CS-GY 6763: Lecture 5
Dimensionality reduction, near neighbor search in high dimensions

NYU Tandon School of Engineering, Prof. Christopher Musco

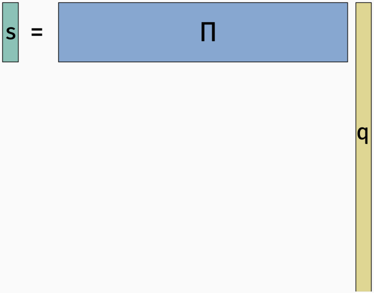
PROJECT

- If you are doing a project, find a partner and sign-up to present for reading group slot **by Monday, 10/9**. We need presenters for next Friday!

Lemma (Johnson-Lindenstrauss, 1984)

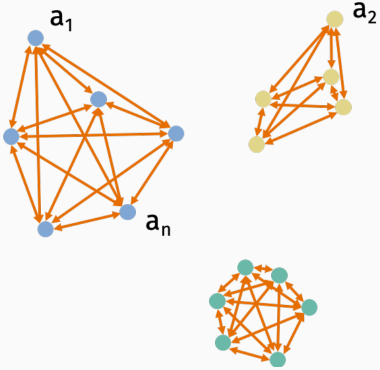
For any set of n data points $\mathbf{q}_1, \ldots, \mathbf{q}_n \in \mathbb{R}^d$ *there exists a* *linear map* $\mathbf{\Pi} : \mathbb{R}^d \to \mathbb{R}^k$ *where* $k = O\left(\frac{\log n}{\epsilon^2}\right)$ *such that* *for all* i, j,

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



k-means clustering: For data set $\mathbf{a}_1, \ldots, \mathbf{a}_n$, find clusters $C_1, \ldots, C_k \subseteq \{1, \ldots, n\}$ to minimize:

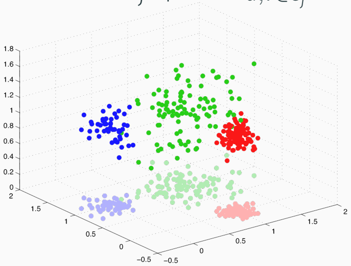
$$Cost(C_1, \ldots, C_k) = \sum_{j=1}^{k} \frac{1}{2|C_j|} \sum_{u,v \in C_j} \|\mathbf{a}_u - \mathbf{a}_v\|_2^2.$$



SAMPLE APPLICATION

k-means clustering: For data set $\mathbf{a}_1, \ldots, \mathbf{a}_n$, find clusters $C_1, \ldots, C_k \subseteq \{1, \ldots, n\}$ to minimize:

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



Claim: If I find the optimal clustering for $\mathbf{\Pi a}_1, \ldots, \mathbf{\Pi a}_n$ then its cost is less than $(1 + \epsilon)$ times the cost of the best clustering obtained with the original data.

$\mathbf{\Pi} \in \mathbb{R}^{k \times d}$ can chosen so that each entry equals $\frac{1}{\sqrt{k}}\mathcal{N}(0,1)$, or each entry equals $\frac{1}{\sqrt{k}} \pm 1$ with equal probability.

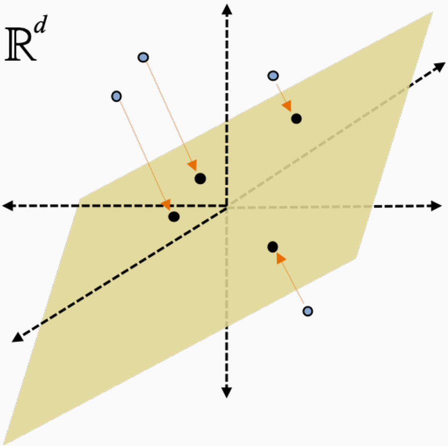
-2.1384	2.9080	-0.3538	0.0229	0.5201	-0.2938	-1.3320	-1.3617	-0.1952
-0.8396	0.8252	-0.8236	-0.2620	-0.0200	-0.8479	-2.3299	0.4550	-0.2176
1.3546	1.3790	-1.5771	-1.7502	-0.0348	-1.1201	-1.4491	-0.8487	-0.3031
-1.0722	-1.0582	0.5080	-0.2857	-0.7982	2.5260	0.3335	-0.3349	0.0230
0.9610	-0.4686	0.2820	-0.8314	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;
```

Lots of other constructions work.



Intuition: Multiplying by a random matrix mimics the process of projecting onto a random k dimensional subspace in d dimensions.

Intermediate result:

Lemma (Distributional JL Lemma)

Let $\mathbf{\Pi} \in \mathbb{R}^{k\times d}$ *be chosen so that each entry equals* $\frac{1}{\sqrt{k}}\mathcal{N}(0,1)$, *where* $\mathcal{N}(0,1)$ *denotes a standard Gaussian random variable.*

If we choose $k = O\left(\frac{\log(1/\delta)}{\epsilon^2}\right)$, *then for any vector* $\mathbf{x}$, *with probability* $(1-\delta)$:

$$(1-\epsilon)\|\mathbf{x}\|_2^2 \leq \|\mathbf{\Pi}\mathbf{x}\|_2^2 \leq (1+\epsilon)\|\mathbf{x}\|_2^2$$

Given this lemma, how do we prove the traditional Johnson-Lindenstrauss lemma?

We have a set of vectors $\mathbf{q}_1, \ldots, \mathbf{q}_n$. Fix $i, j \in 1, \ldots, n$.

Let $\mathbf{x} = \mathbf{q}_i - \mathbf{q}_j$. By linearity, $\mathbf{\Pi x} = \mathbf{\Pi}(\mathbf{q}_i - \mathbf{q}_j) = \mathbf{\Pi q}_i - \mathbf{\Pi q}_j$.

