CS-GY 9223 I: Lecture 4
Near neighbor search + locality sensitive hashing

NYU Tandon School of Engineering, Prof. Christopher Musco

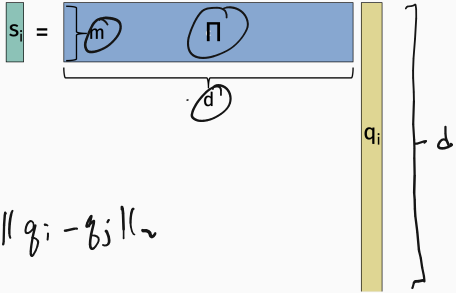
ADMINISTRATIVE

- Problem set 1.
- Reading group.

Lemma (Johnson-Lindenstrauss, 1984)

For any set of n data points $\mathbf{q}_1, \ldots, \mathbf{q}_n \in \mathbb{R}^d$ there exists a linear map $\boldsymbol{\Pi} : \mathbb{R}^d \to \mathbb{R}^m$ where $m = O\left(\frac{\log n}{\epsilon^2}\right)$ such that for all i, j,

$$(1-\epsilon)\|\mathbf{q}_i - \mathbf{q}_j\|_2 \leq \|\overset{s_i}{\boldsymbol{\Pi}\mathbf{q}_i} - \overset{s_j}{\boldsymbol{\Pi}\mathbf{q}_j}\|_2 \leq (1+\epsilon)\|\mathbf{q}_i - \mathbf{q}_j\|_2.$$



$m = O\left(\frac{\log n}{\epsilon^2}\right)$

$\|s_i - s_j\|_2 \approx \|q_i - q_j\|_2$

$\mathbf{\Pi} \in \mathbb{R}^{k\times d}$ be chosen so that each entry equals $\frac{1}{\sqrt{m}}\mathcal{N}(0,1)$.

... or each entry equals $\frac{1}{\sqrt{m}} \pm 1$ with equal probability.

-2.1384	2.9080	-0.3538	0.0229	0.5201	-0.2938	-1.3320	-1.3617	-0.1952
-0.8396	0.8252	-0.8236	-0.2620	-0.0200	-0.8479	-2.3299	0.4550	-0.2176
1.3546	1.3790	-1.5771	-1.7502	-0.0348	-1.1201	-1.4491	-0.8487	-0.3031
-1.0722	-1.0582	0.5080	-0.2857	-0.7982	2.5260	0.3335	-0.3349	0.0230
0.9610	-0.4686	0.2820	-0.8314	1.0187	1.6555	0.3914	0.5528	0.0513
0.1240	-0.2725	0.0335	-0.9792	-0.1332	0.3075	0.4517	1.0391	0.8261
1.4367	1.0984	-1.3337	-1.1564	-0.7145	-1.2571	-0.1303	-1.1176	1.5270
-1.9609	-0.2779	1.1275	-0.5336	1.3514	-0.8655	0.1837	1.2607	0.4669
-0.1977	0.7015	0.3502	-2.0026	-0.2248	-0.1765	-0.4762	0.6601	-0.2097
-1.2078	-2.0518	-0.2991	0.9642	-0.5890	0.7914	0.8620	-0.0679	0.6252

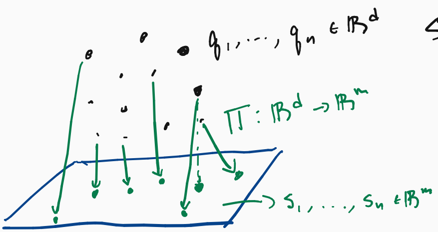
```
>> Pi = randn(m,d);
>> s = (1/sqrt(m))*Pi*q;
```

1	1	-1	-1	-1	-1	-1	-1	1	-1	-1	1	-1	-1	1	1	-1
1	1	1	-1	1	-1	-1	-1	1	1	1	1	-1	1	-1	-1	-1
1	1	-1	-1	-1	1	-1	-1	1	1	-1	1	-1	1	-1	1	-1
-1	-1	-1	1	1	-1	-1	-1	-1	-1	-1	-1	-1	1	1	1	1
1	-1	1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	1	1	-1	1
1	-1	-1	1	-1	1	1	-1	-1	-1	1	-1	-1	-1	1	1	1
1	1	-1	1	1	-1	1	-1	1	-1	1	-1	1	1	1	-1	-1
-1	-1	-1	-1	-1	-1	1	-1	1	1	-1	-1	1	-1	-1	1	1
-1	-1	1	1	1	1	-1	-1	1	-1	1	1	1	-1	1	-1	1
-1	1	-1	1	-1	1	1	-1	-1	1	-1	1	-1	-1	1	-1	1

```
>> Pi = 2*randi(2,m,d)-3;
>> s = (1/sqrt(m))*Pi*q;
```

Often called “random projections”. Why?

RANDOMIZED JL CONSTRUCTIONS



S = m dimensional subspace

$s_i = \Pi q_i$ for all i.

