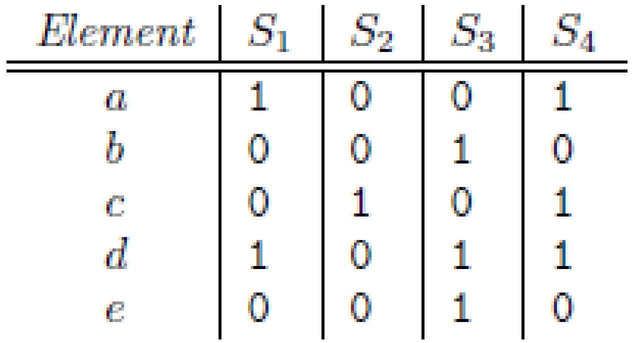
Computing Minhash Signatures

For additional examples, the signature columns for S1 and S3 agree in half the rows (true similarity 1/4).

while the signatures of S1 and S2 estimate 0 as their Jaccard similarity (the correct value).

Excercise

1. Compute the Jaccard similarity of each of the pairs of columns.



Element	S_1	S_2	S_3	S_4
a	1	0	0	1
b	0	0	1	0
c	0	1	0	1
d	1	0	1	1
e	0	0	1	0

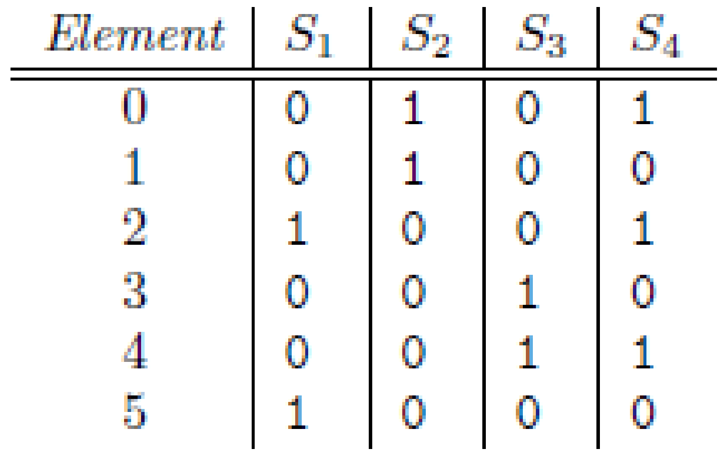
2. Compute, for each pair of columns of that figure, the fraction of the 120 permutations of the rows that make the two columns hash to the same value.

Excercise

Using the data from Fig. 3.4, add to the signatures of the columns the values of the following hash functions.

(a) h3(x) = 2x + 4 mod 5.

(b) h4(x) = 3x − 1 mod 5.



Element	S_1	S_2	S_3	S_4
0	0	1	0	1
1	0	1	0	0
2	1	0	0	1
3	0	0	1	0
4	0	0	1	1
5	1	0	0	0

Figure 3.5: Matrix for Exercise 3.3.3

Excercise

For the above data,

a) Compute the minhash signature for each column if we use the following three hash functions: h1(x) = 2x + 1 mod 6; h2(x) = 3x + 2 mod 6; h3(x) = 5x + 2 mod 6.

b) Which of these hash functions are true permutations?

c) How close are the estimated Jaccard similarities for the six pairs of columns to the true Jaccard similarities?

Distance Measures

Suppose we have a set of points, called a space.

A distance measure on this space is a function $d(x, y)$ that takes two points in the space as arguments and produces a real number, and satisfies the following axioms:

1. $d(x, y) \geq 0$ (no negative distances).

2. $d(x, y) = 0$ if and only if $x = y$ (distances are positive, except for the distance from a point to itself).

3. $d(x, y) = d(y, x)$ (distance is symmetric).

4. $d(x, y) \leq d(x, z) + d(z, y)$ (the triangle inequality).

1. Euclidean Distances

The most familiar distance measure is the one we normally think of as “distance.”

An ***n-dimensional Euclidean space*** is one where points are vectors of n real numbers.

The conventional distance measure in this space, which we shall refer to as the ***L2-norm***, is defined:

$$d([x_1, x_2, \ldots, x_n],\ [y_1, y_2, \ldots, y_n]) = \sqrt{\sum_{i=1}^{n} (x_i - y_i)^2}$$

1. Euclidean Distances

For any constant r, we can define the ***Lr-norm*** to be the distance measure ***d*** defined by,

$$d([x_1, x_2, \ldots, x_n],\ [y_1, y_2, \ldots, y_n]) = (\sum_{i=1}^{n} |x_i - y_i|^r)^{1/r}$$

The case r = 2 is the usual ***L2-norm*** just mentioned. Another common distance measure is the ***L1-norm, or Manhattan distance.***

L∞-norm, which is the limit as r approaches infinity of the Lr-norm. As r gets larger, only the dimension with the largest difference matters, so formally, the L∞-norm is defined as the maximum of **|xi – yi|** over all dimensions i.

