CS-GY 6763: Lecture 4
Dimensionality reduction + Near neighbor search in high dimensions

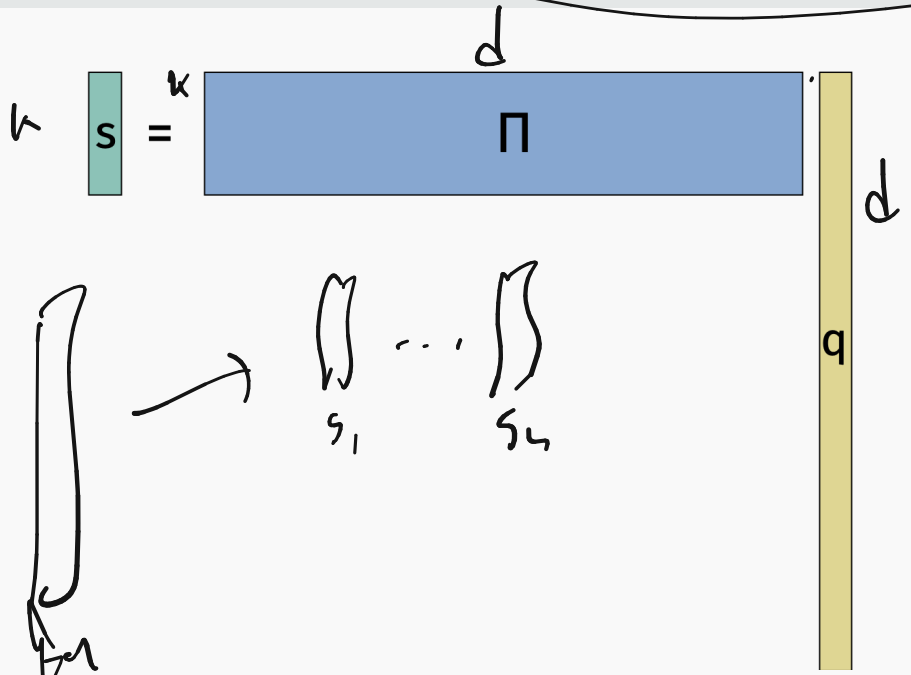
NYU Tandon School of Engineering, Prof. Christopher Musco

EUCLIDEAN DIMENSIONALITY REDUCTION

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

$$(1-\epsilon)\|\mathbf{q}_i - \mathbf{q}_j\|_2^2 \leq \|\Pi\mathbf{q}_i - \Pi\mathbf{q}_j\|_2^2 \leq (1+\epsilon)\|\mathbf{q}_i - \mathbf{q}_j\|_2^2.$$



Please remember: This is equivalent to:

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

$$(1-\epsilon)\|\mathbf{q}_i - \mathbf{q}_j\|_2^2 \leq \|\mathbf{\Pi}\mathbf{q}_i - \mathbf{\Pi}\mathbf{q}_j\|_2^2 \leq (1+\epsilon)\|\mathbf{q}_i - \mathbf{q}_j\|_2^2.$$

because for small ϵ, $(1+\epsilon)^2 = 1 + O(\epsilon)$ and $(1-\epsilon)^2 = 1 - O(\epsilon)$.

$\leq 1 + 3\epsilon$ $\quad$ $\geq 1 - 2\epsilon$

And this is equivalent to:

Lemma (Johnson-Lindenstrauss, 1984)

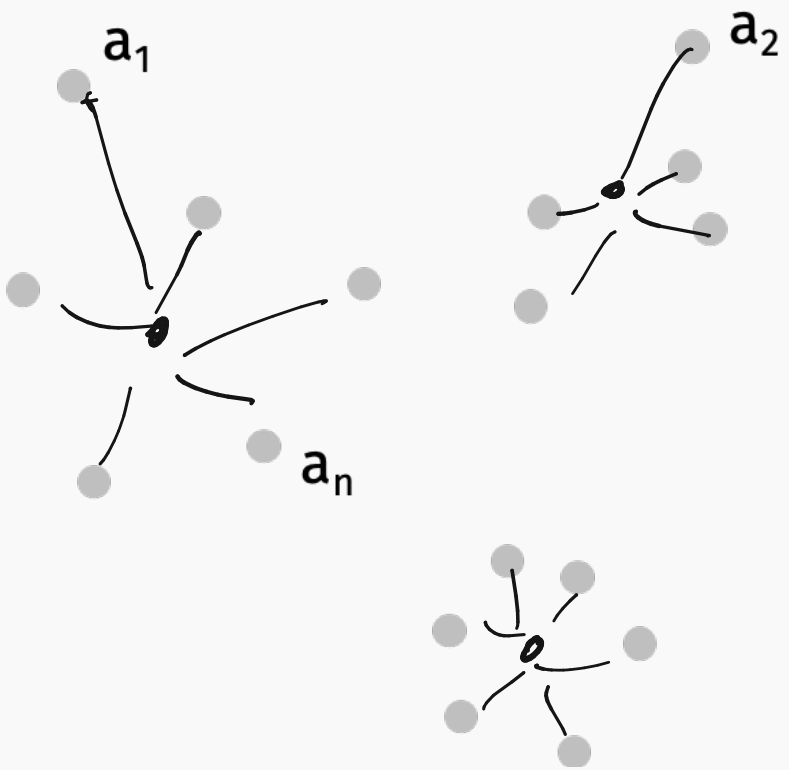
For any set of n data points $\mathbf{q}_1, \ldots, \mathbf{q}_n \in \mathbb{R}^d$ there exists a linear map $\Pi : \mathbb{R}^d \to \mathbb{R}^k$ where $k = O\left(\frac{\log n}{\epsilon^2}\right)$ such that for all i, j,