$Var[X] = \mathbb{E}[X^2] - \cancel{\mathbb{E}[X]^2}$ 0

When is a map $P : \mathbb{R}^d \to \mathbb{R}^m$ a projection operator? $\to$ maps each q_i to closest point in S.

$$\underbrace{\boxed{\;P\;}}_{m \times d}$$

PP^T = Identity

Is Π a projection operator? variance

No. $\mathbb{E}[\Pi\Pi^T] = dI$

$$\mathbb{E}\left[[\Pi\Pi^T]_{ij}\right] = \mathbb{E}\left[\sum_{\ell=1}^{d} \Pi_{i\ell}\Pi^T_{\ell j}\right] = \sum_{\ell=1}^{d} \mathbb{E}\left[\Pi_{i\ell}\Pi^T_{\ell j}\right] \quad . \quad d$$

$i \neq j = 0$

$j \neq i$ $\quad \mathbb{E}\left[[\Pi\Pi^T]_{ij}\right] = 0.$ $\quad i = j$ $\quad \sum_{\ell=1}^{d} \underbrace{\mathbb{E}\left[\Pi_{i\ell} \cdot \Pi^T_{i\ell}\right]}_{1 \,=\, Var[N(0,1)]}$ $\to$ normal

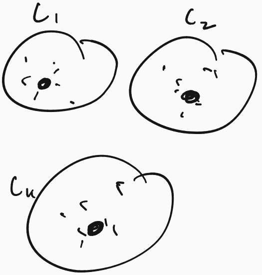
k-means objective: Find clusters $C_1, \ldots, C_k \subseteq \{1, \ldots, n\}$ to minimize:

$X_1, \ldots, X_n \in \mathbb{R}^d$

$$Cost(C_1, \ldots, C_k) = \sum_{j=1}^{k} \frac{1}{2|C_j|} \sum_{u,v \in C_j} \|\mathbf{x}_u - \mathbf{x}_v\|_2^2.$$

$C_1 \cup \ldots \cup C_k = \{1, \ldots, n\}$

$C_i \cap C_j = \emptyset$ for all $i, j \in 1, \ldots, k$

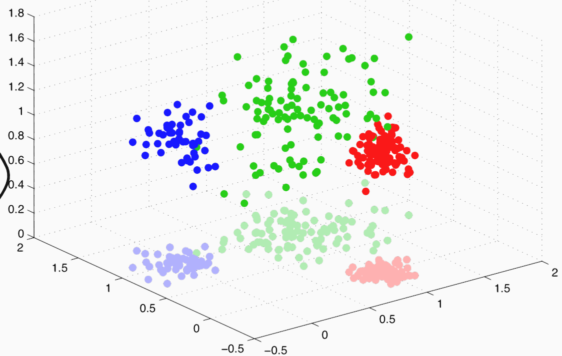


Approximation algorithm: Find optimal clusters $\tilde{C}_1^*, \ldots, \tilde{C}_k^*$ for the k dimension data set $\mathbf{\Pi X}_1, \ldots, \mathbf{\Pi X}_n$.

Want to prove:

$$\text{Cost}(\tilde{C}_1^*, \ldots, \tilde{C}_k^*) \leq (1+\epsilon) \min_{C_1,\ldots,C_k} \text{Cost}(C_1, \ldots, C_k)$$

$k = O(\log n / \epsilon^2)$



JL Lemma : For all u, v, $(1-\epsilon)\|X_u - X_v\|_2^2 \leq \|\Pi X_u - \Pi X_v\|_2^2 \leq (1+\epsilon)\|X_u - X_v\|_2^2$

$$Cost(C_1, \ldots, C_k) = \sum_{j=1}^{k} \frac{1}{2|C_j|} \sum_{u,v \in C_j} \|\mathbf{x}_u - \mathbf{x}_v\|_2^2.$$

$$\widetilde{Cost}(C_1, \ldots, C_k) = \sum_{j=1}^{k} \frac{1}{2|C_j|} \sum_{u,v \in C_j} \|\Pi \mathbf{x}_u - \Pi \mathbf{x}_v\|_2^2.$$

$$\leq \sum_{j=1}^{k} \frac{1}{2|C_j|} \sum_{u,v \in C_j} (1+\epsilon)\|X_u - X_v\|_2^2 \rightarrow \text{by JL}$$

$$= (1+\epsilon) \sum_{j=1}^{k} \frac{1}{2|C_j|} \sum_{u,v \in C_j} \|X_u - X_v\|_2^2 = (1+\epsilon)\, Cost(C_1, \ldots, C_k)$$

For any $C_1, \ldots, C_k$,

$$(1-\epsilon)Cost(C_1, \ldots, C_k) \leq \underbrace{\widetilde{Cost}(C_1, \ldots, C_k) \leq (1+\epsilon)Cost(C_1, \ldots, C_k)}$$

We prove this

Proving left hand side is similar.

Let $C_1^*, \ldots, C_k^* = \arg\min Cost(C_1, \ldots, C_k)$ and
$\tilde{C}_1^*, \ldots, \tilde{C}_k^* = \arg\min \widetilde{Cost}(C_1, \ldots, C_k)$

Want to prove: $\underline{Cost(\tilde{C}_1^*, \ldots, \tilde{C}_k^*)} \leq (1 + O(\epsilon))Cost(C_1^*, \ldots, C_k^*)$

John Shin's comment:
$1/\varepsilon^2$ dependence is really bad! $\varepsilon = .01 \to 1/\varepsilon^2 = 10{,}000$.

$$Cost(\tilde{C}_1^*, \ldots, \tilde{C}_k^*) \leq \frac{1}{1-\varepsilon}\underline{\widetilde{Cost}(\tilde{C}_1^*, \ldots, \tilde{C}_k^*)}$$

$$\leq \frac{1}{1-\varepsilon}\underline{\widetilde{Cost}(C_1^*, \ldots, C_k^*)}$$

$$\leq \frac{(1+\varepsilon)}{(1-\varepsilon)}\underline{Cost(C_1^*, \ldots, C_k^*)}$$