Example:

Consider the two-dimensional Euclidean space (the customary plane) and the points (2, 7) and (6, 4). Find L2-norm, L1- norm and L∞-norm.

Ans: The L2-norm gives a distance of((2 − 6)^2 + (7 − 4)^2)^1/2= (4^2 + 3^2)^2= 5.

The L1-norm gives a distance of |2 − 6| + |7 − 4| = 4 + 3 = 7.

The L∞-norm gives a distance of max(|2 − 6|, |7 − 4|) = max(4, 3) = 4

Jaccard Distance

Jaccard distance of sets by d(x, y) = 1- Jaccard Similarity= 1 − SIM(x, y).

That is, the Jaccard distance is 1 minus the ratio of the sizes of the intersection and union of sets x and y.

We must verify that this function is a distance measure.

1. d(x, y) is nonnegative because the size of the intersection cannot exceed the size of the union.
2. d(x, y) = 0 if x = y, because x ∪ x = x ∩ x = x. However, if x 6= y, then the size of x ∩ y is strictly less than the size of x ∪ y, so d(x, y) is strictly positive.

Jaccard Distance

3. d(x, y) = d(y, x) because both union and intersection are symmetric; i.e., $x \cup y = y \cup x$ and $x \cap y = y \cap x$.

4. triangle inequality: SIM(x, y) is the probability a random minhash function maps x and y to the same value.

Thus, the Jaccard distance d(x, y) is the probability that a random minhash function does not send x and y to the same value

Jaccard Distance

We can therefore translate the condition $d(x, y) \leq d(x, z) + d(z, y)$ to the statement that if h is a random minhash function.

I.e the probability that $h(x) \neq h(y)$ is no greater than the sum of the probability that $h(x) \neq h(z)$ and the probability that $h(z) \neq h(y)$.

However, this statement is true because whenever $h(x) \neq h(y)$, at least one of $h(x)$ and $h(y)$ must be different from $h(z)$. They could not both be $h(z)$, because then $h(x)$ and $h(y)$ would be the same.

Cosine Distance

The cosine distance makes sense in spaces that have dimensions, including Euclidean spaces and discrete versions of Euclidean spaces, such as spaces where points are vectors with integer components or Boolean (0 or 1) components.

Then the cosine distance between two points is the angle that the vectors to those points make.

This angle will be in the range 0 to 180 degrees, regardless of how many dimensions the space has.

Cosine Distance

Given two vectors x and y, the cosine of the angle between them is **the dot product x.y / the L2-norms of x and y** (i.e., their Euclidean distances from the origin).

Recall that the dot product of vectors.

$$[x_1, x_2, \ldots, x_n].[y_1, y_2, \ldots, y_n] \text{ is } \sum_{i=1}^{n} x_i y_i$$

Cosine Distance

Let our two vectors be x = [1, 2,−1] and = [2, 1, 1]. The dot product x.y is 1 × 2 + 2 × 1 + (−1) × 1 = 3.

The L2-norm of both vectors is √6.

For example, x has L2-norm(1^2 + 2^2 + (−1)^2)1/2 = √6.

Thus, the cosine of the angle between x and y is 3/(√6√6) or 1/2.

The angle whose cosine is ½ is 60 degrees, so that is the cosine distance between x and y.

Exercise 3.5.5: Compute the cosines of the angles between each of the following pairs of vectors.[5]

(a) $(3, -1, 2)$ and $(-2, 3, 1)$.

(b) $(1, 2, 3)$ and $(2, 4, 6)$.

(c) $(5, 0, -4)$ and $(-1, -6, 2)$.

(d) $(0, 1, 1, 0, 1, 1)$ and $(0, 0, 1, 0, 0, 0)$.

Edit Distance

This distance makes sense when points are strings.

The distance between two strings x = x1x2 · · · xn and y = y1y2 · · · ym is the smallest number of insertions and deletions of single characters that will convert x to y.

Eg: The edit distance between the strings **x = abcde** and y **=**

acfdeg is 3. To convert x to y:

1. Delete b.

2. Insert f after c.

3. Insert g after e.

Edit Distance

No sequence of fewer than three insertions and/or deletions will convert x to y. Thus, d(x, y) = 3.