$$(1-\epsilon)\|\mathbf{\Pi}\mathbf{q}_i - \mathbf{\Pi}\mathbf{q}_j\|_2^2 \leq \|\mathbf{q}_i - \mathbf{q}_j\|_2^2 \leq (1+\epsilon)\|\mathbf{\Pi}\mathbf{q}_i - \mathbf{\Pi}\mathbf{q}_j\|_2^2.$$

because for small ϵ, $\frac{1}{1+\epsilon} = 1 - O(\epsilon)$ and $\frac{1}{1-\epsilon} = 1 + O(\epsilon)$.

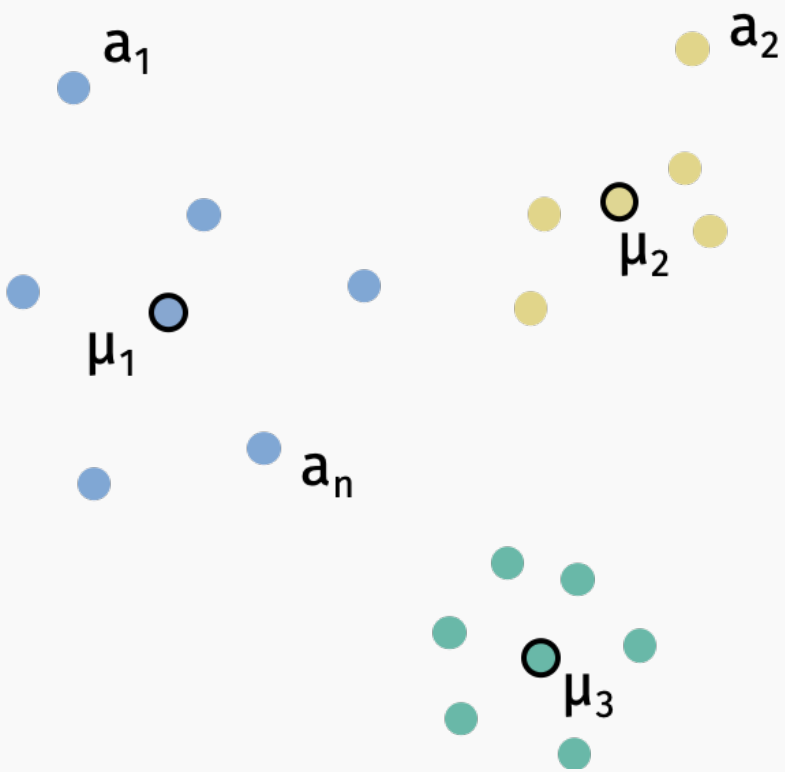
k-means clustering: Give data points $\mathbf{a}_1, \ldots, \mathbf{a}_n \in \mathbb{R}^d$, find centers $\boldsymbol{\mu}_1, \ldots, \boldsymbol{\mu}_k \in \mathbb{R}^d$ to minimize:

$$Cost(\boldsymbol{\mu}_1, \ldots, \boldsymbol{\mu}_k) = \sum_{i=1}^{n} \min_{j=1,\ldots,k} \|\boldsymbol{\mu}_j - \mathbf{a}_i\|_2^2$$



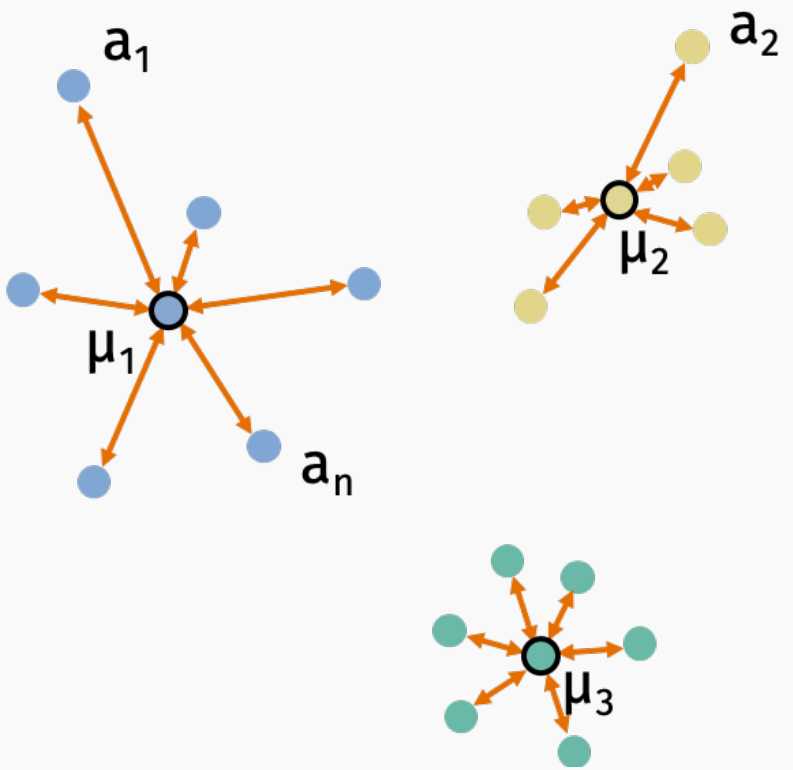
k-means clustering: Give data points $\mathbf{a}_1, \ldots, \mathbf{a}_n \in \mathbb{R}^d$, find centers $\boldsymbol{\mu}_1, \ldots, \boldsymbol{\mu}_k \in \mathbb{R}^d$ to minimize:

$$Cost(\boldsymbol{\mu}_1, \ldots, \boldsymbol{\mu}_k) = \sum_{i=1}^{n} \min_{j=1,\ldots,k} \|\boldsymbol{\mu}_j - \mathbf{a}_i\|_2^2$$



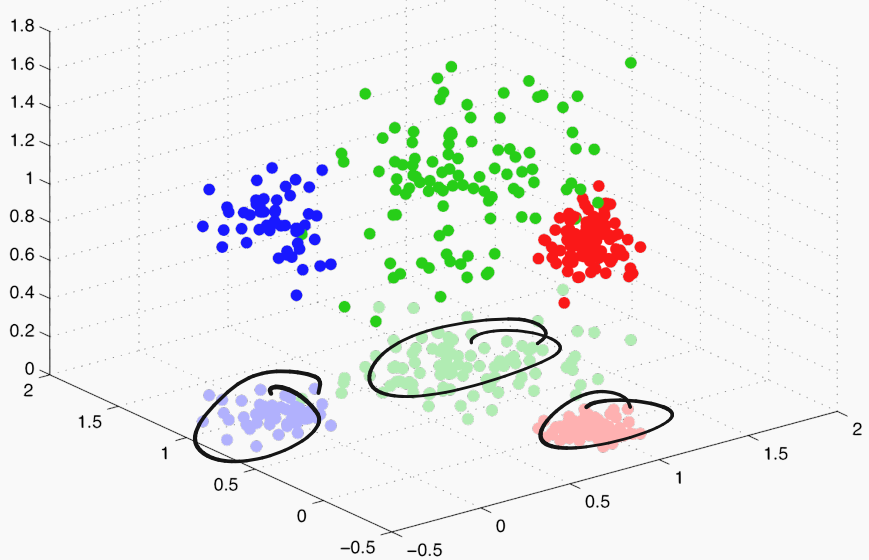
k-means clustering: Give data points $\mathbf{a}_1, \ldots, \mathbf{a}_n \in \mathbb{R}^d$, find centers $\boldsymbol{\mu}_1, \ldots, \boldsymbol{\mu}_k \in \mathbb{R}^d$ to minimize:

$$Cost(\boldsymbol{\mu}_1, \ldots, \boldsymbol{\mu}_k) = \sum_{i=1}^{n} \min_{j=1,\ldots,k} \|\boldsymbol{\mu}_j - \mathbf{a}_i\|_2^2$$



NP hard to solve exactly, but there are many good approximation algorithms. All depend at least linearly on the dimension d.

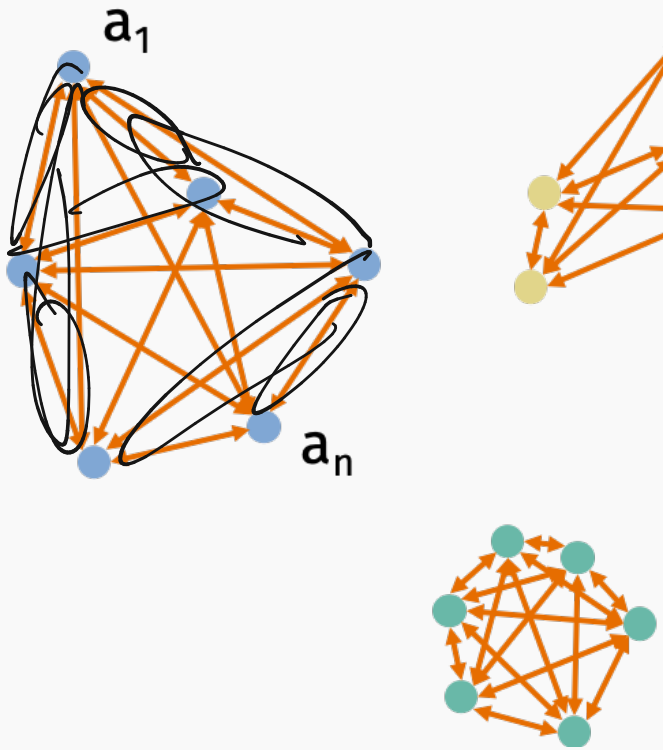
Approximation scheme: Find clusters $\tilde{C}_1, \ldots, \tilde{C}_k$ for the $k = O\left(\frac{\log n}{\epsilon^2}\right)$ dimension data set $\mathbf{\Pi a}_1, \ldots, \mathbf{\Pi a}_n$.



Argue these clusters are near optimal for $\mathbf{a}_1, \ldots, \mathbf{a}_n$.