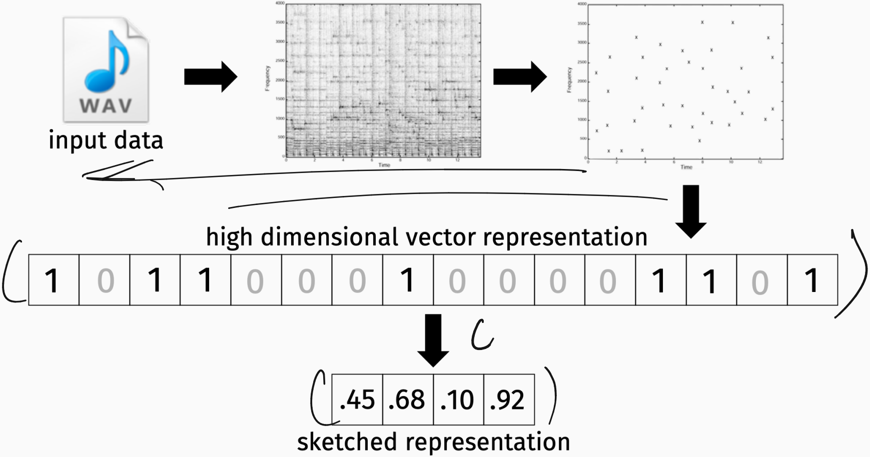
$$\leq (1+4\varepsilon)\, Cost(C_1^*, \ldots, C_k^*)$$

$\approx (1+\varepsilon)(1+O(\varepsilon))$
$\approx (1+O(\varepsilon))$

for any $\varepsilon < 1/2$

- this is a weakness of JL. Not always good for highly accurate approximations
- in some applications (including kmeans) JL tends to do better than the theory predicts.

If wanted $\leq (1+\varepsilon)$ just set $k = O\left(\frac{\log n}{(\varepsilon/4)^2}\right)$ to begin with.



Goal: Given input vectors **q** and **y**, $C(\mathbf{q})$ and $C(\mathbf{y})$ should be similar if **q** and **y** are similar.

$$\begin{array}{lcccccc} q: & 0 & 0 & 1 & 1 & 0 & 1 \\ y: & 0 & 1 & 1 & 0 & 0 & 1 \end{array}$$

Other Example: Binary valued vectors.

Definition (Jaccard Similarity)

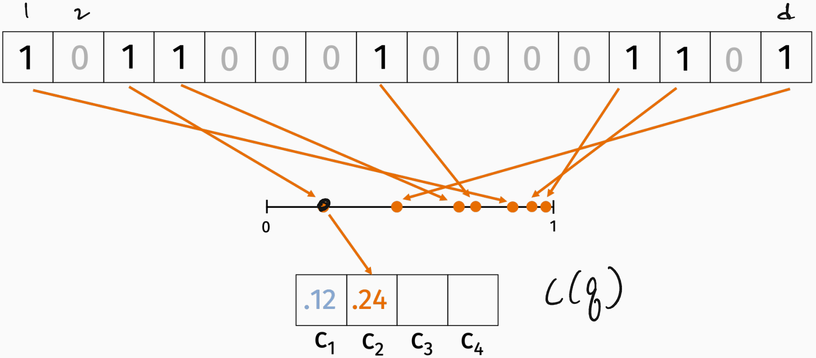
$$J(\mathbf{q}, \mathbf{y}) = \frac{|\mathbf{q} \cap \mathbf{y}|}{|\mathbf{q} \cup \mathbf{y}|} = \frac{\text{\# of non-zero entries in common}}{\text{total \# of non-zero entries}} \quad \frac{2}{4} = \frac{1}{2}$$

$0 \leq J(\mathbf{q}, \mathbf{y}) \leq 1.$

for example above

MINHASH

- Choose k random hash functions $h_1, \ldots, h_k : \{1, \ldots, d\} \rightarrow [0, 1]$.
- For $i \in 1, \ldots, k$, let $c_i = \min_{j, \mathbf{q}_j = 1} h_i(j)$.
- $C(\mathbf{q}) = [c_1, \ldots, c_k]$.



$C(q)$

$\min(h_2(1), h_2(3), h_2(4), h_2(8) \ldots) = .24$

Actually, we proved: $\Pr[c_i(\mathbf{q}) = c_i(\mathbf{y})] = v$ for all i.

Example 1: Binary valued vectors.

If $J(\mathbf{q}, \mathbf{y}) = v$ then the expected number of common entries between $C(\mathbf{q})$ and $C(\mathbf{y})$ is v.

C(q)	.12	.24	.76	.35
C(y)	**.12**	**.98**	**.76**	**.11**

Using a Chernoff bound, we proved that if C maps to dimension $O\left(\frac{\log(1/\delta)}{\epsilon^2}\right)$, we can approximate the Jaccard similarity between any two binary vectors to accuracy ϵ with probability $1-\delta$.

$J(y, q) \geq .9$

Common goal: Find all fingerprints in database $\mathbf{q}_1, \ldots, \mathbf{q}_n \in \mathbb{R}^d$ that are close to some input finger print $\mathbf{y} \in \mathbb{R}^d$.

- Audio + video search. $n = 10$ million
- Finding duplicate or near duplicate documents.
- Seismic applications (here they want all pairs of close fingerprints). $n =$ millions

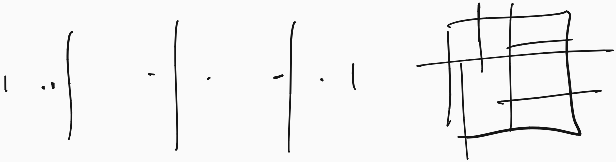
sketch dimension

Does similarity sketching help in these applications?

$m \ll d$

$O(nd)$ time $\qquad\qquad$ $O(nm)$

$O(nd)$ space $\quad$ improvement $\longrightarrow$ $\quad$ $O(nm)$



Kd trees Quadtrees

$O(2^d)$

$2^{O\left(\frac{\log(n)}{\epsilon^2}\right)}$

New goal: Sublinear $o(n)$ time to find near neighbors.