By the Distributional JL Lemma, with probability $1 - \delta$,

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

Finally, set $\delta = \frac{1}{n^2}$. Since there are $< n^2$ total i, j pairs, by a union bound we have that with probability 9/10, the above will hold <u>for all</u> i, j, as long as we compress to:

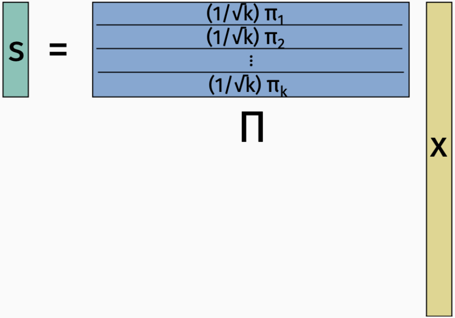
$$k = O\left(\frac{\log(1/(1/n^2))}{\epsilon^2}\right) = O\left(\frac{\log n}{\epsilon^2}\right) \text{ dimensions.} \quad \square$$

Want to argue that, with probability $(1-\delta)$,

$$(1-\epsilon)\|\mathbf{x}\|_2^2 \leq |\mathbf{\Pi x}\|_2^2 \leq (1+\epsilon)\|\mathbf{x}\|_2^2$$

Claim: $\mathbb{E}\|\mathbf{\Pi x}\|_2^2 = \|\mathbf{x}\|_2^2$.

Some notation:



So each $\boldsymbol{\pi}_i$ contains $\mathcal{N}(0,1)$ entries.

Intermediate Claim: Let $\boldsymbol{\pi}$ be a length d vector with $\mathcal{N}(0,1)$ entries.

$$\mathbb{E}\left[\|\boldsymbol{\Pi}\mathbf{x}\|_2^2\right] = \mathbb{E}\left[\left(\langle \boldsymbol{\pi}, \mathbf{x}\rangle\right)^2\right].$$

Goal: Prove $\mathbb{E}\|\boldsymbol{\Pi}\mathbf{x}\|_2^2 = \|\mathbf{x}\|_2^2$.

$$\langle \boldsymbol{\pi}, \mathbf{x} \rangle = Z_1 \cdot \mathbf{x}[1] + Z_2 \cdot \mathbf{x}[2] + \ldots + Z_d \cdot \mathbf{x}[d]$$

where each $Z_1, \ldots, Z_d$ is a standard normal $\mathcal{N}(0, 1)$.

We have that $Z_i \cdot \mathbf{x}[i]$ is a normal $\mathcal{N}(0, \mathbf{x}[i]^2)$ random variable.

Goal: Prove $\mathbb{E}\|\boldsymbol{\Pi}\mathbf{x}\|_2^2 = \|\mathbf{x}\|_2^2$. Established: $\mathbb{E}\|\boldsymbol{\Pi}\mathbf{x}\|_2^2 = \mathbb{E}\left[(\langle \boldsymbol{\pi}, \mathbf{x} \rangle)^2\right]$

What type of random variable is $\langle \boldsymbol{\pi}, \mathbf{x} \rangle$?

Fact (Stability of Gaussian random variables)

$$\mathcal{N}(\mu_1, \sigma_1^2) + \mathcal{N}(\mu_2, \sigma_2^2) = \mathcal{N}(\mu_1 + \mu_2, \sigma_1^2 + \sigma_2^2)$$

$$\begin{aligned}\langle \boldsymbol{\pi}, \mathbf{x} \rangle &= \mathcal{N}(0, \mathbf{x}[1]^2) + \mathcal{N}(0, \mathbf{x}[2]^2) + \ldots + \mathcal{N}(0, \mathbf{x}[d]^2) \\ &= \mathcal{N}(0, \|\mathbf{x}\|_2^2).\end{aligned}$$

So $\mathbb{E}\|\boldsymbol{\Pi}\mathbf{x}\|_2^2 = \mathbb{E}\left[(\langle \boldsymbol{\pi}, \mathbf{x} \rangle)^2\right] = \mathbb{E}\left[\mathcal{N}(0, \|\mathbf{x}\|_2^2)\right] = \|\mathbf{x}\|_2^2$, as desired.

Want to argue that, with probability $(1-\delta)$,

$$(1-\epsilon)\|\mathbf{x}\|_2^2 \leq \|\mathbf{\Pi}\mathbf{x}\|_2^2 \leq (1+\epsilon)\|\mathbf{x}\|_2^2$$

1. $\mathbb{E}\|\mathbf{\Pi}\mathbf{x}\|_2^2 = \|\mathbf{x}\|_2^2$.
2. Need to use a concentration bound.

$$\|\mathbf{\Pi}\mathbf{x}\|_2^2 = \frac{1}{k}\sum_{i=1}^{k}\left(\langle \boldsymbol{\pi}_i, \mathbf{x}\rangle\right)^2 = \frac{1}{k}\sum_{i=1}^{k}\mathcal{N}(0, \|\mathbf{x}\|_2^2)$$

"Chi-squared random variable with k degrees of freedom."

Lemma

Let Z be a Chi-squared random variable with k degrees of freedom.

$$\Pr[|\mathbb{E}Z - Z| \geq \epsilon \mathbb{E}Z] \leq 2e^{-k\epsilon^2/8}$$

Goal: Prove $\|\mathbf{\Pi x}\|_2^2$ concentrates within $1 \pm \epsilon$ of its expectation, which equals $\|\mathbf{x}\|_2^2$.

If high dimensional geometry is so different from low-dimensional geometry, why is dimensionality reduction possible? Doesn't Johnson-Lindenstrauss tell us that high-dimensional geometry can be approximated in low dimensions?

