IIT KHARAGPUR

IIT GANDHINAGAR

NPTEL ONLINE
CERTIFICATION COURSES

NPTEL

Scalable Data Science

Lecture 10: Frequent Elements: SpaceSaving and CountMin

Anirban Dasgupta

Computer Science and Engineering

IIT GANDHINAGAR

IIT Gandhinagar

Indian Institute of Technology Gandhinagar

Streaming model revisited

- Data is seen as incoming sequence
  - can be just element-ids, or (id, frequency update) tuple
- Arrival only streams
- Arrival + departure
  - Negative updates to frequencies possible
  - Can represent fluctuating quantities, e.g.

Review: Frequency Estimation in one pass

- Given input stream, length m, want a sketch that can answer frequency queries at the end
  - For give item x, return an estimate of the frequency

- Deterministic algorithm by Misra and Gries
  - f_x = original frequency of item x . Return $\widehat{f_x}$ such that

$$f_x - \epsilon m \leq \widehat{f_x} \leq f_x$$

  - Space = $O(\frac{1}{\epsilon}\log n)$

Space Saving Algorithm

- Keep k counters and items in hand

Initialize:

- Set all counters to 0

Process(x)

- if x is same as any item in hand, increment its counter
- else if number of items $< k$, store x with counter $= 1$
- else replace item with smallest counter by x, increment counter

Query(q)

- If q is in hand return its counter, else 0

Example

Analysis

- Smallest counter value, min, is at most ϵm
  - Counters sum to m, by induction
  - $1/\epsilon$ counters, so average is ϵm, hence smallest is less

Analysis

Claim 1: All items with true count $> \epsilon m$ are present in hand at the end

Analysis

<u>Claim 1</u>: All items with true count $> \epsilon m$ are present in hand at the end

- Smallest counter value, min, is at most ϵm
  - Counters sum to m, by induction
  - $1/\epsilon$ counters, so average is ϵm, hence smallest is less
- True count of an uncounted item is between 0 and min
  - Proof by induction, true initially, min increases monotonically
  - Consider last time the item was dropped

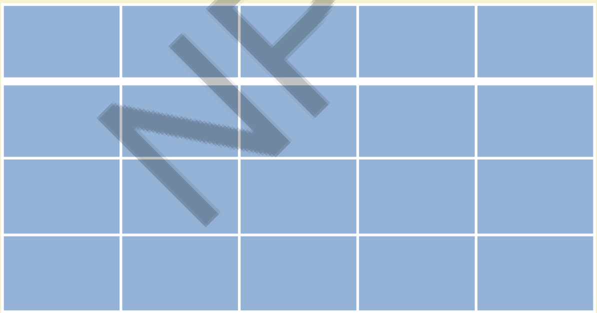
Counter based vs "sketch" based

- Counter based methods
  - Misra-Gries, Space-Saving, ….
  - Work for arrival only streams
  - In practice somewhat more efficient: space, and especially update time

- Sketch based methods
  - "Sketch" is informally defined as a "compact" data structure that allows both inserts and deletes
  - Use hash functions to compute a linear transform of the input
  - Work naturally for arrivals + departure

Count-min sketch

- Model input stream as a vector over U
  - f_x is the entry for dimension x
- Creates a small summary $w \times d$
- Use w hash functions, each maps $U \to [1, d]$



Count Min Sketch

Initialize

- Choose $h_1, .., h_w,\ A[w, d] \leftarrow 0$

Process(x, c):

- For each $i \in [w],\ A[i, h_i(x)] \mathrel{+}= c$

Query(q):

- Return $\min_i A[i, h_i(x)]$

Example

h1			
h2			

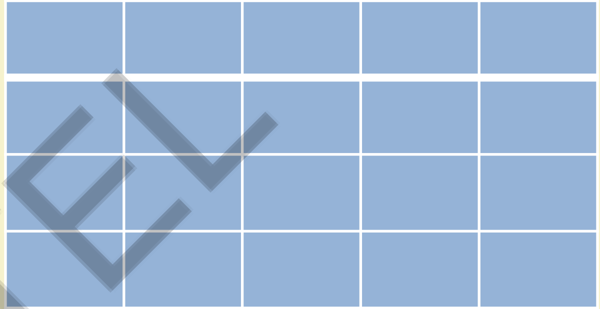
	h1	h2
	2	1
	1	2
	1	3
	3	2

Guarantees

Space = $O(wd)$

Update time = $O(w)$

$x, +c$



Each item is mapped to one bucket per row

Guarantees

- $w = \frac{2}{\epsilon}$ $\mathrm{d} = \log\left(\frac{1}{\delta}\right)$

$Y_1 \ldots . Y_w$ be the w estimates, i.e. $Y_i = A[i, h_i(x)]$, $\widehat{f_x} = \min_i Y_i$

Each estimate $\widehat{f_x}$ always satisfies $\widehat{f_x} \geq f_x$

Guarantees

- $w = \frac{2}{\epsilon}$ $\quad \mathrm{d} = \log\left(\frac{1}{\delta}\right)$

$Y_1 \ldots . Y_w$ be the w estimates, i.e. $Y_i = A[i, h_i(x)]$, $\quad \widehat{f_x} = \min_i Y_i$

Each estimate $\widehat{f_x}$ always satisfies $\widehat{f_x} \geq f_x$

$E[Y_i] = \sum_{y:h_i(y)=h_i(x)} f_y = f_x + \epsilon(m - f_x)/2$

Guarantees

- $w = \frac{2}{\epsilon} \quad \mathrm{d} = \log\left(\frac{1}{\delta}\right)$

$Y_1 \ldots . Y_w$ be the w estimates, i.e. $Y_i = A[i, h_i(x)], \quad \widehat{f_x} = \min_i Y_i$