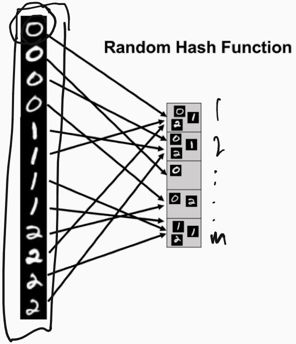
Can we also improve the n dependence?
More important in many applications.

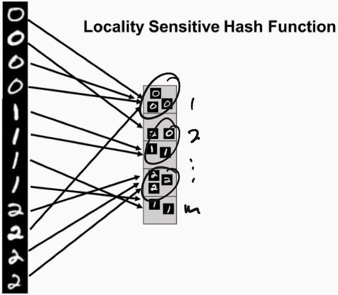
– possible in low-dimensions using
Kd-trees, quad trees, etc

Let $h : \mathbb{R}^d \to \{1, \ldots, m\}$ be a random hash function.

We call h locality sensitive if $\Pr[h(\mathbf{q}) == h(\mathbf{y})]$ is:

- Higher when $\mathbf{q}$ and $\mathbf{y}$ are more similar.
- Lower when $\mathbf{q}$ and $\mathbf{y}$ differ substantially.





LSH for Jaccard similarity:

- Let $c : \{0,1\}^d \to [0,1]$ be a single instantiation of MinHash.
- Let $g : [0,1] \to \{1,\ldots,m\}$ be a fully random hash function.
- Let $h(\mathbf{x}) = g(c(\mathbf{x}))$.

LSH for Jaccard similarity:

- Let $c : \{0,1\}^d \to [0,1]$ be a single instantiation of MinHash. uniformly
- Let $g : [0,1] \to \{1,\ldots,m\}$ be a fully random hash function.
- Let $h(\mathbf{x}) = g(c(\mathbf{x}))$.

$g(c(q)) = g(c(y))$
when $c(q) = c(y)$

If $J(\mathbf{q},\mathbf{y}) = v$,

Case 1: $c(q) = c(y)$
(happens with prob. v)
$\Pr[h(q) = h(y)] = 1$.

Case 2: $c(q) \neq c(y)$
(happens with prob. $1-v$)
$\Pr[h(q) = h(y)] = 1/m$ why?

$$\Pr[h(\mathbf{q}) == h(\mathbf{y})] = v \cdot 1 + (1-v) \cdot \frac{1}{m}$$

$$= \boxed{v + \frac{(1-v)}{m}}$$

usually very small (think of $m = O(n)$)

$h(q) = g(\alpha)$
$h(y) = g(\beta)$ where $\alpha \neq \beta$.
Since g is uniformly random, $\Pr[g(\alpha) = g(\beta)] = 1/m$

Basic approach for near neighbor search in a database.

Pre-processing:

- Select random LSH function $h : \underbrace{\{0,1\}^d}_{} \to \underline{1, \ldots, m.}$
- Create table T with m slots. $m = O(n)$ (we won't discuss choice of m rigorously)
- For $i = 1, \ldots, n$, insert $\mathbf{q}_i$ into $T(h(\mathbf{q}_i))$.

for Jaccard similarity

$$h = g(c(q_i))$$

uniform random hash   MinHash

Basic approach for near neighbor search in a database.

Pre-processing:

- Select random LSH function $h : \{0,1\}^d \rightarrow 1, \ldots, m$.
- Create table T with m slots.
- For $i = 1, \ldots, n$, insert $\mathbf{q}_i$ into $T(h(\mathbf{q}_i))$.

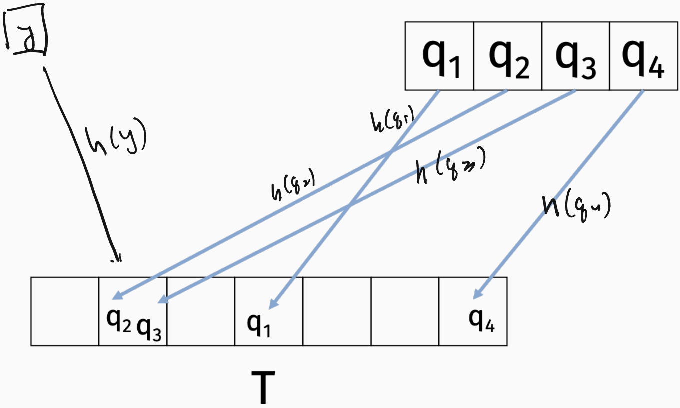
$J(q, y)$ is large
$\downarrow$
$\Pr[h(y) == h(q)]$ large.

Query:

- Want to find near neighbors of input $\mathbf{y} \in \{0,1\}^d$.
- Linear scan through all vectors in $T(h(\mathbf{y}))$.

to find best matches

only 1/m buckets → hopefully << n elements to go through


y
h(y)
q1
q2
q3
q4
h(q1)
h(q2)
h(q3)
h(q4)
q2 q3
q1
q4
T

Two main considerations:

- False Negative Rate: What's the probability we do not find a vector that is close to $\mathbf{y}$?
- False Positive Rate: What's the probability we need to scan over vectors that are not close to $\mathbf{y}$?

False Negative Rate

Suppose the nearest database point **q** has $J(\mathbf{y}, \mathbf{q}) = .4$.

What's the probability we do not find q?