Hard case: $\mathbf{x}_1, \ldots, \mathbf{x}_n \in \mathbb{R}^d$ are all mutually orthogonal unit vectors:

$$\|\mathbf{x}_i - \mathbf{x}_j\|_2^2 = 2 \qquad \text{for all } i, j.$$

When we reduce to k dimensions with JL, we still expect these vectors to be nearly orthogonal. Why?

Hard case: $\mathbf{x}_1, \ldots, \mathbf{x}_n \in \mathbb{R}^d$ are all mutually orthogonal unit vectors:

$$\|\mathbf{x}_i - \mathbf{x}_j\|_2^2 = 2 \qquad \text{for all } i, j.$$

From our result earlier, in $O(\log n/\epsilon^2)$ dimensions, there exists $2^{O(\epsilon^2 \cdot \log n/\epsilon^2)} \geq n$ unit vectors that are close to mutually orthogonal. $O(\log n/\epsilon^2)$ = just enough dimensions.

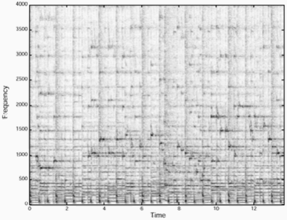
Alternative view: Without additional structure, we expect that learning/inference in d dimenions requires $2^{O(d)}$ data points. If we really had a data set that large, then the JL bound would be vacous, since $\log(n) = O(d)$.

The Johnson-Lindenstrauss Lemma let us sketch vectors and preserve their ℓ_2 Euclidean distance.

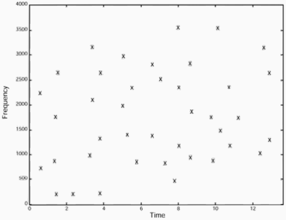
We also have dimensionality reduction techniques that preserve alternative measures of similarity.

How does **Shazam** match a song clip against a library of 8 million songs (32 TB of data) in a fraction of a second?

How does **Shazam** match a song clip against a library of 8 million songs (32 TB of data) in a fraction of a second?



Spectrogram extracted from audio clip.



Processed spectrogram: used to construct audio "fingerprint" $\mathbf{q} \in \{0,1\}^d$.

Each clip is represented by a high dimensional binary vector $\mathbf{q}$.

1	0	1	1	0	0	0	1	0	0	0	0	1	1	0	1

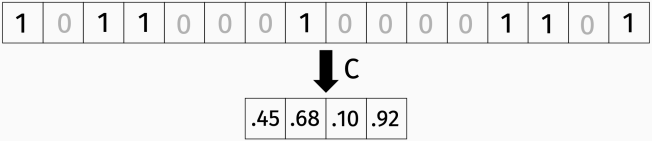
Given $\mathbf{q}$, find any nearby “fingerprint” $\mathbf{y}$ in a database – i.e. any $\mathbf{y}$ with $\text{dist}(\mathbf{y}, \mathbf{q})$ small.

Challenges:

- Database is possibly huge: $O(nd)$ bits.
- Expensive to compute $\text{dist}(\mathbf{y}, \mathbf{q})$: $O(d)$ time.

Goal: Design a more compact sketch for comparing $\mathbf{q}, \mathbf{y} \in \{0,1\}^d$. Ideally $\ll d$ space/time complexity.

$$C(\mathbf{q}) \in \mathbb{R}^k$$
$$C(\mathbf{y}) \in \mathbb{R}^k$$



As in Johnson-Lindenstrauss compressions, we want that $C(\mathbf{q})$ is similar to $C(\mathbf{y})$ if $\mathbf{q}$ is similar to $\mathbf{y}$.

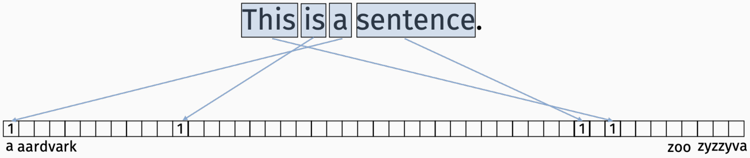
Definition (Jaccard Similarity)

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

Natural similarity measure for binary vectors. $0 \leq J(\mathbf{q}, \mathbf{y}) \leq 1$.

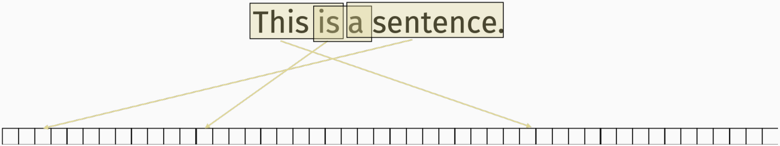
Can be applied to any data which has a natural binary representation (more than you might think).

"Bag-of-words" model:

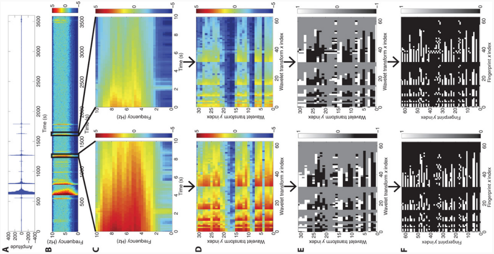


How many words do a pair of documents have in common?

"Bag-of-words" model:



How many bigrams do a pair of documents have in common?



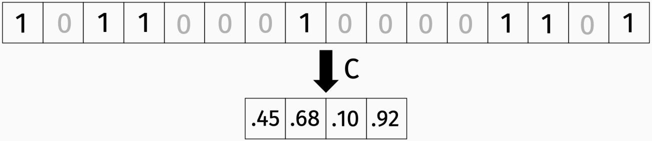
Feature extract pipeline for earthquake data.

(see paper by Rong et al. posted on course website)

APPLICATIONS: DOCUMENT SIMILARITY

- Finding duplicate or new duplicate documents or webpages.