Each estimate $\widehat{f_x}$ always satisfies $\widehat{f_x} \geq f_x$

$$E[Y_i] = \sum_{y: h_i(y) = h_i(x)} f_y = f_x + \epsilon(m - f_x)/2$$

Applying Markov's inequality,

$$\Pr[Y_i - f_x > \epsilon m] \leq \frac{\epsilon(m - f_x)}{2\epsilon m} \leq \frac{1}{2}$$

Guarantee

- Since we are taking minimum of $\log\left(\frac{1}{\delta}\right)$ such random variables,

$$\Pr\left[\,\widehat{f_x} > f_x + \epsilon m\right] \leq 2^{-\log\left(\frac{1}{\delta}\right)} \leq \delta$$

Guarantee

- Since we are taking minimum of $\log\left(\frac{1}{\delta}\right)$ such random variables,

$$\Pr\left[\,\widehat{f_x} > f_x + \epsilon m\right] \leq 2^{-\log\left(\frac{1}{\delta}\right)} \leq \delta$$

- Hence, with probability $1 - \delta$, for any query x

$$f_x \leq \widehat{f_x} \leq f_x + \epsilon m$$

Summary

- Two algorithms for frequency estimation
  - Counter based: Space Saving
  - Sketch based: Count-Min
- Guiding principle: use error bounds as design parameters of the data structure
- More to come…

References:

- Primary references for this lecture
  - Lecture slides by Graham Cormode *http://dmac.rutgers.edu/Workshops/WGUnifyingTheory/Slides/cormode.pdf*
  - Lecture notes by Amit Chakrabarti: http://www.cs.dartmouth.edu/~ac/Teach/data-streams-lecnotes.pdf
  - Sketch techniques for approximate query processing, Graham Cormode. http://dimacs.rutgers.edu/~graham/pubs/papers/sk.pdf

Thank You!!

IIT KHARAGPUR

IIT GANDHINAGAR

NPTEL ONLINE
CERTIFICATION COURSES

NPTEL

Scalable Data Science

Lecture 10: Frequent Elements: CountSketch

Anirban Dasgupta
Computer Science and Engineering
IIT GANDHINAGAR

IIT Gandhinagar
Indian Institute of Technology Gandhinagar

Streaming model revisited

- Data is seen as incoming sequence
  - can be just element-ids, or (id, frequency update) tuple
- Arrival only streams
- Arrival + departure
  - Negative updates to frequencies possible
  - Can represent fluctuating quantities, e.g.

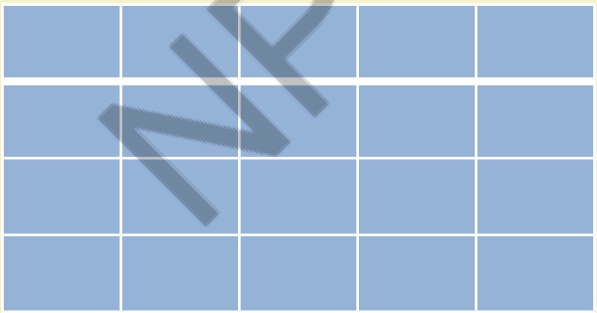
Review: Frequency Estimation in one pass

- Given input stream, length m, want a sketch that can answer frequency queries at the end
  - For give item x, return an estimate of the frequency

- Algorithms seen
  - Deterministic counter based algorithms: Misra-Gries, SpaceSaving
  - Count-Min sketch

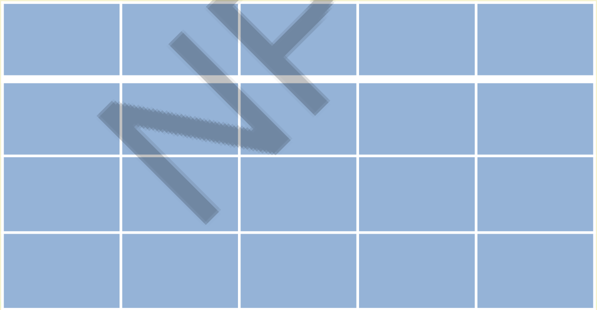
Recall: Count-min sketch

- Model input stream as a vector over U
  - f_x is the entry for dimension x
- Creates a small summary $w \times d$
- Use w hash functions, each maps $U \rightarrow [1, d]$



Count-sketch

- Model input stream as a vector over U
  - f_x is the entry for dimension x
- Creates a small summary $w \times d$
- Use w hash functions, $h_i: U \rightarrow [1, d]$
- w sign hash function, each maps $\mathrm{g_i}$: $U \rightarrow \{-1, +1\}$



Count Min Sketch

Initialize

- Choose $h_1, .., h_w,\ A[w, d] \leftarrow 0$

Process(x, c):

- For each $i \in [w],\ A[i, h_i\,(x)] += c \times g_i(x)$

Query(q):

- Return median$\{g_i(x)A[i, h_i(x)]\}$

Example

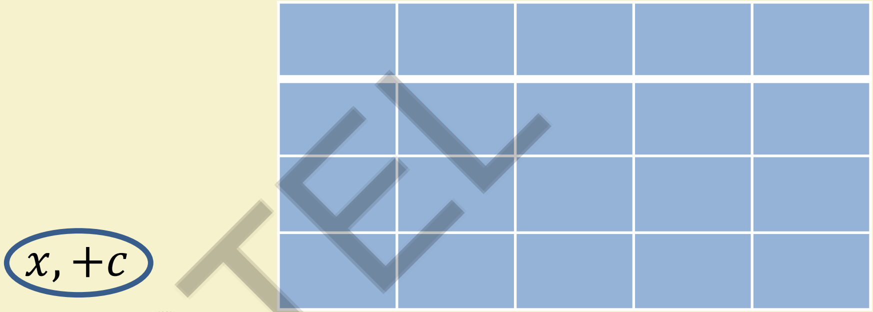
h1			
h2			

	h1,g1	h2,g2
	2,+	1,+
	3,-	2,+
	1,+	3,-
	2,-	3,+

Guarantees

Space = $O(wd)$

Update time = $O(w)$



Each item is mapped to one bucket per row

Guarantees

- $w = \frac{2}{\epsilon^2} \quad \mathrm{d} = \log\left(\frac{1}{\delta}\right)$