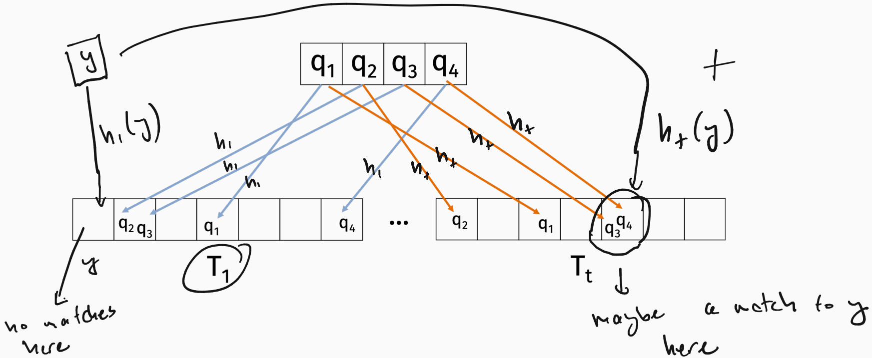
$\Pr[\text{find } q] \approx .4$

$\Pr[\text{don't find } q] = 1 - .4 = .6$

60%

thats terrible...

REDUCING FALSE NEGATIVE RATE



Pre-processing:

- Select t independent LSH's $h_1, \ldots, h_t : \{0,1\}^d \rightarrow 1, \ldots, m$.
- Create tables $T_1, \ldots, T_t$, each with m slots.
- For $i = 1, \ldots, n$, $\underbrace{j = 1, \ldots, t}$, insert $\mathbf{q}_i$ into $T_j(h_j(\mathbf{q}_i))$.

Query:

- Want to find near neighbors of input $\mathbf{y} \in \{0,1\}^d$.
- Linear scan through all vectors in $T_1(h_1(\mathbf{y})), T_2(h_2(\mathbf{y})), \ldots, T_t(h_t(\mathbf{y}))$.

$t = 10$

99%

t repetitions

Query:

- Want to find near neighbors of input $\mathbf{y} \in \{0,1\}^d$.
- Linear scan through all vectors in $T_1(h_1(\mathbf{y})), T_2(h_2(\mathbf{y})), \ldots, T_t(h_t(\mathbf{y}))$.

Suppose the nearest database point $\mathbf{q}$ has $J(\mathbf{y}, \mathbf{q}) = .4$.

What's the probability we find q?

$\Pr\{\text{don't find } q?\} = \Pr[\text{don't find } q \text{ in Table 1 } \underline{\underline{\text{and}}}$

$\text{don't find } q \text{ in Table 2 } \underline{\underline{\text{and}}}$

$\vdots$

$\text{don't find } q \text{ in Table } T]$

$= \underline{\underline{(1-.4)^t}}$

$\Pr[\text{find } q] = 1-(1-.4)^t.$

$\boxed{\text{If } t=10: \quad 1-(1.4)^{10} \geq .99}$

$t = 10$

89%

Suppose there is some other database point $\mathbf{q}_j$ with $\boxed{J(\mathbf{y}, \mathbf{q}_j) = .2?}$ What is the probability we will consider that point in our original scheme?

$\Pr[h(y) = h(q)] \approx .2$ when $J(q, y) = .2$

why approx?

In the new scheme?

$$1 - (1 - .2)^{10} = 89\%$$

remember

$$\Pr[h(y) = h(q)] = .2 + \underbrace{\frac{(1-.2)}{m}}_{\approx 0}$$

h is uniformly random $g([.1 \; .6 \; .2 \; .98]) \to \{1, \ldots, m\}$

↓

$Pr(h(I) = h(Z)) = 1/m$ when $I \neq Z$.

Change our locality sensitive hash function.

$h(q) : g(.1, .2, .7)$

$h(y) = g(.1, .9, .7)$

Tunable LSH for Jaccard similarity:

- Choose parameter $s \in \mathbb{Z}^+$. $= \{1, 2, \ldots\}$ positive integers
- Let $c_1, \ldots, c_s : \{0,1\}^d \to [0,1]$ be random MinHashs.
- Let $g : [0,1]^s \to \{1, \ldots, m\}$ be a fully random hash function.
- Let $h(\mathbf{x}) = g(c_1(\mathbf{x}), \ldots, c_s(\mathbf{x}))$.

If $J(\mathbf{q}, \mathbf{y}) = v$,

Case 1: $c_1(q) = c_1(y), \ldots, c_s(q) = c_s(y)$

Happens with prob. v^s

$Pr[h(q) = h(y)] = 1$.

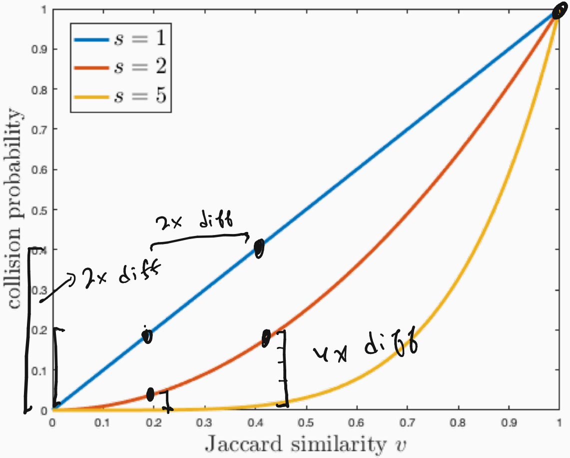
will depend on s

$$\Pr[h(\mathbf{q}) == h(\mathbf{y})] = v^s \cdot 1 + \underbrace{(1 - v^s) \cdot 1/m}_{\approx 0}$$

$\approx v^s$

Case 2: $c_i(q) \neq c_i(y)$ for some i:

Happens w/ prob $(1 - v^s)$ $Pr[h(q) = h(y)] = 1/m$


$s = 1$
$s = 2$
$s = 5$
collision probability
Jaccard similarity v
0
0.1
0.2
0.3
0.4
0.5
0.6
0.7
0.8
0.9
1
2x diff
2x diff
4x diff

$t=3$

78%

Parameter: $S = 1$.