- Change detection for high-speed web caches.
- Finding near-duplicate emails or customer reviews which could indicate spam.

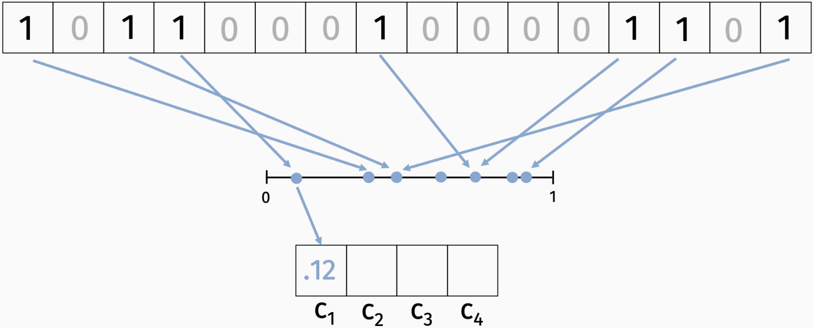
Goal: Design a compact sketch $C : \{0,1\} \to \mathbb{R}^k$:



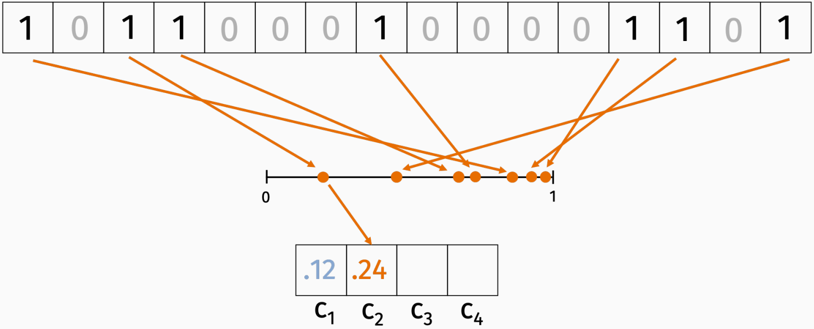
Want to use $C(\mathbf{q}), C(\mathbf{y})$ to approximately compute the Jaccard similarity $J(\mathbf{q},\mathbf{y}) = \frac{|\mathbf{q}\cap\mathbf{y}|}{|\mathbf{q}\cup\mathbf{y}|}$.

MinHash (Broder, '97):

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

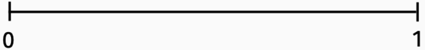


- Choose k random hash functions $h_1, \ldots, h_k : \{1, \ldots, n\} \rightarrow [0, 1]$.
- For $i \in 1, \ldots, k$,
    - Let $c_i = \min_{j, \mathbf{q}_j = 1} h_i(j)$.
- $C(\mathbf{q}) = [c_1, \ldots, c_k]$.



Claim: For all i, $\Pr[c_i(\mathbf{q}) = c_i(\mathbf{y})] = J(\mathbf{q}, \mathbf{y}) = \frac{|\mathbf{q} \cap \mathbf{y}|}{|\mathbf{q} \cup \mathbf{y}|}$.

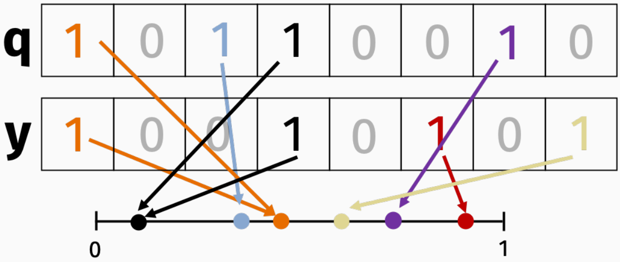
q	1	0	1	1	0	0	1	0
y	1	0	0	1	0	1	0	1



Proof:

1. For $c_i(\mathbf{q}) = c_i(\mathbf{y})$, we need that $\arg\min_{i \in \mathbf{q}} h(i) = \arg\min_{i \in \mathbf{y}} h(i)$.

Claim: $\Pr[c_i(\mathbf{q}) = c_i(\mathbf{y})] = J(\mathbf{q}, \mathbf{y})$.



2. Every non-zero index in $\mathbf{q} \cup \mathbf{y}$ is equally likely to produce the lowest hash value. $c_i(\mathbf{q}) = c_i(\mathbf{y})$ only if this index is 1 in both $\mathbf{q}$ and $\mathbf{y}$. There are $\mathbf{q} \cap \mathbf{y}$ such indices. So:

$$\Pr[c_i(\mathbf{q}) = c_i(\mathbf{y})] = \frac{|\mathbf{q} \cap \mathbf{y}|}{|\mathbf{q} \cup \mathbf{y}|} = J(\mathbf{q}, \mathbf{y})$$

Let $J = J(\mathbf{q}, \mathbf{y})$ denote the Jaccard similarity between **q** and **y**.

Return: $\tilde{J} = \frac{1}{k}\sum_{i=1}^{k} \mathbb{1}[c_i(\mathbf{q}) = c_i(\mathbf{y})]$.

Unbiased estimate for Jaccard similarity:

$$\mathbb{E}\tilde{J} =$$

C(q)	.12	.24	.76	.35

C(y)	.12	.98	.76	.11

The more repetitions, the lower the variance.

Let $J = J(\mathbf{q}, \mathbf{y})$ denote the true Jaccard similarity.

Estimator: $\tilde{J} = \frac{1}{k}\sum_{i=1}^{k} \mathbb{1}[c_i(\mathbf{q}) = c_i(\mathbf{y})]$.

$$\text{Var}[\tilde{J}] =$$

Plug into Chebyshev inequality. How large does k need to be so that with probability $> 1 - \delta$, $|J - \tilde{J}| \leq \epsilon$?

