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

NYU Tandon School of Engineering, Prof. Christopher Musco

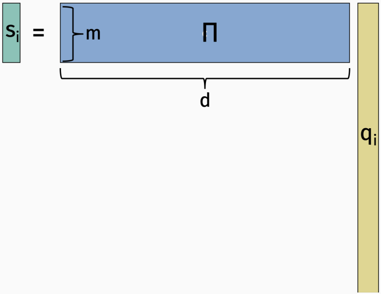
ADMINISTRATIVE

- 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* $\mathbf{\Pi} : \mathbb{R}^d \to \mathbb{R}^m$ *where* $m = O\left(\frac{\log n}{\epsilon^2}\right)$ *such that for all* i, j,

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



$\mathbf{\Pi} \in \mathbb{R}^{k\times d}$ 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

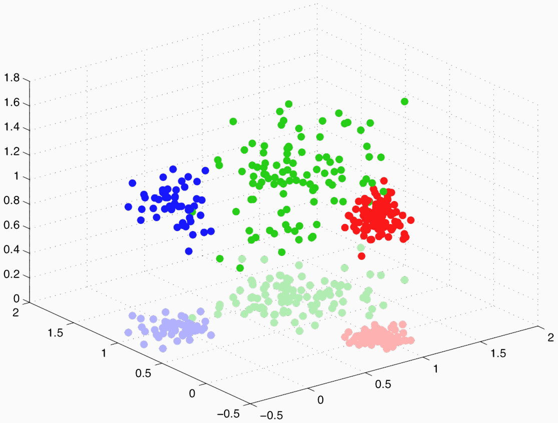
```
>> Pi = 2*randi(2,m,d)-3;
>> s = (1/sqrt(m))*Pi*q;
```

Often called "random projections". Why?

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

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

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



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

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

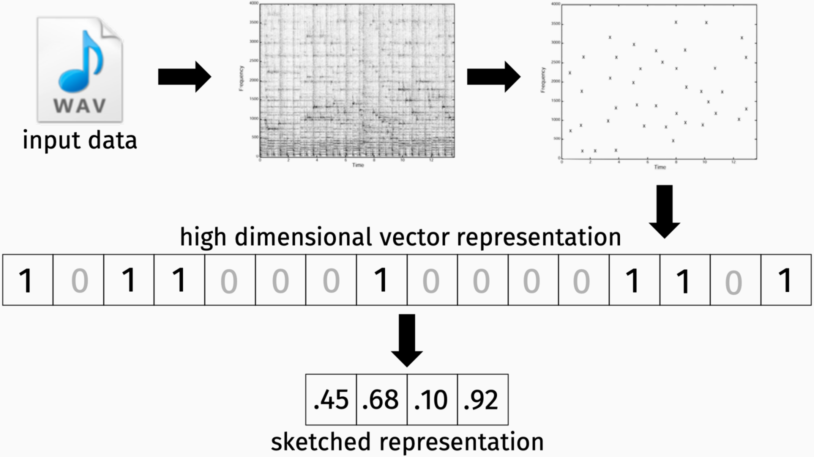
For any $C_1,\ldots,C_k$,

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

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

SIMILARITY SKETCHING



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

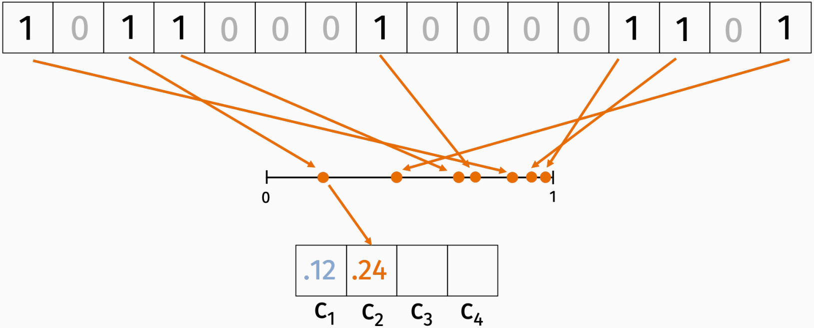
Other Example: Binary valued vectors.

Definition (Jaccard Similarity)

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

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

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



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 $1 - \delta$.

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.
- Finding duplicate or near duplicate documents.
- Seismic applications (here they want all pairs of close fingerprints).