Equivalent form: Find clusters $C_1, \ldots, C_k \subseteq \{1, \ldots, n\}$ to minimize:

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



Exercise: Prove this to your self.

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

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

Claim: For any clusters $C_1, \ldots, C_k$:

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

Suppose we use an approximation algorithm to find clusters $B_1, \ldots, B_k$ such that:

$$\widetilde{Cost}(B_1, \ldots, B_k) \leq (1+\alpha)\widetilde{Cost}^*$$

Then:

$$\begin{aligned} Cost(B_1, \ldots, B_k) &\leq \frac{1}{1-\epsilon}\widetilde{Cost}(B_1, \ldots, B_k) \\ &\leq (1+O(\epsilon))(1+\alpha)\widetilde{Cost}^* \\ &\leq (1+O(\epsilon))(1+\alpha)(1+\epsilon)Cost^* \\ &= (1+O(\alpha+\epsilon))\,Cost^* \end{aligned}$$

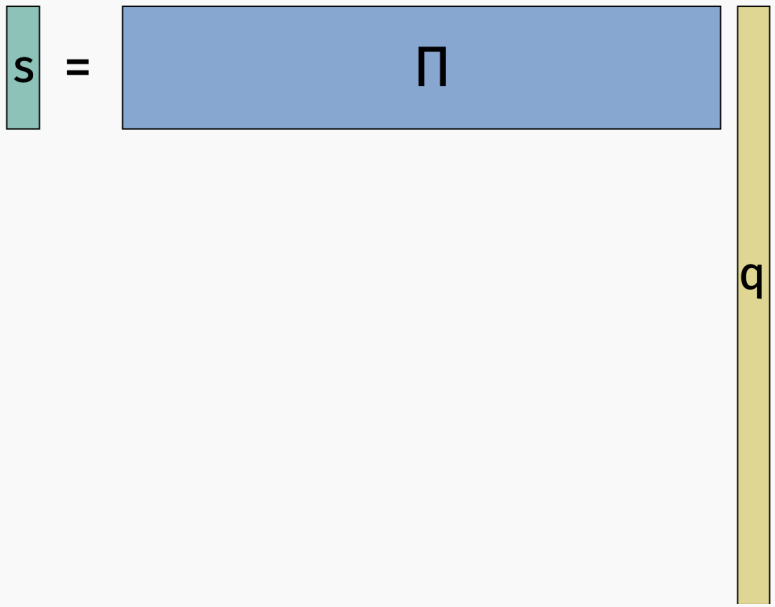
$(1+O(\epsilon))(1+\alpha)(1+\epsilon) = (1+O(\alpha+\epsilon))\,Cost^*$

$Cost^* = \min_{C_1,\ldots,C_k} Cost(C_1, \ldots, C_k)$ and
$\widetilde{Cost}^* = \min_{C_1,\ldots,C_k} \widetilde{Cost}(C_1, \ldots, C_k)$

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 Π: $\mathbb{R}^d \to \mathbb{R}^k$ where $k = O\left(\frac{\log n/\delta}{\epsilon^2}\right)$ such that for all i, j,

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



Remarkably, Π can be chosen completely at random!

One possible construction: Random Gaussian.

$$\Pi_{i,j} = \frac{1}{\sqrt{k}}\mathcal{N}(0,1)$$

The map Π is oblivious to the data set. This stands in contrast to e.g. PCA, amoung other differences.

[Indyk, Motwani 1998] [Arriage, Vempala 1999] [Achlioptas 2001] [Dasgupta, Gupta 2003].

Many other possible choices suffice – you can use random $\{+1, -1\}$ variables, sparse random matrices, pseudorandom Π. Each with different advantages.