Chebyshev inequality: As long as $k = O\left(\frac{1}{\epsilon^2 \delta}\right)$, then with prob. $1 - \delta$,

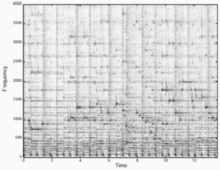
$$J(\mathbf{q}, \mathbf{y}) - \epsilon \leq \tilde{J}(C(\mathbf{q}), C(\mathbf{y})) \leq J(\mathbf{q}, \mathbf{y}) + \epsilon.$$

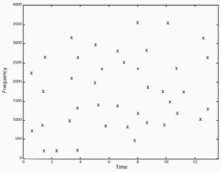
And $\tilde{J}$ only takes $O(k)$ time to compute! Independent of original fingerprint dimension d.

Can be improved to $\log(1/\delta)$ dependence. Can anyone tell me how?

SIMILARITY SKETCHING

input data





high dimensional vector representation

1	0	1	1	0	0	0	1	0	0	0	0	1	1	0	1

.45	.68	.10	.92

sketched representation

BREAK

Common goal: Find all vectors in database $\mathbf{q}_1, \ldots, \mathbf{q}_n \in \mathbb{R}^d$ that are close to some input query vector $\mathbf{y} \in \mathbb{R}^d$. I.e. find all of $\mathbf{y}$'s "nearest neighbors" in the database.

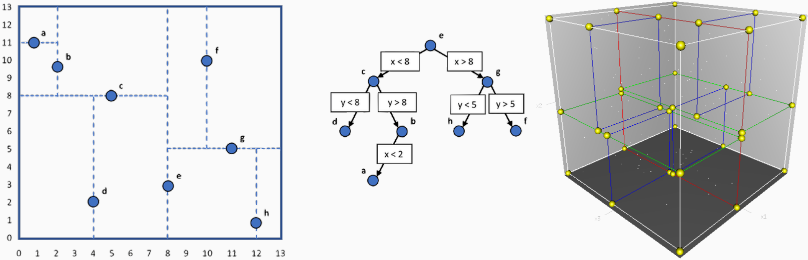
- The Shazam problem.
- Audio + video search.
- Finding duplicate or near duplicate documents.
- Detecting seismic events.

How does similarity sketching help in these applications?

- Improves runtime of "linear scan" from $O(nd)$ to $O(nk)$.
- Improves space complexity from $O(nd)$ to $O(nk)$. This can be super important – e.g. if it means the linear scan only accesses vectors in fast memory.

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

This problem can already be solved for a small number of dimensions using space partitioning approaches (e.g. kd-tree).



Runtime is roughly $O(d \cdot \min(n, 2^d))$, which is only sublinear for $d = o(\log n)$.

Only been attacked much more recently:

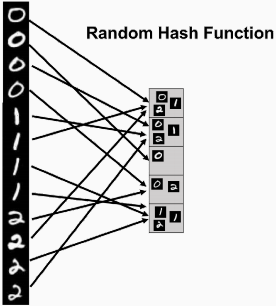
- Locality-sensitive hashing [Indyk, Motwani, 1998]
- Spectral hashing [Weiss, Torralba, and Fergus, 2008]
- Vector quantization [Jégou, Douze, Schmid, 2009]

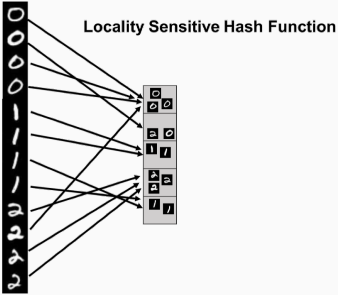
Key Insight of LSH: Trade worse space-complexity for better time-complexity. I.e. typically use more than $O(n)$ space.

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

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

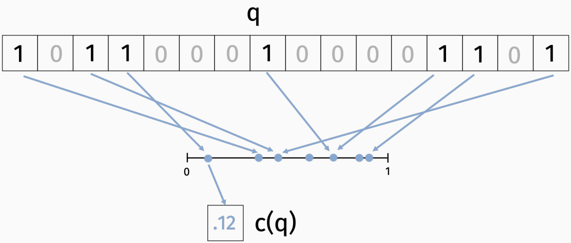
- Higher when $\mathbf{q}$ and $\mathbf{y}$ are more similar, i.e. $s(\mathbf{q}, \mathbf{y})$ is higher.
- Lower when $\mathbf{q}$ and $\mathbf{y}$ are more dissimilar, i.e. $s(\mathbf{q}, \mathbf{y})$ is lower.





LSH for $s(\mathbf{q}, \mathbf{y})$ equal to 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 uniform random hash function.
- Let $h(\mathbf{q}) = g(c(\mathbf{q}))$.



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 uniform random hash function.
- Let $h(\mathbf{x}) = g(c(\mathbf{x}))$.

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

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

Basic approach for near neighbor search in a database.

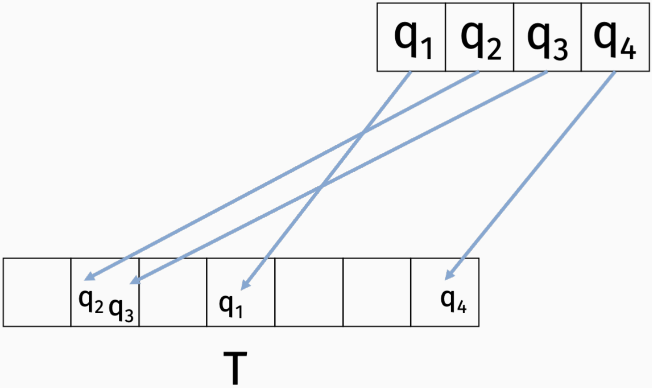
Pre-processing:

- Select random LSH function $h : \{0,1\}^d \to 1, \ldots, m$.