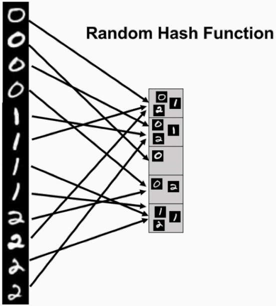
Does similarity sketching help in these applications?

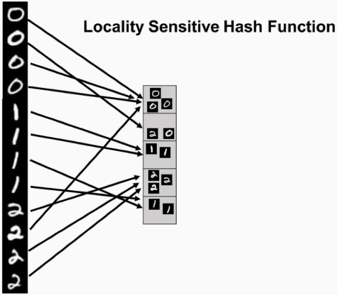
New goal: Sublinear $o(n)$ time to find near neighbors.

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

LSH for Jaccard similarity:

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

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

$$\Pr[h(\mathbf{q}) == h(\mathbf{y})] =$$

Basic approach for near neighbor search in a database.

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 $T(h(\mathbf{q}_i))$.

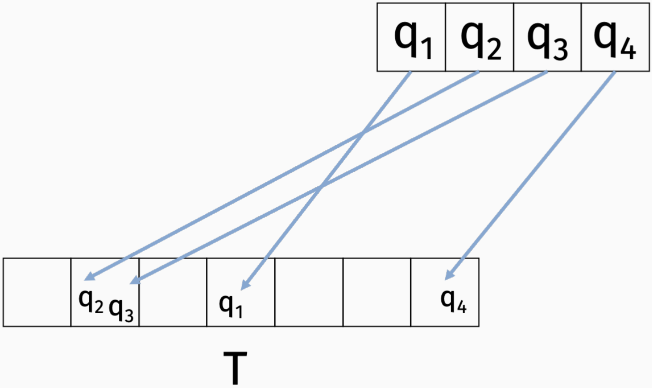
Basic approach for near neighbor search in a database.

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 $T(h(\mathbf{q}_i))$.

Query:

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

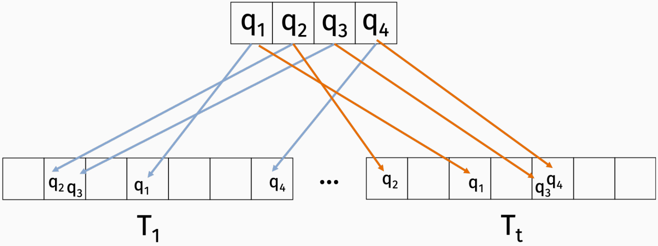

q1 q2 q3 q4
q2 q3
q1
q4
T

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 $\mathbf{y}$?

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

What's the probability we do not find q?



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 $i = 1, \ldots, n$, $j = 1, \ldots, t$, 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 $T_1(h_1(\mathbf{y})), T_2(h_2(\mathbf{y})), \ldots, T_t(h_t(\mathbf{y}))$.

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?

WHAT HAPPENS TO FALSE POSITIVES?

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

In the new scheme?

Change our locality sensitive hash function.

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

$$\Pr[h(\mathbf{q}) == h(\mathbf{y})] =$$

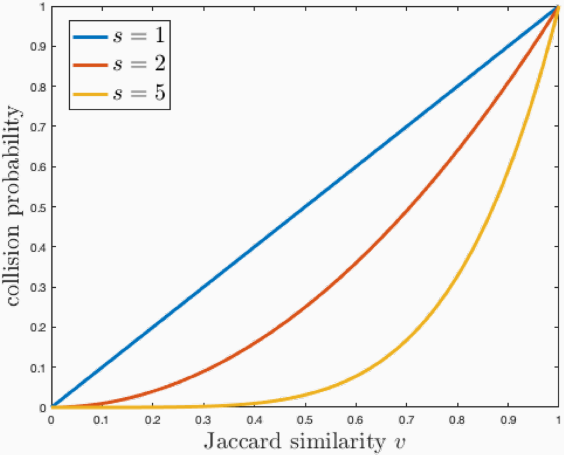
Change our locality sensitive hash function.

Tunable LSH for Jaccard similarity:

- Choose parameter $s \in \mathbb{Z}^+$.
- 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 $J(\mathbf{q}, \mathbf{y}) = v$,

$$\Pr\left[h(\mathbf{q}) == h(\mathbf{y})\right] =$$


$s = 1$
$s = 2$
$s = 5$
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

Parameter: $S = 1$.