Another way to define and calculate the edit distance d(x, y) is to compute a **longest common subsequence (LCS)** of x and y.

An **LCS of x and y is a string that is constructed by deleting positions from x and y**, and that is as long as any string that can be constructed that way.

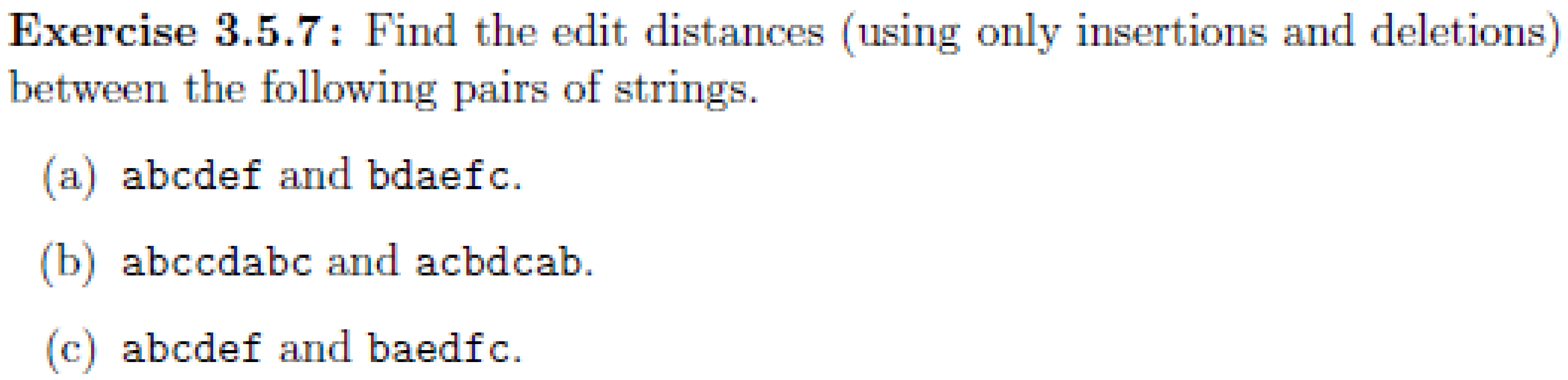
The edit distance d(x, y) can be calculated as the length of x plus the length of y minus twice the length of their LCS.

Edit Distance

- Example: The strings **x = abcde** and **y = acfdeg** have a unique LCS, which is **acde.**
- We can be sure it is the longest possible, because it contains every symbol appearing in both x and y.
- Fortunately, these common symbols appear in the same order in both strings, so we are able to use them all in an LCS.
- Note that the length of x is 5, the length of y is 6, and the length of their LCS is 4. The edit distance is thus **5 + 6 – 2 × 4 = 3**, which agrees with the direct calculation.

Edit Distance

Home work: Find edit Distance for x = aba and y = bab.



Exercise 3.5.7: Find the edit distances (using only insertions and deletions) between the following pairs of strings.

(a) `abcdef` and `bdaefc`.

(b) `abccdabc` and `acbdcab`.

(c) `abcdef` and `baedfc`.

Hamming Distance

Given a space of vectors, we define the Hamming distance between two vectors to be the number of components in which they differ.

The Hamming distance between the vectors **10101 and 11110 is 3**. That is, these vectors differ in the second, fourth, and fifth components, while they agree in the first and third components.

Hamming Distance

Find the Hamming distances between each pair of the following vectors: 000000, 110011, 010101, and 011100.

Locality-Sensitive Functions

- The LSH technique to distinguish strongly between pairs at a low distance from pairs at a high distance.
- These functions can apply to different distance measure.
- There are three conditions that we need for a family of functions:
- In many cases, the function **f will “hash” items**, and the decision will be based on whether or not the result is equal.
- Because it is convenient to use the notation **f(x) = f(y)** to mean that f(x, y) is “yes; make x and y a **candidate pair,”.**

Locality-Sensitive Functions

- We also use **f(x) ≠ f(y)** to mean “do not make x and y a candidate pair unless some other function concludes we should do so.”
- A collection of functions of this form will be called a f**amily of functions.**
- Let d1 < d2 be two distances according to some distance measure d. A

  family **F** of functions is said to be (d1, d2, p1, p2)-sensitive if for every f in F:
  - If d(x, y) ≤ d1, then the probability that f(x) = f(y) is at least p1.
  - If d(x, y) ≥ d2, then the probability that f(x) = f(y) is at most p2.