RANDOMIZED JL CONSTRUCTIONS

Let $\mathbf{\Pi} \in \mathbb{R}^{k \times d}$ be 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;
```

or

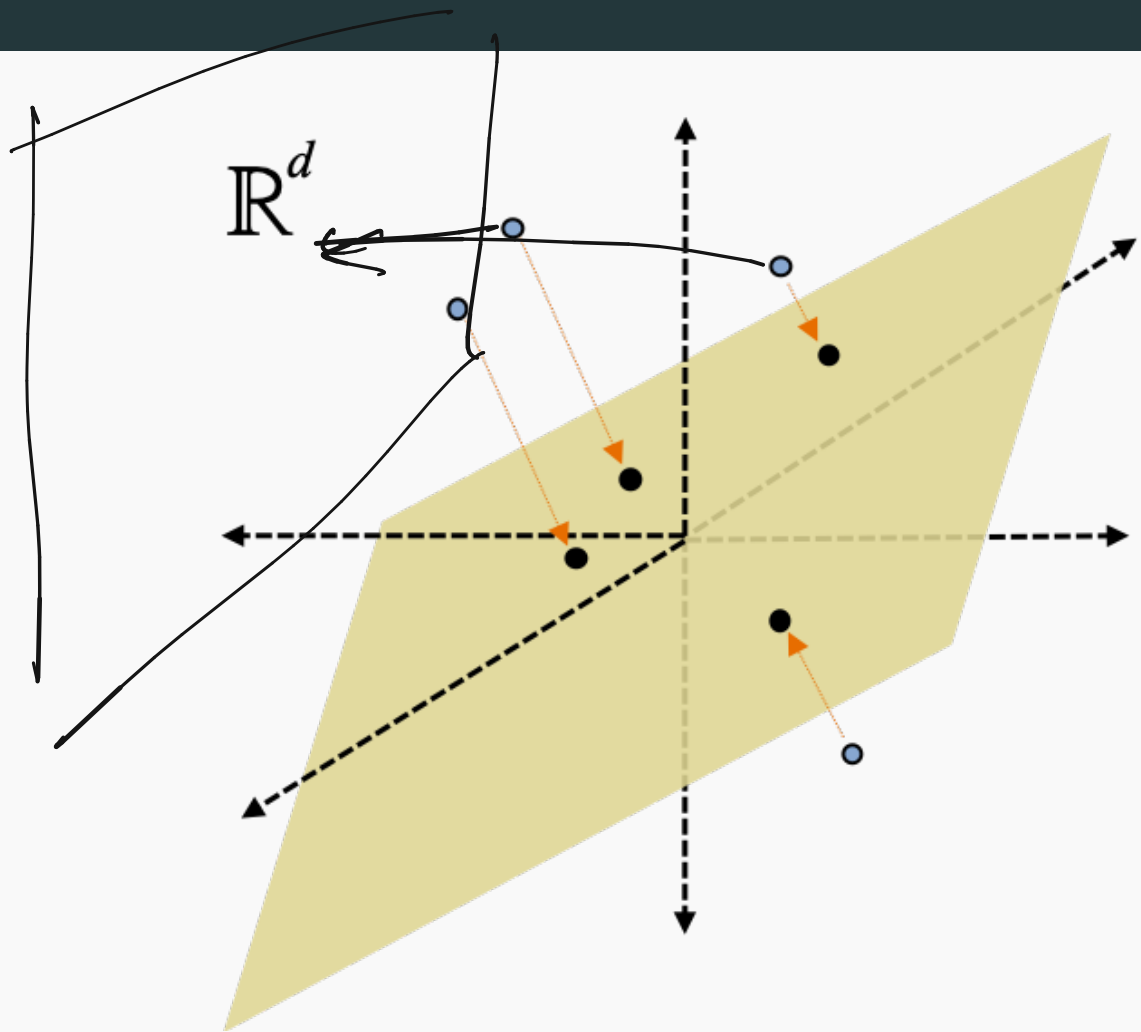
1	1	-1	-1	-1	-1	-1	-1	1	-1	-1	1	-1	-1	1	1	-1
1	1	1	-1	1	-1	-1	-1	1	1	1	1	-1	1	-1	-1	-1
1	1	-1	-1	-1	1	-1	-1	1	1	-1	1	-1	1	-1	1	-1
-1	-1	-1	1	1	-1	-1	-1	-1	-1	-1	-1	-1	1	1	1	1
1	-1	1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	1	1	-1	1
1	-1	-1	1	-1	1	1	-1	-1	-1	1	-1	-1	-1	1	1	1
1	1	-1	1	1	-1	1	-1	1	-1	1	-1	1	1	1	-1	-1
-1	-1	-1	-1	-1	-1	1	-1	1	1	-1	-1	1	-1	-1	1	1
-1	-1	1	1	1	1	-1	-1	1	-1	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;
```

A random orthogonal matrix also works. I.e. with $\mathbf{\Pi}\mathbf{\Pi}^T = \mathbf{I}_{k \times k}$.

For this reason, the JL operation is often called a "random projection", even though it technically isn't a projection when entries are i.i.d.

RANDOM PROJECTION



Intuitively, close points will remain close after projection, and far points will remain far.

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?

JL FROM DISTRIBUTIONAL JL

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 $1 - \delta$,

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

JL FROM DISTRIBUTIONAL JL

By the Distributional JL Lemma, with probability $1 - \delta$, we have that for any $\mathbf{q}_i$, $\mathbf{q}_j$,