- Create table T with $m = O(n)$ slots.[1]
- 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 $\mathbf{q} \in T(h(\mathbf{y}))$ and return any that are close to $\mathbf{y}$. Time required is $O(d \cdot |T(h(\mathbf{y}))|)$.

[1] Enough to make the $O(1/m)$ term negligible.


q_1
q_2
q_3
q_4
q_2 q_3
q_1
q_4
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 that a vector in $T(h(\mathbf{y}))$ is not close to $\mathbf{y}$?

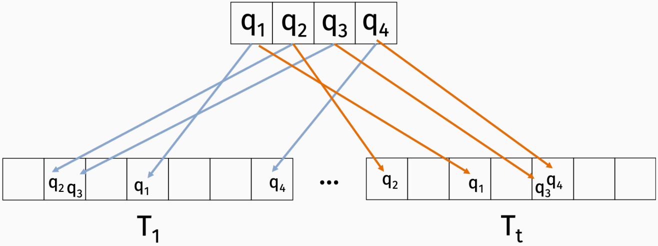
A higher false negative rate means we miss near neighbors.

A higher false positive rate means increased runtime – we need to compute $J(\mathbf{q}, \mathbf{y})$ for every $\mathbf{q} \in T(h(\mathbf{y}))$ to check if it's actually close to $\mathbf{y}$.

Note: The meaning of "close" and "not close" is application dependent. E.g. we might specify that we want to find anything with Jaccard similarity $> .4$, but not with Jaccard similarity $< .2$.

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 $h_1, \ldots, h_t : \{0,1\}^d \to 1, \ldots, m$.
- 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 $T_1(h_1(\mathbf{y})) \cup T_2(h_2(\mathbf{y})) \cup \ldots, T_t(h_t(\mathbf{y}))$.

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

What's the probability we find q?

(10, 99%)

WHAT HAPPENS TO FALSE POSITIVES?

Suppose there is some other database point **z** with $J(\mathbf{y}, \mathbf{z}) = .2$.

What is the probability we will need to compute $J(\mathbf{z}, \mathbf{y})$ in our hashing scheme with one table? I.e. the probability that **y** hashes into at least one bucket containing **z**.

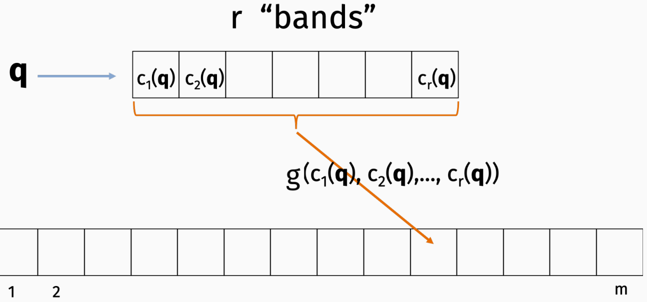
In the new scheme with $t = 10$ tables?

(89%)

Change our locality sensitive hash function.

Tunable LSH for Jaccard similarity:

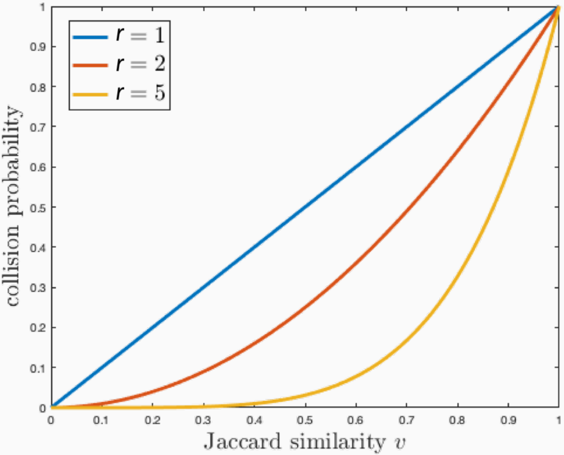
- Choose parameter $r \in \mathbb{Z}^+$.
- Let $c_1, \ldots, c_r : \{0,1\}^d \rightarrow [0,1]$ be random MinHash.
- Let $g : [0,1]^r \rightarrow \{1, \ldots, m\}$ be a uniform random hash function.
- Let $h(\mathbf{x}) = g(c_1(\mathbf{x}), \ldots, c_r(\mathbf{x}))$.



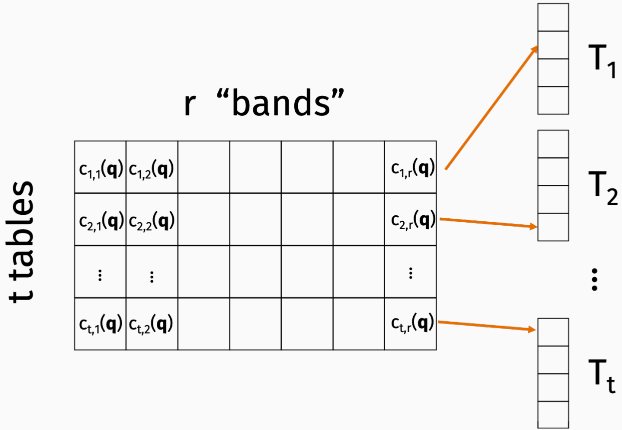
Tunable LSH for Jaccard similarity:

- Choose parameter $r \in \mathbb{Z}^+$.
- Let $c_1, \ldots, c_r : \{0,1\}^d \to [0,1]$ be random MinHash.
- Let $g : [0,1]^r \to \{1, \ldots, m\}$ be a uniform random hash function.
- Let $h(\mathbf{x}) = g(c_1(\mathbf{x}), \ldots, c_r(\mathbf{x}))$.