$Y_1 \ldots . Y_w$ be the w estimates, i.e. $Y_i = g_i(x) A[i, h_i(x)], \quad \widehat{f_x} = \underset{i}{\text{median}}\, Y_i$

$$E[Y_i] = E\left[g_i(x)\, A[i, h_i(x)]\right] = E\left[g_i(x) \sum_{h_i(y)=h_i(x)} f_y\, g_i(y)\right]$$

Guarantees

$$E[Y_i] = E\left[g_i(x)\, A[i, h_i(x)]\right] = E\left[g_i(x) \sum_{h_i(y)=h_i(x)} f_y\, g_i(y)\right]$$

Notice that for $x \neq y$, $E[g_i(x)\; g_i(y)] = 0$!

$$E[Y_i] = g_i(x)^2 f_x = f_x$$

We analyse the variance in order to bound the error

For simplicity assume hash functions all independent

Variance analysis

$$|f|_2^2 = \sum_x f_x^2$$

Using simple algebra, as well as independence of hash functions,

$$var(Y_i) = \frac{\left(\sum_y f_y^2 \ - \ f_x^2\right)}{d} \le \frac{{|f|_2}^2}{d}$$

Using Chebyshev's inequality

$$\Pr[\ |Y_i \ - f_x| > \epsilon |f|_2] \le \frac{1}{d\epsilon^2} \le \frac{1}{3}$$

$$d = \frac{3}{\epsilon^2}$$

Finally, use analysis of median-trick with $w = \log\left(\frac{1}{\delta}\right)$

Final Guarantees

- Using space $O\left(\frac{1}{\epsilon^2}\log\left(\frac{1}{\delta}\right)\log(n)\right)$, for any query x, we get an estimate, with prob $1 - \delta$

$$f_x - \epsilon|f|_2 \leq \widehat{f_x} \leq f_x + \epsilon|f|_2$$

Comparisons

Algorithm	$\widehat{f_x} - f_x$	Space $\times\, log(n)$	Error prob	Model
Misra-Gries	$[-\epsilon\|f\|_1, 0]$	$1/\epsilon$	0	Insert Only
SpaceSaving	$[0, \epsilon\|f\|_1]$	$1/\epsilon$	0	Insert Only
CountMin	$[0, \epsilon\|f\|_1]$	$\log\left(\frac{1}{\delta}\right)/\epsilon$	δ	Insert
CountSketch	$[-\epsilon\|f\|_2, \epsilon\|f\|_2]$	$\log\left(\frac{1}{\delta}\right)/\epsilon^2$	δ	Insert+Delete

Summary

- CM and Count Sketch to answer point queries about frequencies
  - two user-defined parameters, ϵ and δ
  - Linear sketch, hence can be combined across distributed streams
- Count Sketch handle departures naturally
  - As long as –ve frequencies are not present
  - For CM, we need to consider median instead of minm
- Extensions to handle range queries and others…
- Actual performance much better than theoretical bound

References:

- Primary references for this lecture
    - Lecture slides by Graham Cormode
      http://dmac.rutgers.edu/Workshops/WGUnifyingTheory/Slides/cormode.pdf
    - Lecture notes by Amit Chakrabarti: http://www.cs.dartmouth.edu/~ac/Teach/data-streams-lecnotes.pdf
    - Sketch techniques for approximate query processing, Graham Cormode.
      http://dimacs.rutgers.edu/~graham/pubs/papers/sk.pdf

Thank You!!

IIT KHARAGPUR

IIT GANDHINAGAR

NPTEL ONLINE CERTIFICATION COURSES

NPTEL

Scalable Data Science

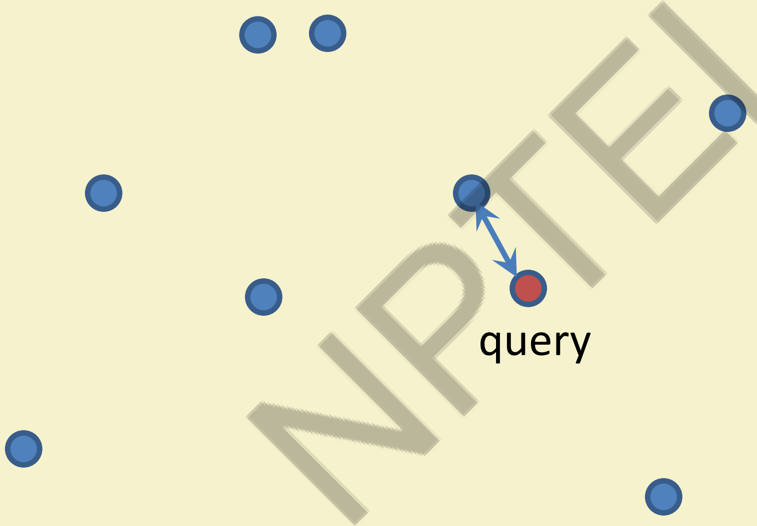
Lecture 11: Near Neighbors

Anirban Dasgupta

Computer Science and Engineering

IIT GANDHINAGAR

Finding Near Neighbors



Given a set of data points and a query

Can we find what is the nearest datapoint to the query?