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

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

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

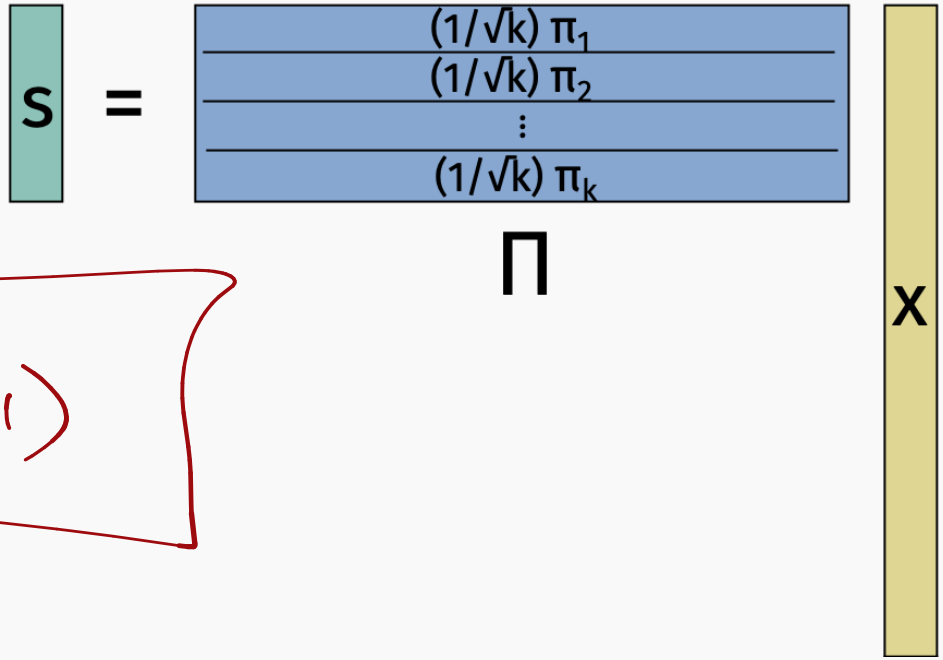
PROOF OF DISTRIBUTIONAL JL

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

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

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

Some notation:



$\frac{1}{\sqrt{k}}$ $\mathcal{N}(0,1)$

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

PROOF OF DISTRIBUTIONAL JL

Intermediate Claim:

$$\mathbb{E}\left[\|\mathbf{\Pi}\mathbf{x}\|_2^2\right] = \mathbb{E}\left[\langle \boldsymbol{\pi}_1, \mathbf{x}\rangle^2\right].$$

$= \|x\|_2^2$

$$\mathbb{E}\left[\sum_{i=1}^{k} S_i^2\right] = \sum_{i=1}^{k} \mathbb{E}[S_i^2] = \sum_{i=1}^{k} \mathbb{E}\left(\frac{1}{\sqrt{k}}\langle \pi_i, x\rangle\right)^2$$

$$= \frac{1}{k}\sum_{i=1}^{k} \mathbb{E}\langle \pi_i, x\rangle^2$$

$\pi_1, \ldots \pi_k$

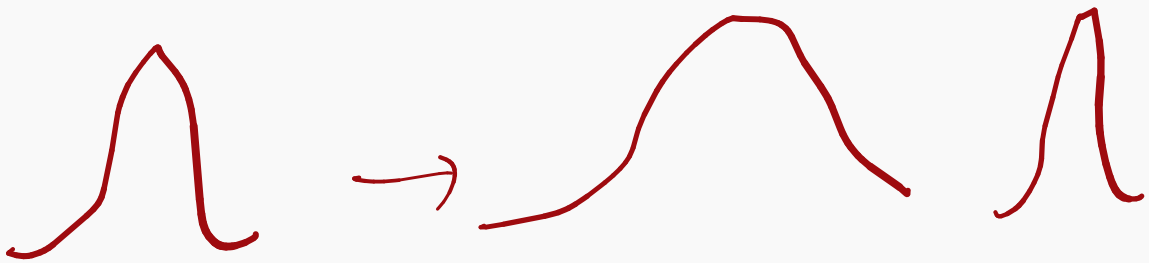
Goal: Prove $\mathbb{E}\left[\|\mathbf{\Pi}\mathbf{x}\|_2^2\right] = \|\mathbf{x}\|_2^2$.

PROOF OF DISTRIBUTIONAL JL

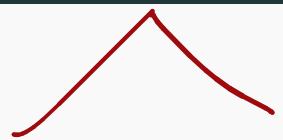
$$\langle \boldsymbol{\pi}_1, \mathbf{x} \rangle = Z_1 \cdot \mathbf{x}(1) + Z_2 \cdot \mathbf{x}(2) + \ldots + Z_d \cdot \mathbf{x}(d)$$

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

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



Goal: Prove $\mathbb{E}\left[\|\boldsymbol{\Pi}\mathbf{x}\|_2^2\right] = \|\mathbf{x}\|_2^2$. Have: $\mathbb{E}\left[\|\boldsymbol{\Pi}\mathbf{x}\|_2^2\right] = \mathbb{E}\left[\left(\langle \boldsymbol{\pi}_1, \mathbf{x} \rangle\right)^2\right]$



What type of random variable is $\langle \pi_i, x \rangle$?

Fact (Stability of Gaussian random variables)

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

$$\langle \pi_1, x \rangle = \mathcal{N}(0, x(1)^2) + \mathcal{N}(0, x(2)^2) + \ldots + \mathcal{N}(0, x(d)^2)$$
$$= \mathcal{N}(0, \|x\|_2^2).$$

$x(1)^2 + \ldots x(d)^2 = \|x\|_2^2$

So $\mathbb{E}\left[\|\Pi x\|_2^2\right] = \mathbb{E}\left[(\langle \pi_1, x \rangle)^2\right] = \|x\|_2^2$, as desired.

Goal: Prove $\mathbb{E}\left[\|\Pi x\|_2^2\right] = \|x\|_2^2$. Have: $\mathbb{E}\left[\|\Pi x\|_2^2\right] = \mathbb{E}\left[(\langle \pi_1, x \rangle)^2\right]$

PROOF OF DISTRIBUTIONAL JL

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

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

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

CONCENTRATION OF CHI-SQUARED RANDOM VARIABLES

Lemma

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

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

$$\Pr\left[\left|\|\Pi x\|_2^2 - \|x\|_2^2\right| \geq \epsilon \|x\|_2^2\right] \leq 2e^{-k\epsilon^2/8} = \delta$$

$$e^{-k\epsilon^2/8} = \delta/2 \qquad -k\epsilon^2/8 = \ln(\delta/2)$$

$$k\epsilon^2/8 = \ln(2/\delta) \qquad k = \frac{8\ln(2/\delta)}{\epsilon^2}$$

$$= O\left(\frac{\log(1/\delta)}{\epsilon^2}\right)$$

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

CONNECTION TO LAST LECTURE

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?