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

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

Parameter: $S = 2$.

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

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

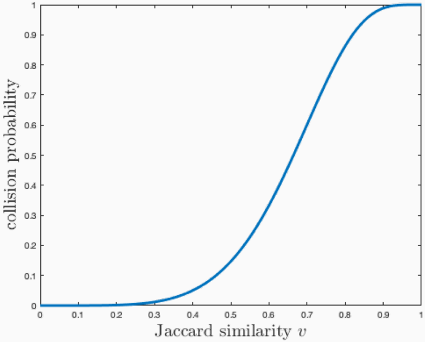
Parameter: $S = 5$.

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

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

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

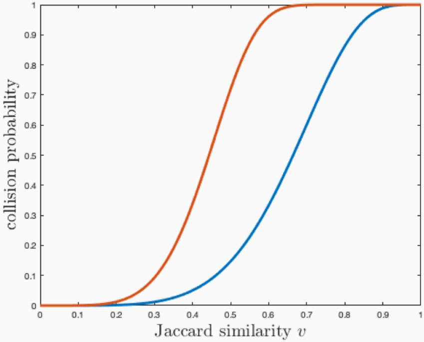
$$1 - (1 - v^s)^t$$



$$s = 5, t = 5$$

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

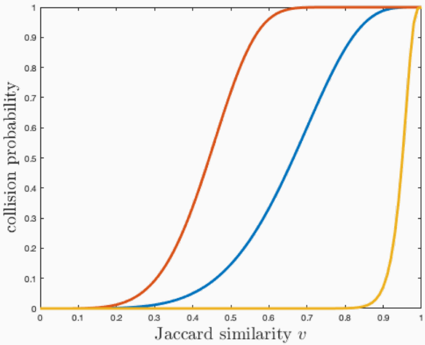
$$1 - (1 - v^s)^t$$



$s = 5, t = 40$

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

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

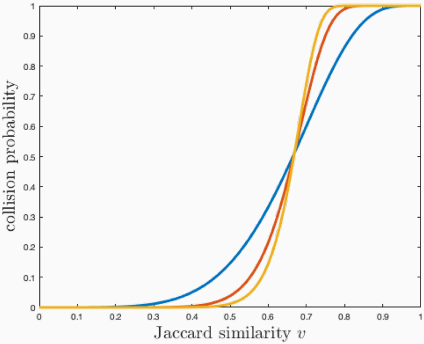


$$s = 40, t = 5$$

s-CURVE TUNING

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

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



Increasing both s and t gives a steeper curve.

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 10,0000 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 $\lesssim 1 - (1 - .7^{25})^{40} = .005$

Total number of items scanned:

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

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

Directly trade space for fast search.

Concrete worst case result:

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:

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

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

Theorem (Indyk, Motwani, 1998)

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

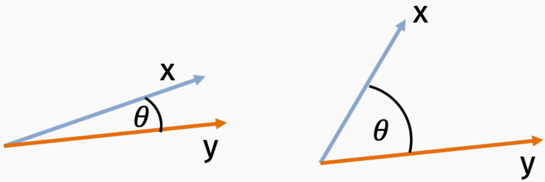
- *Time:* $\tilde{O}\left(n^{1/C}\right)$.
- *Space:* $\tilde{O}\left(n^{1+1/C}\right)$.

Any ideas for how this is done?

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

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

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



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

Cosine similarity is natural "inverse" for Euclidean distance.

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

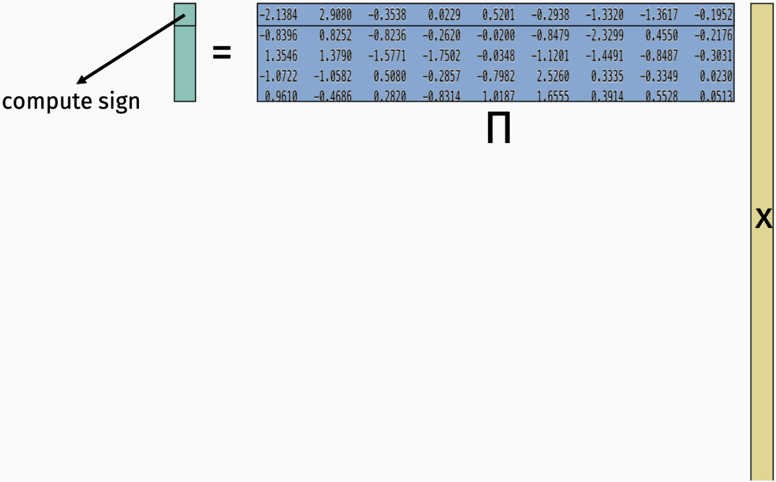
- Suppose for simplicity that $\|\mathbf{x}\|_2^2 = \|\mathbf{y}\|_2^2 = 1$.