- K-nearest neighbors
- $d(p, query) < r$

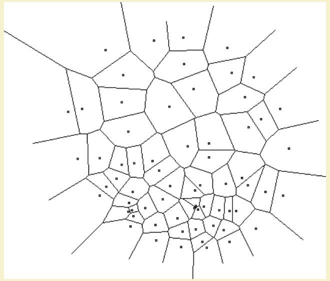
Applications

- Numerous
  - Finding near duplicate webpages / articles
  - Finding similar images in search
  - Clustering
  - Nearest neighbour classifier
- Variants
  - all pairs near neighbors

Naïve solution?

- Naïve scan
  - $O(nd)$ time for each query

- Can we calculate and store the Voronoi partition of the point-set?
  - Will give the exact answer if possible
  - needs $n^{d/2}$ storage for n points in d dimensions



Space Partitioning trees

- Basic idea
  - Recursively partition the space
  - Given the query, prune the dataset using the created partition tree
  - All depends on how to partition

Kd-trees

- Works “well” for “low to medium” dimensions
- Initially proposed by Bentley 1970
- Originally, k was #dimensions
- Idea: each level of the tree uses a single dimension to partition

Algorithm

- Each level has a cutting dimension
- Cycle through the dimensions
- At every step, choose the point which is the median along that dimension, create an axis-aligned partition

Example

Complexity

- Space taken = $O(n)$
- Nearest neighbour search:
  - Defeatist search: only search the child that contain the query point
  - Descending search: maintain the current near neighbour and distance to it. Visit one or both children depending on whether there is intersection
  - Priority search: Maintain a priority queue of the regions depending on distance.
  - Can potentially take $O(n)$

Variants

- Several variants of space partitioning trees possible
  - Random Projection tree chooses a unit direction at random for every node
  - PD tree uses the principal eigenvector of the covariance matrix
  - 2-Mean tree : partition the data into 2 clusters, find the hyperplane that bisects the line connecting them

Possible intuition to analyze

- Does the partitioning algorithm adapt to “intrinsic dimension” ?
  - i.e. if the data has some low-dimensional structure
  - E.g. if the data has “intrinsic dimension” d, then all cells O(d) levels below a cell C has at most ½ the diameter of C

Possible intuition to analyze

- Does the partitioning algorithm adapt to "intrinsic dimension" ?
  - i.e. if the data has some low-dimensional structure
  - E.g. if the data has "intrinsic dimension" d, then all cells O(d) levels below a cell C has at most ½ the diameter of C
- Definition of "intrinsic dimension" is not obvious
  - Ex: covariance dimension is d if the d largest eigenvalues of covariance matrix account for $1 - \epsilon$ fraction of trace

Possible way to analyze

- Does the partitioning algorithm adapt to “intrinsic dimension” ?
  - i.e. if the data has some low-dimensional structure
  - E.g. if the data has “intrinsic dimension” d, then all cells O(d) levels below a cell C has at most ½ the diameter of C
- Definition of “intrinsic dimension” is not obvious
  - Ex: covariance dimension is d if the d largest eigenvalues of covariance matrix account for $1 - \epsilon$ fraction of trace
- Can be shown that RP, PD trees adapt to this dimension, but k-D tree does not

Summary

- Nearest neighbour question
- Number of algorithms for low dimensional data based on space partitioning trees
  - Some of the adapt to the intrinsic dimensionality of data

References:

- Primary references for this lecture
  - Foundations of multidimensional and metric data structures, H. Samet. Morgan Kaufman 2006.
  - “Which space partitioning trees adapt to Intrinsic Dimension”, Verma, Kpotfe, Dasgupta UAI 2009.

Thank You!!

IIT KHARAGPUR

IIT GANDHINAGAR

NPTEL ONLINE
CERTIFICATION COURSES

NPTEL

Scalable Data Science

Lecture 12: Locality Sensitive Hashing

Anirban Dasgupta

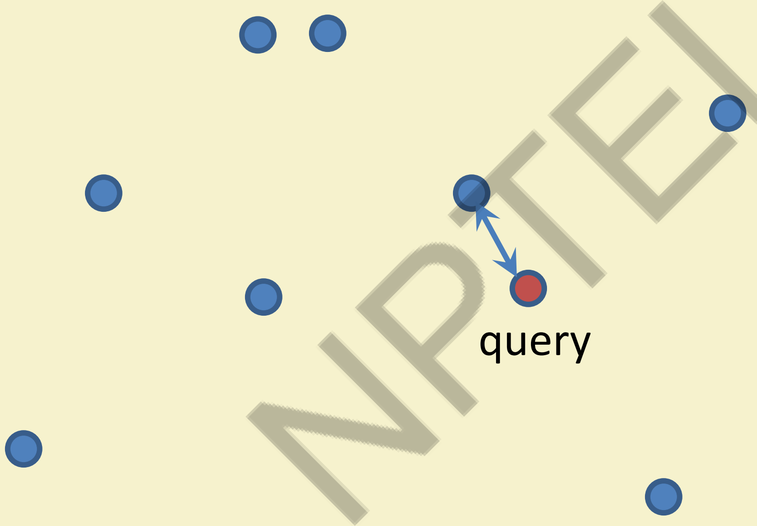
Computer Science and Engineering

IIT GANDHINAGAR

IIT Gandhinagar

Indian Institute of Technology Gandhinagar

Finding Near Neighbors



Given a set of data points and a query

Can we find what is the nearest datapoint to the query?

- K-nearest neighbors
- d(p, query) < r

Defining representation

- We need to define the object representation and distance functions first
  - e.g. is a document just a (multi-)set of characters, or a word X position matrix?