CONNECTION TO DIMENSIONALITY REDUCTION

Hard case: $x_1, \ldots, x_n \in \mathbb{R}^d$ are all mutually orthogonal unit vectors:

$$\|x_i - x_j\|_2^2 = 2 \quad \text{for all } i, j.$$

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.

DIMENSIONALITY REDUCTION

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!

JACCARD SIMILARITY

Another distance measure (actually a similarity measure) between binary vectors in $\{0,1\}^d$:

Definition (Jaccard Similarity)

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

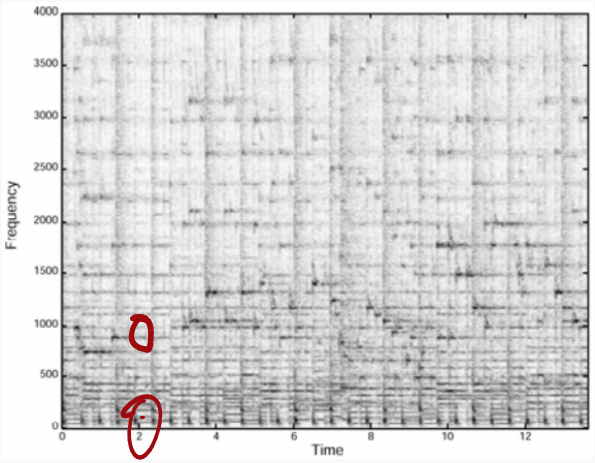
Natural similarity measure for binary vectors. $0 \leq J(\mathbf{q},\mathbf{y}) \leq 1$.

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

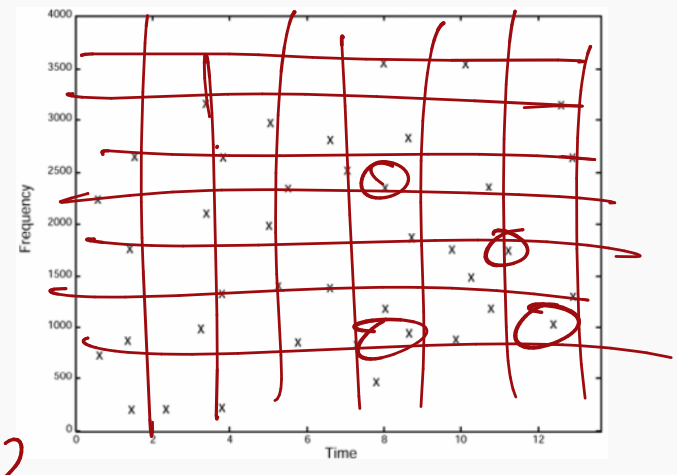
$$\mathbf{y} = \begin{bmatrix} 1 & 0 & 1 & 1 & 0 & 0 \end{bmatrix}$$
$$\mathbf{q} = \begin{bmatrix} 1 & 1 & 0 & 1 & 0 & 0 \end{bmatrix}$$

$\frac{2}{4} = \frac{1}{2}$

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

SIMILARITY ESTIMATION

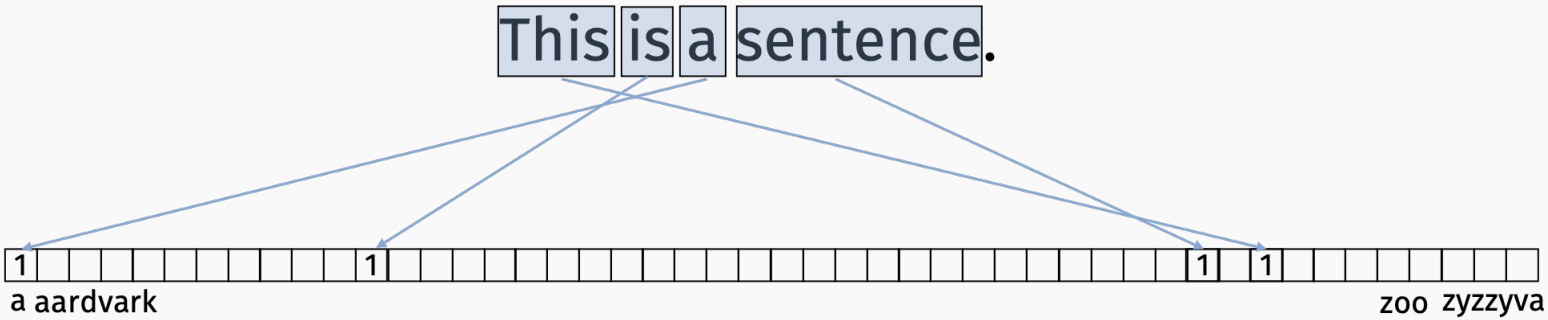
Given $\mathbf{q}$, find any nearby "fingerprint" $\mathbf{y}$ in a database – i.e. any $\mathbf{y}$ with $\text{dist}(\mathbf{y}, \mathbf{q})$ small.