Chance we find $\mathbf{q}_i$ with $J(\mathbf{y}, \mathbf{q}_i) = .8$: $.8$

$$1-(1-.8)^t$$

If $t=3 \rightarrow 1-(1-.8)^3 \approx 99\%$

Chance we need to scan $\mathbf{q}_j$ with $J(\mathbf{y}, \mathbf{q}_j) = .4$:

$$1-(1-.4)^3 \approx 78\%$$

$t = 5$

58%

Parameter: $S = 2$.

Chance we find $\mathbf{q}_i$ with $J(\mathbf{y}, \mathbf{q}_i) = .8$:

$$1 - (1 - .8^2)^t$$

If $\underline{t = 5} \longrightarrow 1 - (1 - .8^2)^5 \geq 99\%$

Chance we need to scan $\mathbf{q}_j$ with $J(\mathbf{y}, \mathbf{q}_j) = .4$:

$$1 - (1 - .4^2)^5 \approx 58\%$$

$t=12$
12%

Parameter: $S = 5$.

Chance we find $\mathbf{q}_i$ with $J(\mathbf{y}, \mathbf{q}_i) = .8$:

$$1-(1-.8^5)^t$$

If $\underline{t=12} \rightarrow 1-(1-.8^5)^{12} \geq 99\%$

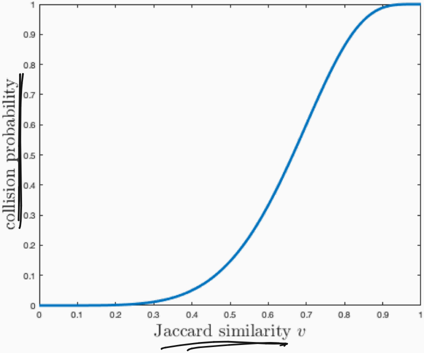
Chance we need to scan $\mathbf{q}_j$ with $J(\mathbf{y}, \mathbf{q}_j) = .4$:

$$1-(1-.4^5)^{12} \approx \underline{12\%}$$

reducing false positive rate a lot!

Probability we see $\mathbf{q}$ when querying $\mathbf{y}$ if $J(\mathbf{q}, \mathbf{y}) = v$:

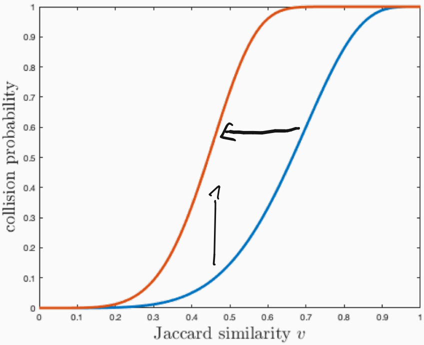
$$\left(1 - (1 - v^s)^t\right)$$



$$s = 5, t = 5$$

Probability we see $\mathbf{q}$ when querying $\mathbf{y}$ if $J(\mathbf{q}, \mathbf{y}) = v$:

$$1 - (1 - v^s)^t$$

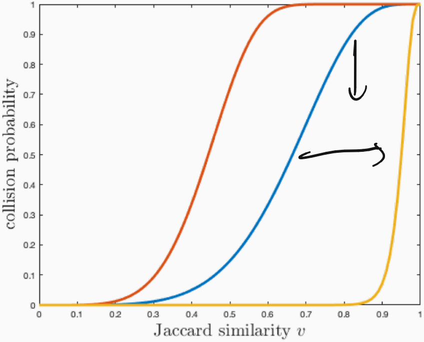


$$s = 5, t = 40$$

Increase T → curve moves left

Probability we see $\mathbf{q}$ when querying $\mathbf{y}$ if $J(\mathbf{q},\mathbf{y}) = v$:

$$\approx 1 - (1 - v^s)^t$$



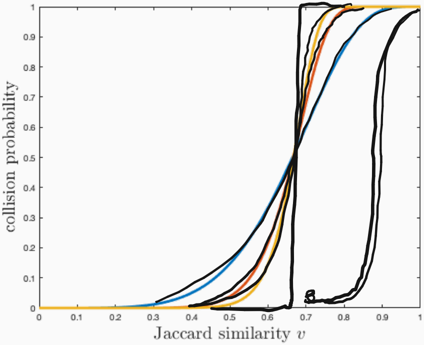
$s = 40, t = 5$

Increase s → curve moves right

s-CURVE TUNING

Probability we see **q** when querying **y** if $J(\mathbf{q}, \mathbf{y}) = v$:

$$1 - (1 - v^s)^t$$



Increasing both s and t gives a steeper curve. (in some place)

Better for search, but worse space complexity.

Use Case 1: Fixed threshold.

- Shazam wants to find match to audio clip $\mathbf{y}$ in a database of 10 million clips.
- There are 10 true matches with $J(\mathbf{y}, \mathbf{q}) > .9$.
- There are 10,000 near matches with $J(\mathbf{y}, \mathbf{q}) \in [.7, .9]$.

With $s = 25$ and $t = 40$,

- Hit probability for $J(\mathbf{y}, \mathbf{q}) > .9$ is $\gtrsim 1 - (1 - .9^{25})^{40} = .95$
- Hit probability for $J(\mathbf{y}, \mathbf{q}) \in [.7, .9]$ is $\lesssim 1 - (1 - .9^{25})^{40} = .95$
- Hit probability for $J(\mathbf{y}, \mathbf{q}) < .7$ is $\lesssim 1 - (1 - .7^{25})^{40} = .005$

$\mathbb{E}[$**Total number of items scanned:**$]$ → using linearity of expectation

$$.95 \cdot 10 + .95 \cdot 10,000 + .005 \cdot 9,989,990 \approx 60,000 \ll 10,000,000.$$

traded faster runtime

for more space

Space complexity: 40 hash tables $\approx 40 \cdot O(n)$.