- Mainly a few standard ways of representing
  - documents as sets
  - images / other objects as vectors

Documents as sets

- Shingle: a set of k consecutive characters that appear in the document
- Document = set of shingles
  - Often are hashed to 64bit numbers for each of storage

A sly fox jumped over the lazy hen → a sly
sly f
ly fo
....

Distance function for sets

- Jaccard similarity $\mathrm{JS}(A, B) = \frac{|A \cap B|}{|A \cup B|}$

- Distance $\mathrm{JD}(A, B) = 1 - JS(A, B)$

Vectors

- Images
  - Vectors over 64x64 or 128x128
- Documents
  - Sets → vectors, possibly with TF-IDF or other weighting
- Distance functions
  - $\ell_2(x,y) = \left(\sum_i (x_i - y_i)^2\right)^{1/2}$
  - …
  - angle between vectors

Hash Tables

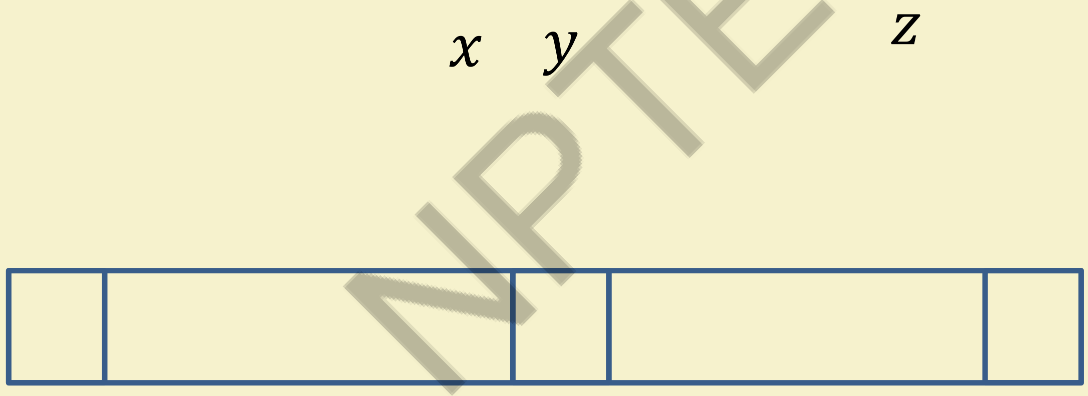
- For exact search we used hashing
- Can we adapt hashing to search for “near”?

Hash Tables

- For exact search we used hashing
- Can we adapt hashing to search for “near”?
- Repurpose “collision”
  - Instead of trying to avoid collisions, now we try to make collisions happen if the data points are nearby

Hash Tables

- Want the following
  - Nearby points should fall in “same” bucket, points further away should fall in different buckets



Locality Sensitive Hashing

[Indyk Motwani]

- Hash family H is *locality sensitive* if

$\Pr[h(x) = h(y)]$ is high if x is close to y

$\Pr[h(x) = h(y)]$ is low if x is far from y

- Not clear such functions exist for all distance functions

Locality sensitive hashing

- Originally defined in terms of a similarity function [C'02]

- Given universe U and a similarity $s: U \times U \rightarrow [0,1]$, does there exist a prob distribution over some hash family H such that

$$\Pr_{h \in H} [h(x) = h(y)] = s(x, y)$$

$$s(x, y) = 1 \rightarrow x = y$$
$$s(x, y) = s(y, x)$$

Hamming distance

- Points are bit strings of length d
- $H(x, y) = |\{i, x_i \neq y_i\}|$

Hamming distance

- Points are bit strings of length d
- $H(x, y) = |\{i, x_i \neq y_i\}|$ $\quad S_H(x, y) = 1 - \frac{H(x,y)}{d}$
  - $x = 1011010001, y = 0111010101$
  - $H(x, y) = 3$ $\quad S_H(x, y) = 1 - \frac{3}{10} = 0.7$

Hamming distance

- Points are bit strings of length d
- $H(x, y) = |\{i, x_i \neq y_i\}|$ $\quad S_H(x, y) = 1 - \frac{H(x,y)}{d}$
- Define a hash function h by sampling a set of positions
  - $x = 1011010001, y = 0111010101$
  - $S = \{1,5,7\}$
  - $h(x) = 100, h(y) = 100$

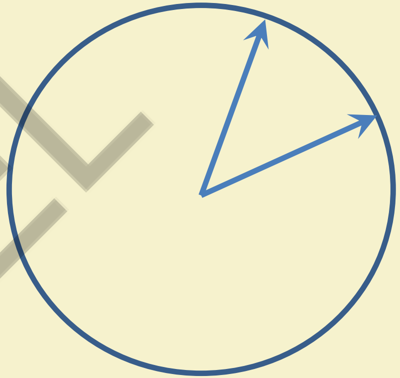
Existence of LSH

- The above hash family is locality sensitive, $k = |S|$

$$\Pr[h(x) = h(y)] = \left(1 - \frac{H(x,y)}{d}\right)^k$$

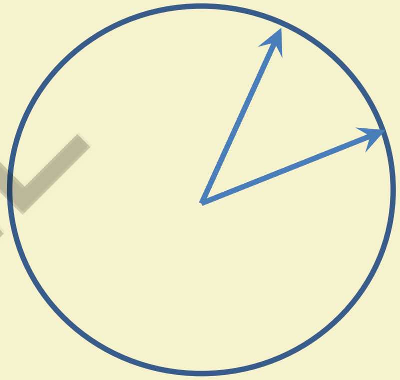
LSH for angle distance

- x, y are unit norm vectors
- $d(x, y) = \cos^{-1}(x \cdot y) = \theta$
- $S(x, y) = 1 - \theta/\pi$



LSH for angle distance

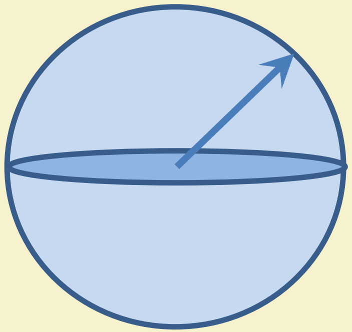
- x, y are unit norm vectors
- $d(x, y) = \cos^{-1}(x \cdot y) = \theta$
- $S(x, y) = 1 - \theta/\pi$



- Choose direction v uniformly at random
  - $h_v(x) = sign(v \cdot x)$
  - $\Pr[h_v(x) = h_v(y)] = 1 - \theta/\pi$

Aside: picking a direction u.a.r.