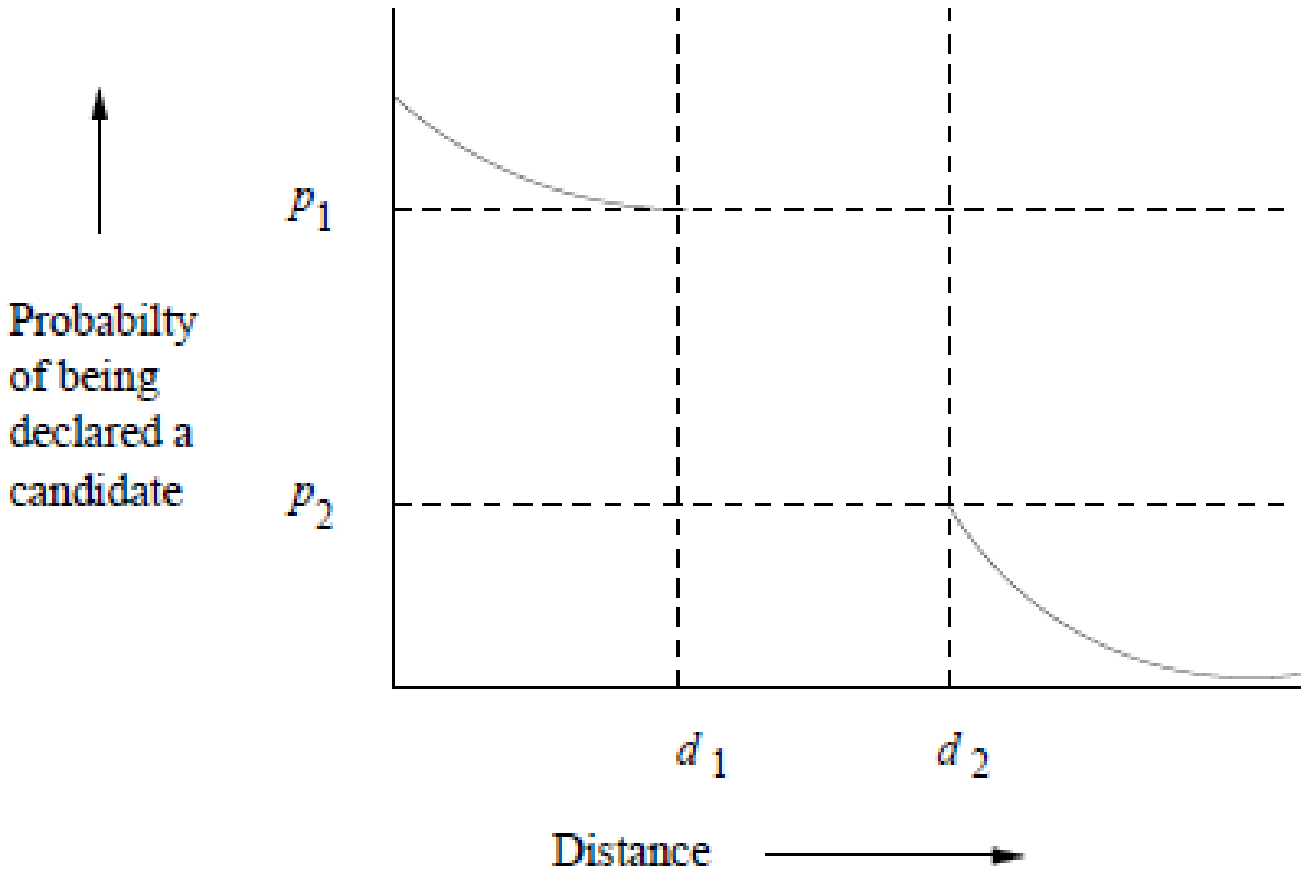


Figure 3.9: Behavior of a (d_1, d_2, p_1, p_2)-sensitive function

Locality-Sensitive Families for Jaccard Distance

- we have only one way to find a family of locality-sensitive functions: use the family of minhash functions, and assume that the distance measure is the Jaccard distance.
- As before, we interpret a minhash function h to make x and y a candidate pair if and only if h(x) = h(y).
- **The family of minhash functions is a (d1, d2, 1−d1, 1−d2)-sensitive family for any d1 and d2, where $0 \leq d1 < d2 \leq 1$.**
- The reason is that if $d(x, y) \leq d1$, where d is the Jaccard distance, then $SIM(x, y) = 1 - d(x, y) \geq 1 - d1$.