If $J(\mathbf{q}, \mathbf{y}) = v$, then $\Pr[h(\mathbf{q}) == h(\mathbf{y})] =$


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

Full LSH cheme has two parameters to tune:



Effect of **increasing number of tables** t on:

False Negatives

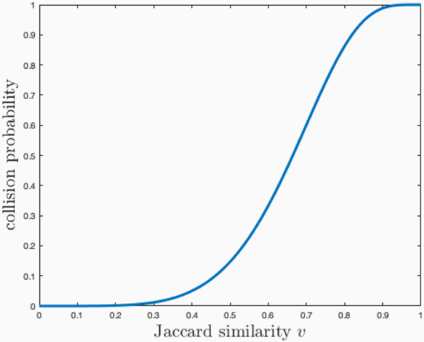
False Positives

Effect of **increasing number of bands** r on:

False Negatives

False Positives

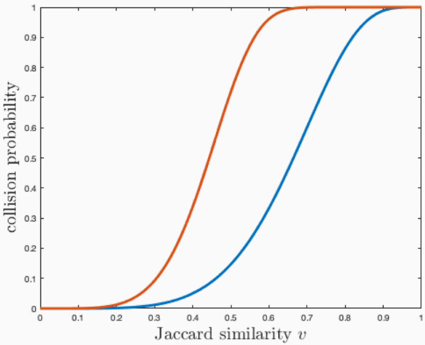
Probability we check **q** when querying **y** if $J(\mathbf{q}, \mathbf{y}) = v$:



$r = 5, t = 5$

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

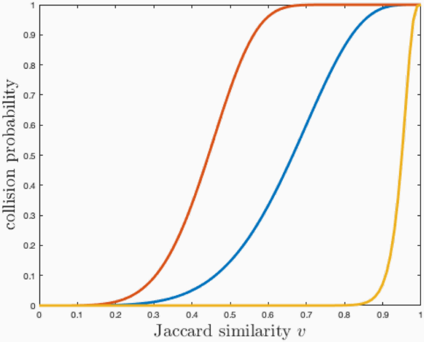
$$\approx 1 - (1 - v^r)^t$$



$$r = 5, t = 40$$

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

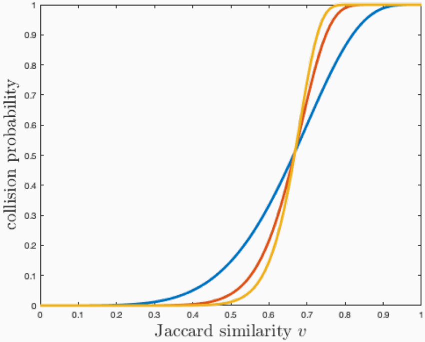
$$\approx 1 - (1 - v^r)^t$$



$r = 40, t = 5$

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

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



Increasing both r and t gives a steeper curve.

Better for search, but worse space complexity.

FIXED THRESHOLD

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 $J(\mathbf{y}, \mathbf{q}) \in [.7, .9]$.
- All other items have $J(\mathbf{y}, \mathbf{q}) < .7$.

With $r = 25$ 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 $\lesssim 1 - (1 - .7^{25})^{40} = .005$

Upper bound on total number of items checked:

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

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

Directly trade space for fast search.

Near Neighbor Search Problem

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 $\mathbf{y}$.

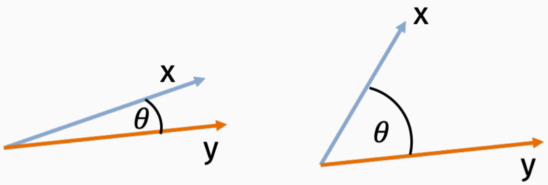
Theorem (Indyk, Motwani, 1998)

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

Good locality sensitive hash functions exists for 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 $\|\mathbf{x}\|_2^2 = \|\mathbf{y}\|_2^2 = 1$.

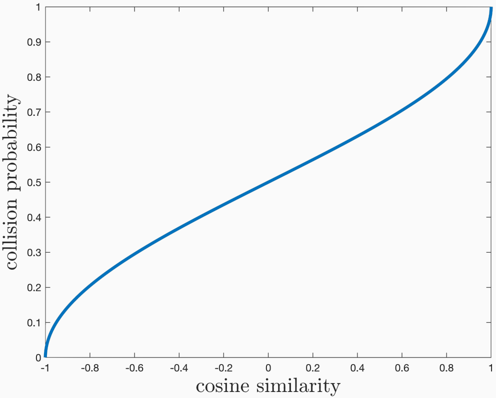
Locality sensitive hash for **cosine similarity**:

- Let $\mathbf{g} \in \mathbb{R}^d$ be randomly chosen with each entry $\mathcal{N}(0, 1)$.
- Let $f : \{-1, 1\} \rightarrow \{1, \ldots, m\}$ be a uniformly random hash function.
- $h : \mathbb{R}^d \rightarrow \{1, \ldots, m\}$ is definied $h(\mathbf{x}) = f(\text{sign}(\langle \mathbf{g}, \mathbf{x} \rangle))$.

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

Theorem (to be prove): If $\cos(\theta(\mathbf{x},\mathbf{y})) = v$, then

$$\Pr[h(\mathbf{x}) == h(\mathbf{y})] = 1 - \frac{\theta}{\pi} + \frac{1}{m} = 1 - \frac{\cos^{-1}(v)}{\pi} + \frac{1}{m}$$

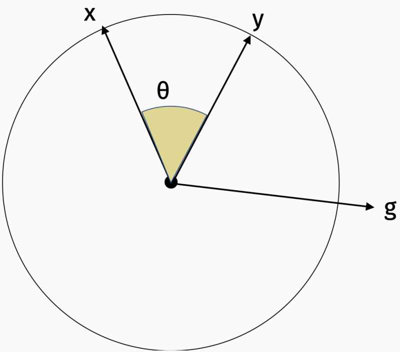


SimHash can be tuned, just like our MinHash based LSH function for Jaccard similarity:

- Let $\mathbf{g}_1, \ldots, \mathbf{g}_r \in \mathbb{R}^d$ be randomly chosen with each entry $\mathcal{N}(0,1)$.