- How to sample a vector $x \in R^d, |x|_2 = 1$ and the direction is uniform among all possible directions



- Generate $x = (x_1, \ldots . x_d), x_i \sim N(0,1)$ iid
- Normalize $\frac{x}{|x|_2}$
  - By writing the pdf of the d-dimensional Gaussian in polar form, easy to see that this is uniform direction on unit sphere

Jaccard distance: minhashing

- Pick a uniform permutation of the element universe U
- For any set S,
  - $h(S) = min_{x \in S}\, h(x)$

- Often easier to visualize if we think of the collection of sets as a $\{0,1\}$ matrix

Example

	S_1	S_2	S_3	S_4
A	1	0	1	0
B	1	0	0	1
C	0	1	0	1
D	0	1	0	1
E	0	1	0	1
F	1	0	1	0
G	1	0	1	0

→

A
C
G
F
B
E
D

→

		S_1	S_2	S_3	S_4
1	**A**	1	0	1	0
2	**C**	0	1	0	1
3	**G**	1	0	1	0
4	**F**	1	0	1	0
5	**B**	1	0	0	1
6	**E**	0	1	0	1
7	**D**	0	1	0	1
		1	**2**	**1**	**2**

[Slide from Evimaria Terzi]

Example

	S_1	S_2	S_3	S_4
A	1	0	1	0
B	1	0	0	1
C	0	1	0	1
D	0	1	0	1
E	0	1	0	1
F	1	0	1	0
G	1	0	1	0

D
B
A
C
F
G
E

		S_1	S_2	S_3	S_4
1	**D**	0	1	0	1
2	**B**	1	0	0	1
3	**A**	1	0	1	0
4	**C**	0	1	0	1
5	**F**	1	0	1	0
6	**G**	1	0	1	0
7	**E**	0	1	0	1

2	**1**	**3**	**1**

Why is this LSH?

- For sets S and T,
  - The first row where one of the two has a 1 belong to $S \cup T$
  - We have equality $h(S) = h(T)$, only if both the rows contain 1
  - This means that this row belongs to $S \cap T$

- Hence, the event $h(S) = h(T)$ is same as the event that a row in $S \cap T$ appears first among all rows in $S \cup T$

$$\Pr[h(S) = h(T)] = \frac{|S \cap T|}{|S \cup T|}$$

Aside: How to choose random permutations

- Picking a uniform at random permutation is expensive
- In theory, need to choose from a family of min-wise independent permutations

- In practice, can use standard hash functions, hash all the values and then sort

Which similarities admit LSH?

- There are various similarities and distance that are used in scientific literature
  - Encyclopedia of distances DL’11
- Will there be an LSH for each one of them?
  - Similarity is LSHable if there exists an LSH for it

[slide courtesy R. Kumar]

LSHable similarities

<u>Thm</u>: S is LSHable → 1 – S is a metric

$$d(x, y) = 0 \ \rightarrow x = y$$
$$d(x, y) = d(y, x)$$
$$d(x, y) + d(y, z) \geq d(x, z)$$

Fix hash function $h \in H$ and define

$$\Delta_h(A, B) = [h(A) \neq h(B)]$$
$$1 \ - \mathrm{S(A, B)} = \Pr_h[\Delta_h(A, B)]$$

LSHable similarities

Thm: S is LSHable → 1 – S is a metric

$$d(x,y) = 0 \ \rightarrow x = y$$
$$d(x,y) = d(y,x)$$
$$d(x,y) + d(y,z) \geq d(x,z)$$

Fix hash function $h \in H$ and define

$$\Delta_h(A,B) = [h(A) \neq h(B)]$$

LSHable similarities

<u>Thm</u>: S is LSHable → 1 – S is a metric

$$d(x,y)=0 \;\rightarrow x=y$$
$$d(x,y)=d(y,x)$$
$$d(x,y)+d(y,z)\geq d(x,z)$$

Fix hash function $h \in H$ and define

$$\Delta_h(A,B)=[h(A)\neq h(B)]$$
$$1\;-\mathrm{S}(\mathrm{A},\mathrm{B})=\Pr_h[\Delta_h(A,B)]$$

Also

$$\Delta_h(A,B)+\Delta_h(B,C)\geq\Delta_h(A,C)$$

Example of non-LSHable similarities

- $d(A, B) = 1 - s(A, B)$
- Sorenson-Dice : $s(A, B) = \frac{2|A \cap B|}{|A| + |B|}$
  - Ex: $A = \{a\}, B = \{b\}, C = \{a, b\}$
  - $s(A, B) = 0, s(B, C) = s(A, C) = 2/3$

Example of non-LSHable similarities

- $d(A,B) = 1 - s(A,B)$
- Sorenson-Dice : $s(A,B) = \frac{2|A \cap B|}{|A|+|B|}$
  - Ex: $A = \{a\}, B = \{b\}, C = \{a,b\}$
  - $s(A,B) = 0, s(B,C) = s(A,C) = \frac{2}{3}$
- Overlap: $s(A,B) = \frac{|A \cap B|}{\min(|A|,|B|)}$
  - $s(A,B) = 0, s(A,C) = 1 = s(B,C)$