Locality-Sensitive Families for Jaccard Distance

(I.e when distance between x and y decreases probability of h(x)=h(y) increases.

when distance between x and y increases probability of h(x)=h(y) decreases.

I.e p1=1-d1.)

Example:

Let d1 = 0.3 and d2 = 0.6.

Then we can assert that the family of minhash functions is a (0.3, 0.6, 0.7, 0.4)-sensitive family.

That is, if the Jaccard distance between x and y is at most 0.3 (i.e., SIM(x, y) ≥ 0.7) then there is at least a 0.7 chance that a minhash function will send x and y to the same value.

And if the Jaccard distance between x and y is at least 0.6 (i.e., SIM(x, y) ≤ 0.4), then there is at most a 0.4 chance that x and y will be sent to the same value.

Note that we could make the same assertion with another choice of d1 and d2; only d1 < d2 is required.

LSH Families for Hamming Distance

- Suppose we have a space of d-dimensional vectors, and h(x, y) denotes the Hamming distance between vectors x and y.
- If we take any one position of the vectors, say the ith position, we can define the function fi(x) to be the ith bit of vector x.
- Then fi(x) = fi(y) if and only if vectors x and y agree in the ith position.
- Then the probability that fi(x) = fi(y) for a randomly chosen i is exactly **1 – h(x, y)/d;**
- i.e., it is the fraction of positions in which x and y agree.

LSH Families for Hamming Distance

Thus, the family **F** consisting of the functions {f1, f2, . . . , fd} is a

(d1, d2, 1 − d1/d, 1 − d2/d)-sensitive

family of hash functions, for any d1 < d2.

1. While Jaccard distance runs from 0 to 1, the Hamming distance on a vector space of dimension d runs from 0 to d.

It is therefore necessary to scale the distances by dividing by d, to turn them into probabilities.

2. While there is essentially an unlimited supply of minhash functions, the

size of the family F for Hamming distance is only d.

Random Hyperplanes and the Cosine Distance

Two vectors x and y that make an angle θ between them.

Note that these vectors may be in a space of many dimensions, but they always define a plane, and the angle between them is measured in this plane.

First, consider a vector v that is normal to the hyperplane whose projection is represented by the dashed line.