- Let $f : \{-1,1\}^r \to \{1,\ldots,m\}$ be a uniformly random hash function.
- $h : \mathbb{R}^d \to \{1,\ldots,m\}$ is defined $h(\mathbf{x}) = f([\text{sign}(\langle \mathbf{g}_1, \mathbf{x}\rangle), \ldots, \text{sign}(\langle \mathbf{g}_r, \mathbf{x}\rangle)])$.

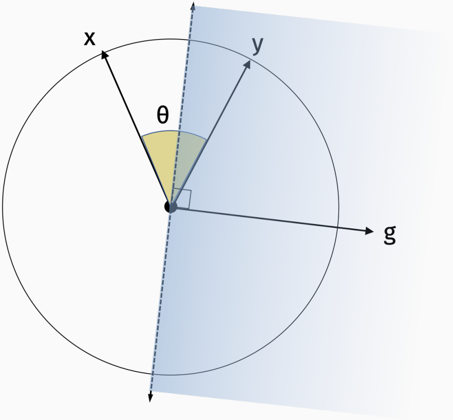
$$\Pr[h(\mathbf{x}) == h(\mathbf{y})] = \left(1 - \frac{\theta}{\Pi}\right)^r$$

To prove: $\Pr[h(\mathbf{x}) == h(\mathbf{y})] = 1 - \frac{\theta}{\pi}$, where $h(\mathbf{x}) = f(\text{sign}(\langle \mathbf{g}, \mathbf{x} \rangle))$ and f is uniformly random hash function.

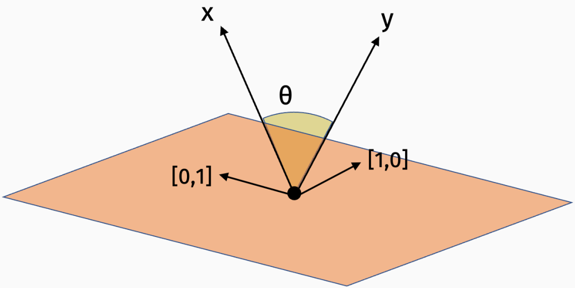


$$\Pr[h(\mathbf{x}) == h(\mathbf{y})] = z + \frac{1-v}{m} \approx z.$$

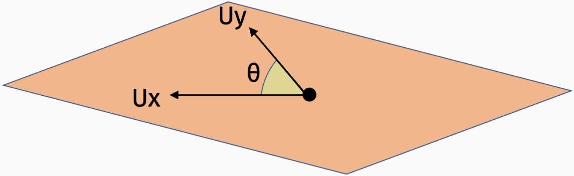
where $z = \Pr[\text{sign}(\langle \mathbf{g}, \mathbf{x} \rangle) == \text{sign}(\langle \mathbf{g}, \mathbf{y} \rangle)]$



$\Pr[h(\mathbf{x}) == h(\mathbf{y})] \approx$ probability **x** and **y** are on the same side of hyperplane orthogonal to **g**.



There is always some rotation matrix $\mathbf{U}$ such that $\mathbf{Ux}, \mathbf{Uy}$ are spanned by the first two-standard basis vectors and have the same cosine similarity as $\mathbf{x}$ and $\mathbf{y}$.



There is always some rotation matrix $\mathbf{U}$ such that $\mathbf{x}, \mathbf{y}$ are spanned by the first two-standard basis vectors.

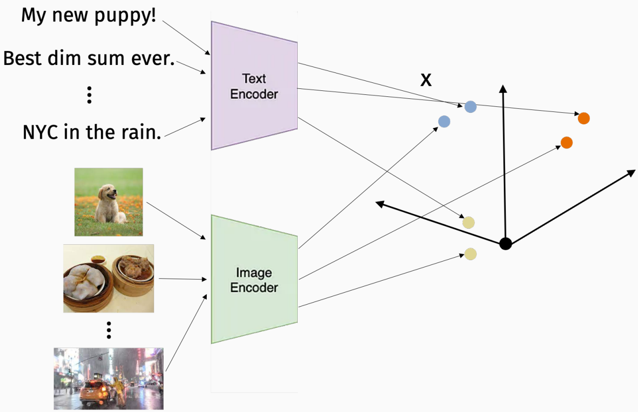
Note: A rotation matrix $\mathbf{U}$ has the property that $\mathbf{U}^T\mathbf{U} = \mathbf{I}$. I.e., $\mathbf{U}^T$ is a rotation matrix itself, which reverses the rotation of $\mathbf{U}$.

Claim:

$$
\begin{aligned}
1 - \frac{\theta}{\pi} &= \Pr[\text{sign}(\langle \mathbf{g}[1,2], (\mathbf{Ux})[1,2] \rangle) == \text{sign}(\langle \mathbf{g}[1,2], (\mathbf{Uy}[1,2] \rangle)] \\
&= \Pr[\text{sign}(\langle \mathbf{g}, \mathbf{Ux} \rangle) == \text{sign}(\langle \mathbf{g}, \mathbf{Uy} \rangle)] \\
&= \Pr[\text{sign}(\langle \mathbf{g}, \mathbf{x} \rangle) == \text{sign}(\langle \mathbf{g}, \mathbf{y} \rangle)]
\end{aligned}
$$

Why?

- High-dimensional vector search is exploding as a research area with the rise of machine-learned multi-modal embeddings for images, text, and more.



Web-scale image search is now a vector search problem.

GRAPH BASED NEAR NEIGBHOR