Example of non-LSHable similarities

- $d(A,B) = 1 - s(A,B)$
- Sorenson-Dice : $s(A,B) = \frac{2|A \cap B|}{|A|+|B|}$
  - Ex: $A = \{a\}, B = \{b\}, C = \{a,b\}$
  - $s(A,B) = 0, s(B,C) = s(A,C) = \frac{2}{3}$
- Overlap: $s(A,B) = \frac{|A \cap B|}{\min(|A|,|B|)}$
  - $s(A,B) = 0, s(A,C) = 1 = s(B,C)$

These similarities are not LSHable

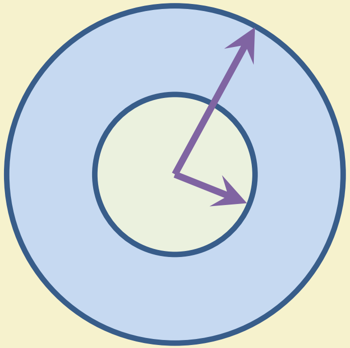
Gap Definition of LSH

IMRS'97, IM'98, GIM'99

- A family is (r, R, p, P) LSH if

$$\Pr_{h \in H}[h(x) = h(y)] \geq p \; if \; d(x,y) \leq r$$

$$\Pr_{h \in H}[h(x) = h(y)] \leq P \; if \; d(x,y) \geq R$$



Gap LSH

- All the previous constructions satisfy the gap definition
  - Ex: for $JS(S,T) = \frac{|S \cap T|}{|S \cup T|}$

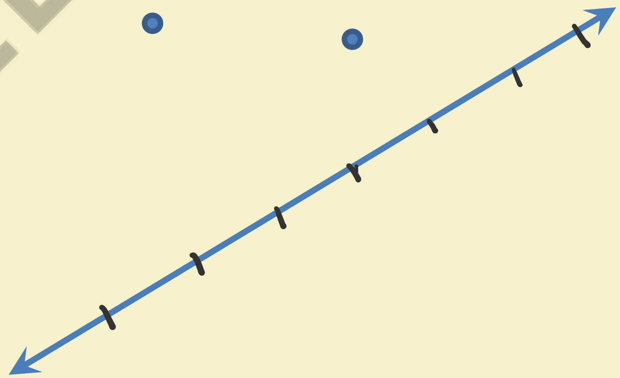
$$JD(S,T) \leq r \rightarrow JS(S,T) \geq 1 - r \rightarrow \Pr[h(S) = h(T)] = JS(S,T) \geq 1 - r$$

$$JD(S,T) \geq R \rightarrow JS(S,T) \leq 1 - R \rightarrow \Pr[h(S) = h(T)] = JS(S,T) \leq 1 - R$$

Hence is a $(r, R, 1-r, 1-R)$ LSH

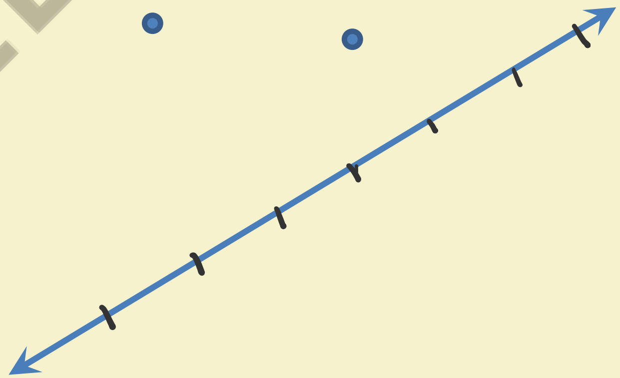
L2 norm

- $d(x, y) = \sqrt{(\sum_i (x_i - y_i)^2}$
- $u =$ random unit norm vector, $w \in R$ parameter, $b \sim Unif[0, w]$
- $h(x) = \lfloor \frac{u \cdot x + b}{w} \rfloor$



L2 norm

- $d(x, y) = \sqrt{(\sum_i (x_i - y_i)^2}$
- u = random unit norm vector, $w \in R$ parameter, $b \sim Unif[0, w]$
- $h(x) = \lfloor \frac{u \cdot x + b}{w} \rfloor$
- If $|x - y|_2 < \frac{w}{2}$, $\Pr[h(x) = h(y)] \geq \frac{1}{3}$
- If $|x - y|_2 > 4w$, $\Pr[h(x) = h(y)] \leq \frac{1}{4}$