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 $\Pr[h(\mathbf{x} == h(\mathbf{y})]$?

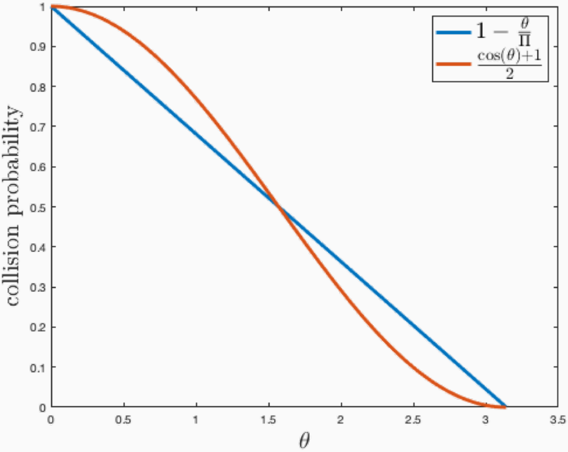
Inspired by Johnson-Lindenstrauss sketching



Locality sensitive hash for cosine similarity:

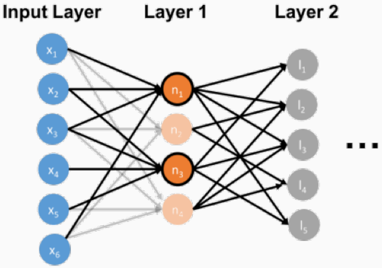
- Let $\mathbf{g} \in \mathbb{R}^d$ be randomly chosen with each entry $\mathcal{N}(0, 1)$.
- $h : \mathbb{R}^d \rightarrow \{-1, 1\}$ is defined $h(\mathbf{x}) = \text{sign}(\langle \mathbf{g}, \mathbf{x} \rangle)$.

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



$1 - \frac{\theta}{\Pi}$
$\frac{\cos(\theta)+1}{2}$
collision probability
θ
0
0.1
0.2
0.3
0.4
0.5
0.6
0.7
0.8
0.9
1
0.5
1.5
2
2.5
3
3.5

Work of Anshumali Shrivastava at Rice University and coauthors.

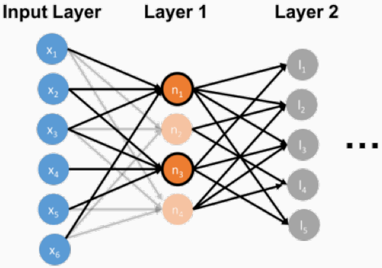


$$n_i = \sigma\left(\sum_{j=1}^{m} w(x_j, n_i)\cdot x_j\right) = \sigma(\langle w_i, x\rangle)$$

- 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.



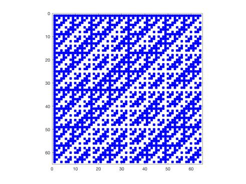
$$n_i = \sigma\left(\sum_{j=1}^{m} w(x_j, n_i) \cdot x_j\right) = \sigma(\langle w_i, x\rangle)$$

- <u>High activation</u> $=$ 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.

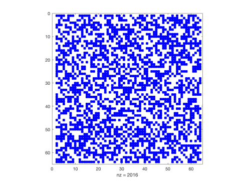
Why can't we just sample entries from vectors?

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

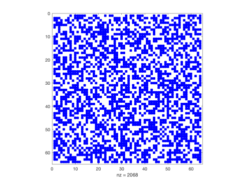
[Ailon, Chazelle, 2009 – The Fast Johnson-Lindenstrauss Transform]



Deterministic Hadamard matrix.



Randomized Hadamard *HD*.



Fully random sign matrix.

fast johnson-lindenstrauss (time permitting)