Directly trade space for fast search.

Naive solution: $40 \cdot n \cdot d$ space

Storing pointers to data vectors: $40 \cdot n \cdot O(1) + nd$ space

Concrete worst case result: (no data assumptions like Shazam case study)

Theorem (Indyk, Motwani, 1998)

If there exists some q with $\|\mathbf{q} - \mathbf{y}\|_0 \leq R$, return a vector $\tilde{\mathbf{q}}$ with $\|\tilde{\mathbf{q}} - \mathbf{y}\|_0 \leq C \cdot R$ in:

If $C = 2$,

- *Time:* $O\left(n^{1/C}\right)$. $\rightarrow O(\sqrt{n})$
- *Space:* $O\left(n^{1+1/C}\right)$. $\rightarrow O(n^{3/2})$

$\|\mathbf{q} - \mathbf{y}\|_0 =$ "hamming distance" = number of elements that differ between $\mathbf{q}$ and $\mathbf{y}$.

Theorem (Indyk, Motwani, 1998)

Let $\mathbf{q}$ *be the closest database vector to* $\mathbf{y}$. *Return a vector* $\tilde{\mathbf{q}}$ *with* $\underline{\|\tilde{\mathbf{q}} - \mathbf{y}\|_0} \leq C \cdot \underline{\|\mathbf{q} - \mathbf{y}\|_0}$ *in:*

No B parameter

- *Time:* $\tilde{O}\left(n^{1/C}\right)$.
- *Space:* $\tilde{O}\left(n^{1+1/C}\right)$.

hides extra log factors

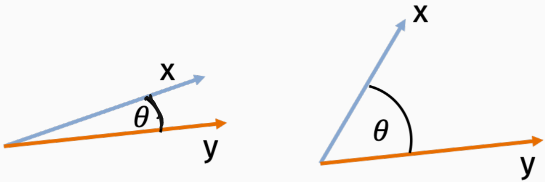
Any ideas for how this is done?

Good locality sensitive hash functions exists for many other similarity measures.

OTHER LSH FUNCTIONS

Good locality sensitive hash functions exists for many other similarity measures.

Cosine similarity $\cos(\theta(\mathbf{x},\mathbf{y})) = \frac{\langle \mathbf{x},\mathbf{y}\rangle}{\|\mathbf{x}\|_2\|\mathbf{y}\|_2}$: $= x^T y$



$$-1 \leq \cos(\theta(\mathbf{x},\mathbf{y})) \leq 1.$$

$\theta = \pi$ $\qquad\qquad$ $\theta = 0$

Cosine similarity is natural "inverse" for Euclidean distance.

Euclidean distance $\|\mathbf{x} - \mathbf{y}\|_2^2$:

- Suppose for simplicity that $\|\mathbf{x}\|_2^2 = \|\mathbf{y}\|_2^2 = 1$.

$$\|x - y\|_2^2 = \underbrace{\|x\|_2^2}_{=1} + \underbrace{\|y\|_2^2}_{=1} - 2x^T y$$

$$= 2(\underbrace{1 - x^T y}_{= 1 - \text{cosine similarity}})$$

SIMHASH

$\begin{bmatrix} .7 \\ 1.2 \\ \vdots \end{bmatrix}$

Locality sensitive hash for cosine similarity:

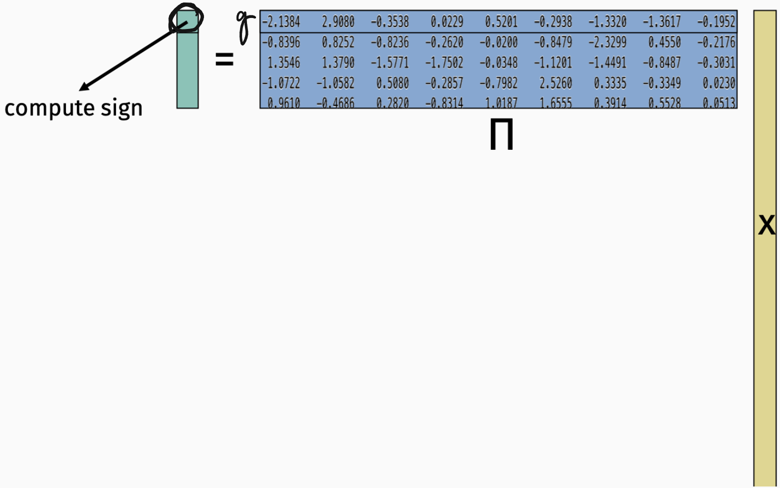
- Let $\mathbf{g} \in \mathbb{R}^d$ be randomly chosen with each entry $\mathcal{N}(0, 1)$.
- $h : \mathbb{R}^d \to \{-1, 1\}$ is defined $h(\mathbf{x}) = \underbrace{\text{sign}(\langle \mathbf{g}, \mathbf{x} \rangle)}$. $\quad$ $\text{sign}(g^T x)$

If $\cos(\theta(\mathbf{x}, \mathbf{y})) = v$, what is $\Pr[h(\mathbf{x}) == h(\mathbf{y})]$?