Random Hyperplanes and the Cosine Distance

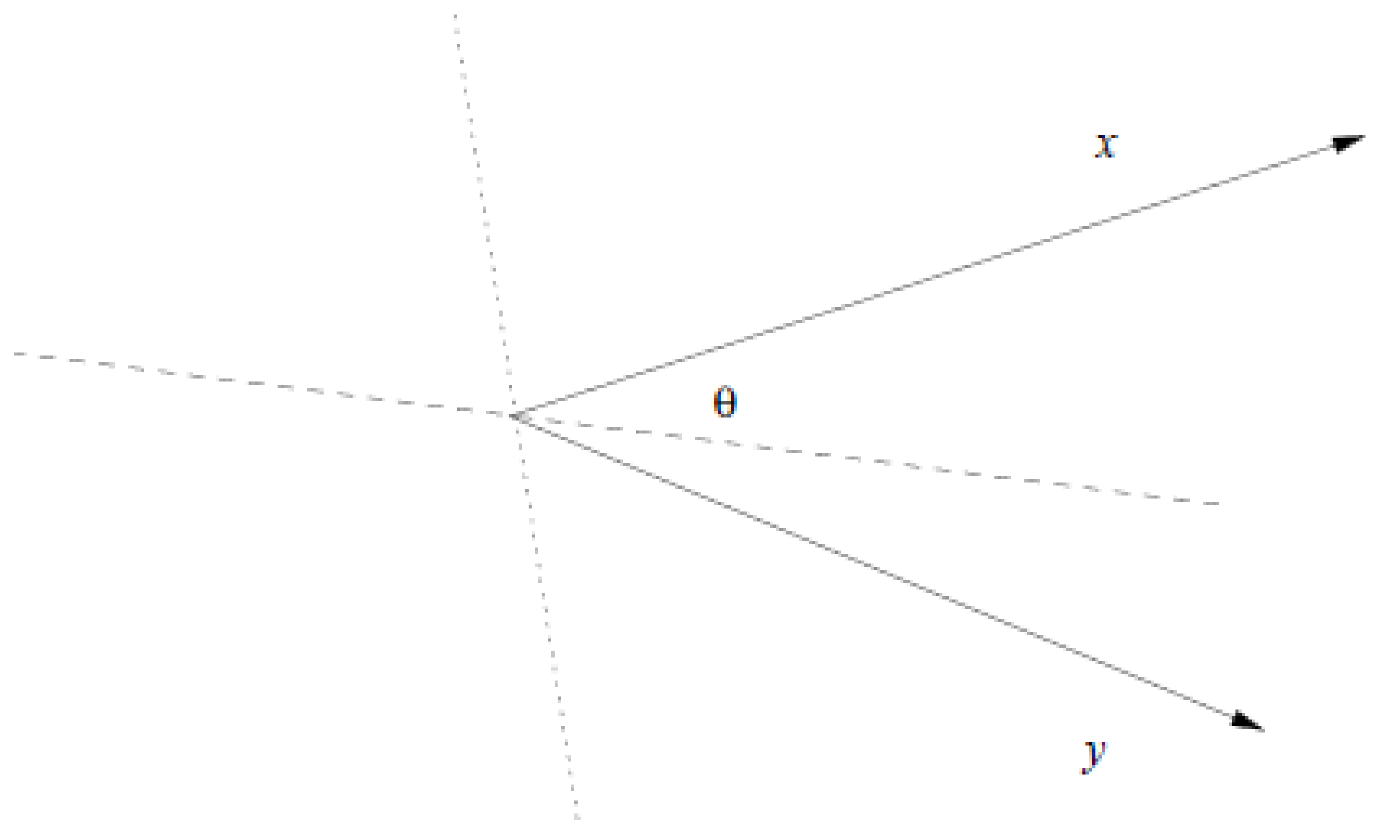


Figure 3.12: Two vectors make an angle θ

Random Hyperplanes and the Cosine Distance

All angles for the line that is the intersection of the random hyperplane and the plane of x and y are equally likely.

Thus, the hyperplane will look like the dashed line with probability θ/180 and will look like the dotted line otherwise.

Thus, each hash function f in our locality-sensitive family F is built from a randomly chosen vector vf .

Given two vectors x and y, say f(x) = f(y) if and only if the dot products vf .x and vf .y have the same sign.

Random Hyperplanes and the Cosine Distance

Then F is a locality-sensitive family for the cosine distance.

That is, F is a (d1, d2, (180 − d1)/180, (180 − d2)/180)-sensitive

Sketches(Signature for cosine)

- Instead of chosing a random vector from all possible vectors, it turns out to be sufficiently random if we restrict our choice to vectors whose components are +1 and −1.
- The dot product of any vector x with a vector v of +1's and −1's is formed by adding the components of x where v is +1 and then subtracting the other components of x – those where v is −1.
- If we pick a collection of random vectors, say v1, v2, . . . , vn, then we can apply them to an arbitrary vector x by computing v1.x, v2.x, . . . , vn.x and then replacing any positive value by +1 and any negative value by −1. The result is called the sketch of x.