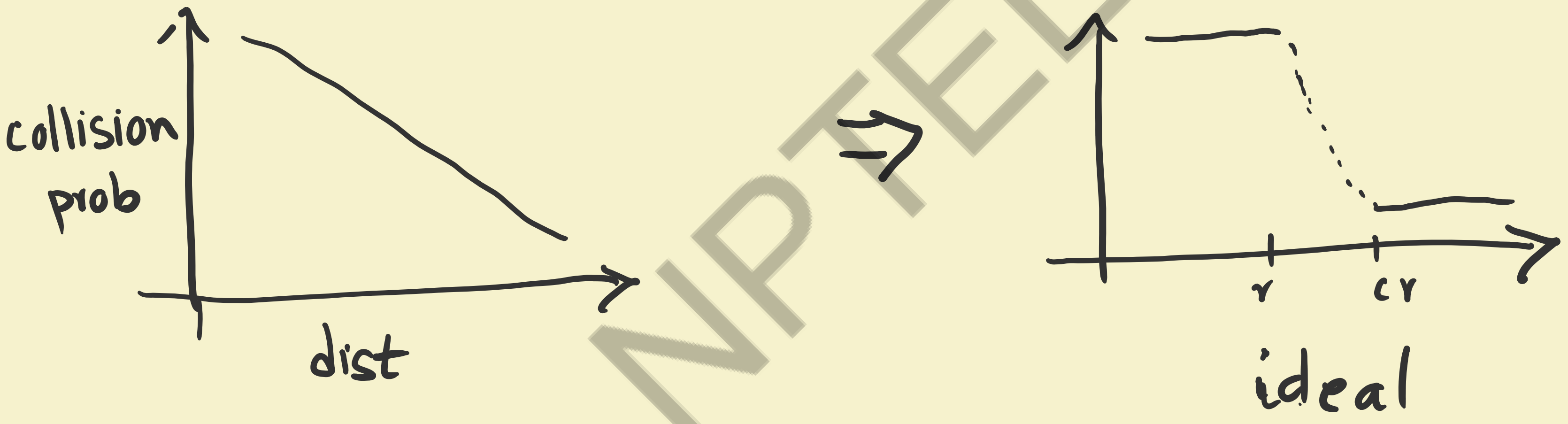


Solving the near neighbour

- (r, c) −near neighbour problem
  - Given query point q, return all points p such that $d(p,q) < r$ and none such that $d(p,q) > cr$
  - Solving this gives a subroutine to solve the "nearest neighbour", by building a data-structure for each r , in powers of $(1 + \epsilon)$

How to actually use it?

- Need to amplify the probability of collisions for “near” points



Band construction

- AND-ing of LSH
  - Define a composite function $H(x) = (h_1(x), \dots h_k(x))$
  - $\Pr[H(x) = H(y)] = \Pi_\mathrm{i} \Pr[h_i(x) = h_i(y)] = \Pr[h_1(x) = h_1(y)]^k$

Band construction

- AND-ing of LSH
  - Define a composite function $H(x) = (h_1(x), \dots h_k(x))$
  - $\Pr[H(x) = H(y)] = \Pi_{\mathrm{i}} \Pr[h_i(x) = h_i(y)] = \Pr[h_1(x) = h_1(y)]^k$

- OR-ing
  - Create L independent hash-tables for $H_1, H_2, \dots H_L$
  - Given query q, search in $\cup_j H_j(q)$

Example

	S_1	S_2	S_3	S_4
A	1	0	1	0
B	1	0	0	1
C	0	1	0	1
D	0	1	0	1
E	0	1	0	1
F	1	0	1	0
G	1	0	1	0

	S1	S2	S3	S3
h1	1	2	1	2
h2	2	1	3	1

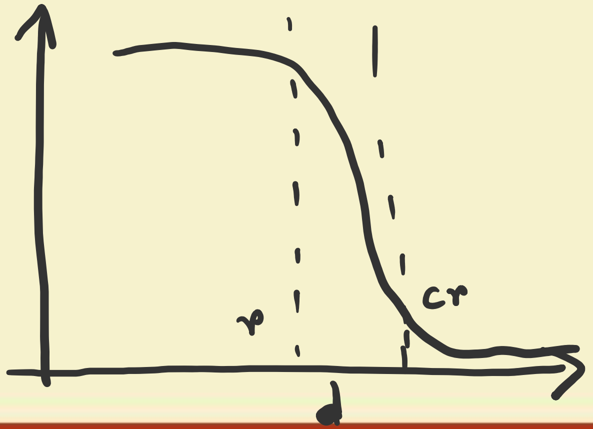
	S1	S2	S3	S3
h3	3	1	2	1
h4	1	3	2	2

Why is this better?

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

- Probability of not finding y as one of the candidates in $\cup_j H_j(q)$

$$1 - \left(1 - (1 - d)^k\right)^L$$



Creating an LSH

- If we have a (r, cr, p, q) LSH
- For any y, with $|q - y| < r$,
  - Prob of y as candidate in $\cup_j H_j(q) \geq 1 - \left(1 - p^k\right)^L$
- For any z, $|q - z| > cr$,
  - Prob of z as candidate in any fixed $H_j(q) \leq q^k$
  - Expected number of such $z \leq Lq^k$

Creating an LSH

$\rho = \frac{\log(p)}{\log(q)}$ $L = n^{\rho}$ $k = \log(n)/\log\left(\frac{1}{q}\right)$

- If we have a (r, cr, p, q) LSH
- For any y, with $|q - y| < r$,
  - Prob of y as candidate in $\cup_j H_j(q) \geq 1 - \left(1 - p^k\right)^L \geq 1 - \frac{1}{e}$
- For any z, $|q - z| > cr$,
  - Prob of z as candidate in any fixed $H_j(q) \leq q^k$
  - Expected number of such $z \leq Lq^k \leq L = n^{\rho}$

Runtime

- Space used = $n^{1+\rho}$
- Query time = n^{ρ}

- We can show that for Hamming, angle etc, $\rho \approx \frac{1}{c}$
  - Can get 2-approx near neighbors in $O(\sqrt{n})$ query time

LSH: theory vs practice

- In order to design LSH in practice, the theoretical parameter values are only a guidance
  - Typically need to search over the parameter space to find a good operating point
  - Data statistics can provide some guidance (will see in next class)

Summary

- Locality sensitive hashing is a powerful tool for near neighbour problems
- Trades off space with query time
- Practical for medium to large datasets with fairly large number of dimensions
  - However, doesn’t really work very well for sparse, very very high dimensional datasets
- LSH and extensions are an area of active research and practice

References:

- Primary references for this lecture
  - Modern Massive Datasets, Rajaraman, Leskovec, Ullman.
  - Survey by Andoni et al. (CACM 2008) available at www.mit.edu/~andoni/LSH

Thank You!!