$\cos(\theta(x, y)) = -1 \quad \Pr[h(x) = h(y)] = 0.$

$\cos(\theta(x, y)) = 1 \quad \Pr[h(x) = h(y)] = 1.$

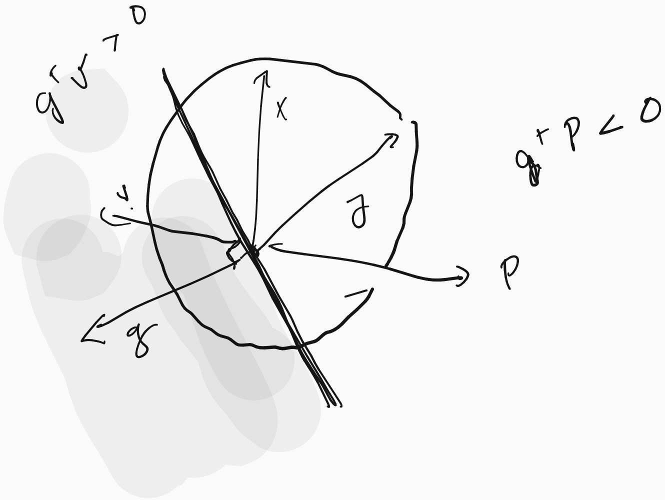
Inspired by Johnson-Lindenstrauss sketching



Locality sensitive hash for cosine similarity:

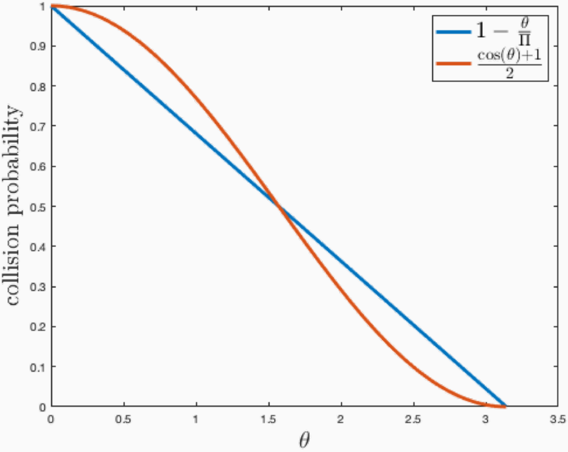
- Let $\mathbf{g} \in \mathbb{R}^d$ be randomly chosen with each entry $\mathcal{N}(0, 1)$.
- $h : \mathbb{R}^d \to \{-1, 1\}$ is definied $h(\mathbf{x}) = \text{sign}(\langle \mathbf{g}, \mathbf{x} \rangle)$.

If $\cos(\theta(\mathbf{x}, \mathbf{y})) = v$, what is $\Pr[h(\mathbf{x} == h(\mathbf{y})]$?

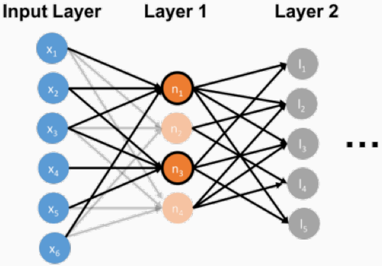


$$1 - \frac{\theta(x,y)}{\pi}$$

SIMHASH ANALYSIS



Work of Anshumali Shrivastava at Rice University and coauthors.

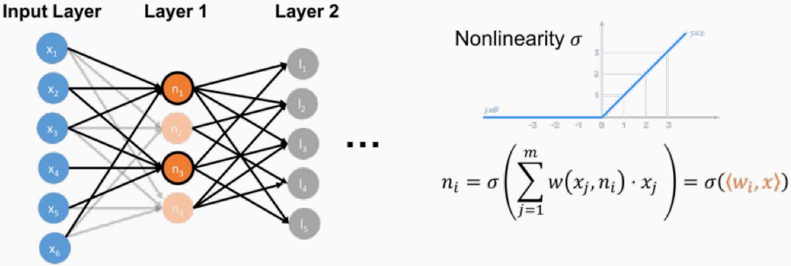


$$n_i = \sigma\left(\sum_{j=1}^{m} w(x_j, n_i) \cdot x_j\right) = \sigma(\langle w_i, x\rangle)$$

- Number of multiplications to evaluate $\mathcal{N}(\mathbf{x})$: $|\mathbf{x}| \cdot |\text{layer } 1| + |\text{layer } 1| \cdot |\text{layer } 2| + |\text{layer } 2| \cdot |\text{layer } 3| + \ldots$.
- For an approximate solution, only consider neurons on each each with high activation.

SIMHASH TO SPEEDUP NEURAL NETWORKS

Work of Anshumali Shrivastava at Rice University and coauthors.



- High activation $=$ large value of $\sigma(\langle \mathbf{w}_i, \mathbf{x}\rangle)$.
- Typically $\sigma(\langle \mathbf{w}_i, \mathbf{x}\rangle)$ increases as $\langle \mathbf{w}_i, \mathbf{x}\rangle$ increases.
- Use LSH/SimHash to quickly find all $\mathbf{w}_i$ for which $\langle \mathbf{w}_i, \mathbf{x}\rangle$ is large and only include these terms in the sum.

Why can't we just sample entries from vectors?

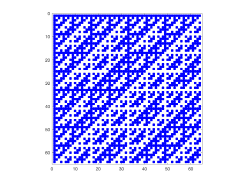
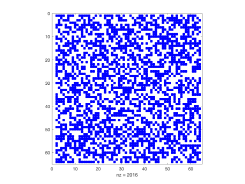
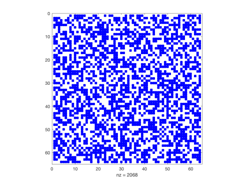
FAST JOHNSON-LINDENSTRAUSS (TIME PERMITTING)

[Ailon, Chazelle, 2009 – The Fast Johnson-Lindenstrauss Transform]

FAST JOHNSON-LINDENSTRAUSS (TIME PERMITTING)

[Ailon, Chazelle, 2009 – The Fast Johnson-Lindenstrauss Transform]

FAST JOHNSON-LINDENSTRAUSS (TIME PERMITTING)

		
Deterministic Hadamard matrix.	Randomized Hadamard *HD*.	Fully random sign matrix.

FAST JOHNSON-LINDENSTRAUSS (TIME PERMITTING)