Sketches

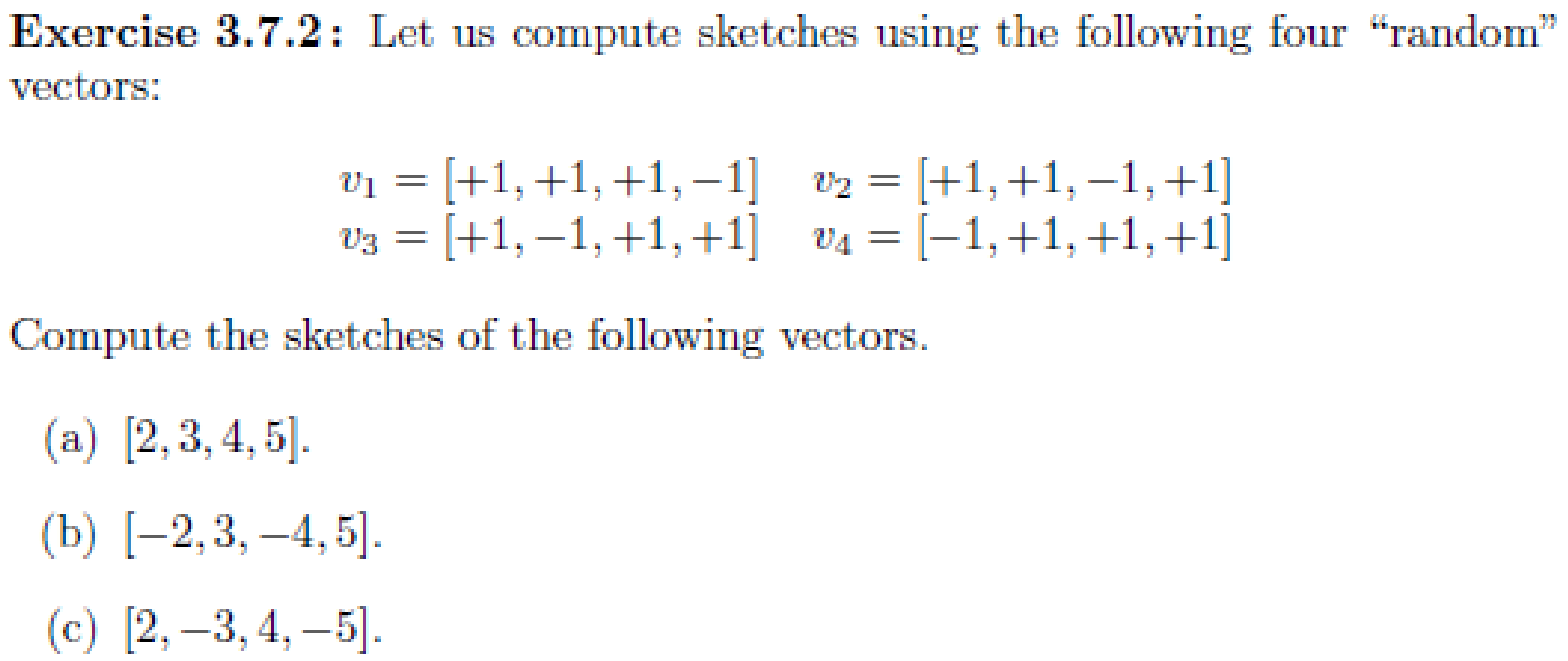
You can handle 0’s arbitrarily, e.g., by chosing a result +1 or −1 at random.(Since there is only a tiny probability of a zero dot product).

Example:

Suppose our space consists of 4-dimensional vectors, and we pick three random vectors: v1 = [+1,−1,+1,+1], v2 = [−1,+1,−1,+1], and v3 = [+1,+1,−1,−1]. For the vector x = [3, 4, 5, 6], the sketch is [+1,+1,−1].

That is, v1.x = 3−4+5+6 = 10. Since the result is positive, the first component of the sketch is +1. Similarly, v2.x = 2 and v3.x = −4, so the second component of the sketch is +1 and the third component is −1.

Sketches



Exercise 3.7.2: Let us compute sketches using the following four "random" vectors:

$$\begin{array}{ll} v_1 = [+1,+1,+1,-1] & v_2 = [+1,+1,-1,+1] \\ v_3 = [+1,-1,+1,+1] & v_4 = [-1,+1,+1,+1] \end{array}$$

Compute the sketches of the following vectors.

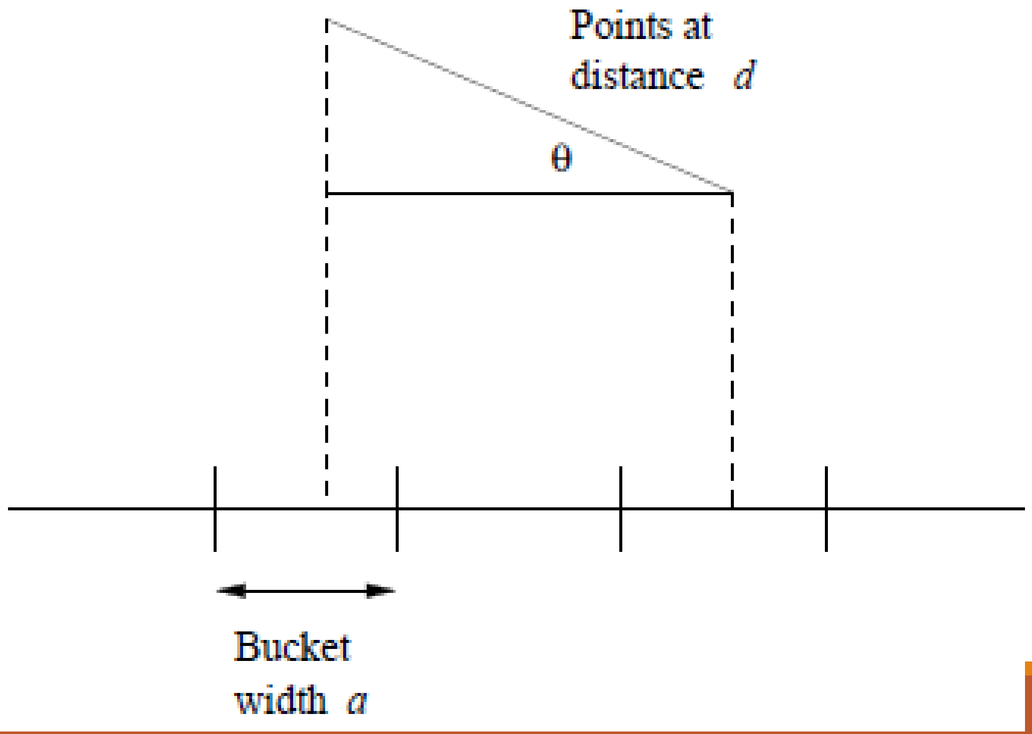
(a) $[2,3,4,5]$.

(b) $[-2,3,-4,5]$.

(c) $[2,-3,4,-5]$.

LSH Families for Euclidean Distance

Two points at distance d ≫ a have a small chance of being hashed to the bucket.



LSH Families for Euclidean Distance

Family F just described forms a (a/2, 2a, 1/2, 1/3)- sensitive family of hash functions.

That is, for distances up to a/2 the probability is at least 1/2 that two points at that distance will fall in the same bucket.

While for distances at least 2a the probability points at that distance will fall in the same bucket is at most 1/3.

Applications of Locality-Sensitive Hashing

1. **Entity Resolution:** This term refers to matching data records that refer to the same real-world entity, e.g., the same person.
2. **Matching Fingerprints:** It is possible to represent fingerprints as sets.
3. **Matching Newspaper Articles:**

Entity Resolution

- It is common to have several data sets available, and to know that they refer to some of the same entities.
- For example, several different bibliographic sources provide information about many of the same books or papers.
- In the general case, we have records describing entities of some type, such as people or books.
- The records may all have the same format, or they may have different formats, with different kinds of information.

An Entity-Resolution Example

- There are many reasons why information about an entity may vary, even if the field in question is supposed to be the same.
- For example, names may be expressed differently in different records because of misspellings, absence of a middle initial, use of a nickname, and many other reasons.
- For example, “Bob S. Jomes” and “Robert Jones Jr.” may or may not be the same person
- If records come from different sources, the fields may differ as well. One source’s records may have an “age” field, while another does not. The second source might have a “date of birth” field.

An Entity-Resolution Example

Real example of how LSH was used to deal with an entity resolution problem.

Company A was engaged by Company B to solicit customers for B. Company B would pay A a yearly fee, as long as the customer maintained their subscription. They later quarreled and disagreed over how many customers A had provided to B. Each had about 1,000,000 records, some of which described the same people; those were the customers A had provided to B. The records had different data fields, but unfortunately none of those fields was “this is a customer that A had provided to B.”

The problem was to match records from the two sets to see if a pair represented the same person.

An Entity-Resolution Example

- The strategy for identifying records involved scoring the differences in three fields: name, address, and phone.
- To create a score describing the likelihood that two records, one from A and the other from B, described the same person, 100 points was assigned to each of the three fields, so records with exact matches in all three fields got a score of 300.
- However, it is not feasible to score all one trillion pairs of records. Thus, a simple LSH was used to focus on likely candidates. Three “hash functions” were used.

An Entity-Resolution Example

In practice, there was no hashing; rather the records were sorted by name, so records with identical names would appear consecutively and get scored for overall similarity of the name, address, and phone.

Validating Record Matches

- In the example at hand, there was an easy way to make that decision, and the technique can be applied in many similar situations.
- It was decided to look at the creation-dates for the records at hand, and to assume that 90 days was an absolute maximum delay between the time the service was bought at Company A and registered at B.
- Thus, a proposed match between two records that were chosen at random, subject only to the constraint that the date on the B-record was between 0 and 90 days after the date on the A-record, would have an average delay of 45 days.

Validating Record Matches

- It was found that of the pairs with a perfect 300 score, the average delay was 10 days.
- If you assume that 300-score pairs are surely correct matches, then you can look at the pool of pairs with any given score s, and compute the average delay of those pairs.
- Suppose that the average delay is x, and the fraction of true matches among those pairs with score s is f. Then $x = 10f + 45(1 - f)$, or $x = 45 - 35f$.
- Solving for f, we find that the fraction of the pairs with score s that are truly matches is $(45 - x)/35$.