Challenges:

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

JACCARD SIMILARITY FOR DOCUMENT COMPARISON

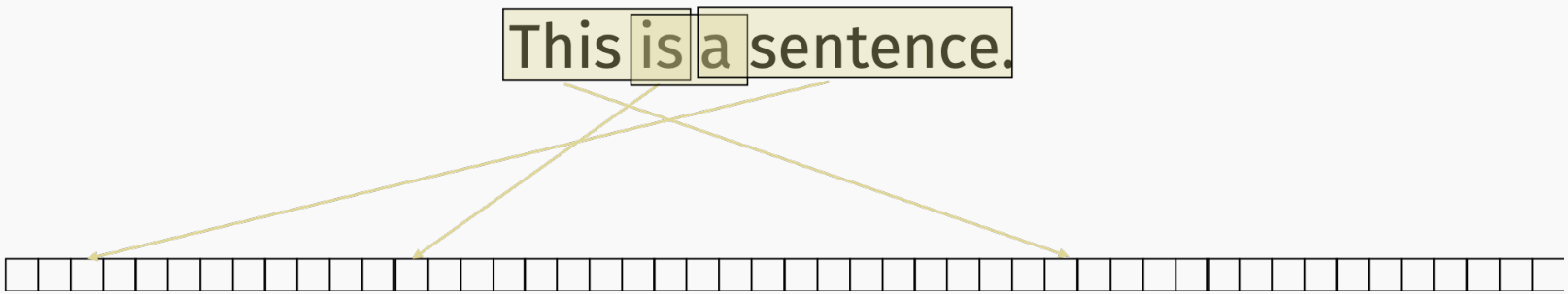
"Bag-of-words" model:



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

JACCARD SIMILARITY FOR DOCUMENT COMPARISON

"Bag-of-words" model:

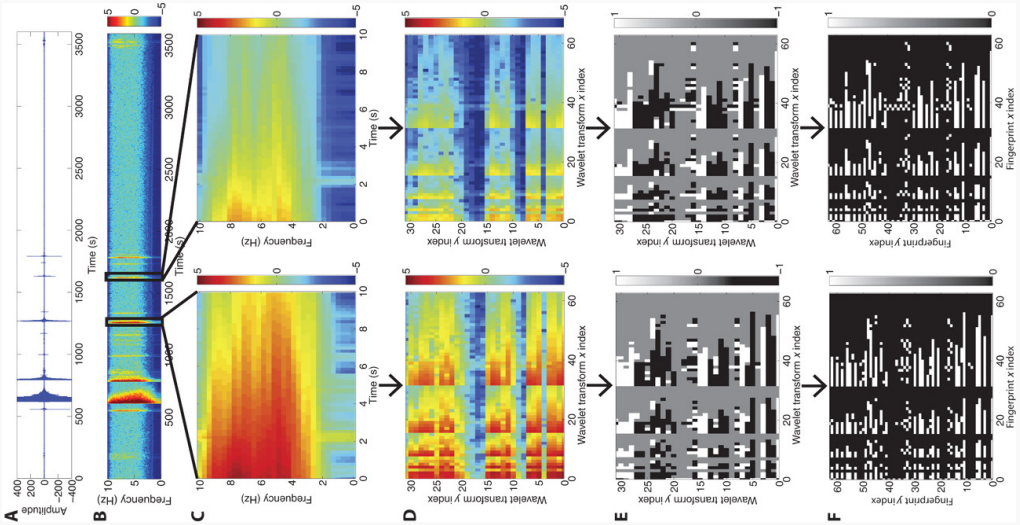


How many bigrams do a pair of documents have in common?

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.

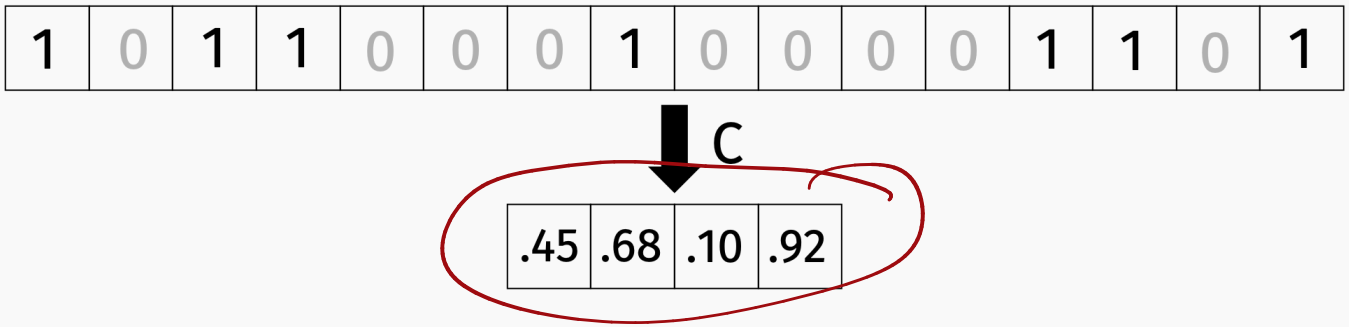
JACCARD SIMILARITY FOR SEISMIC DATA



Feature extract pipeline for earthquake data.

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

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



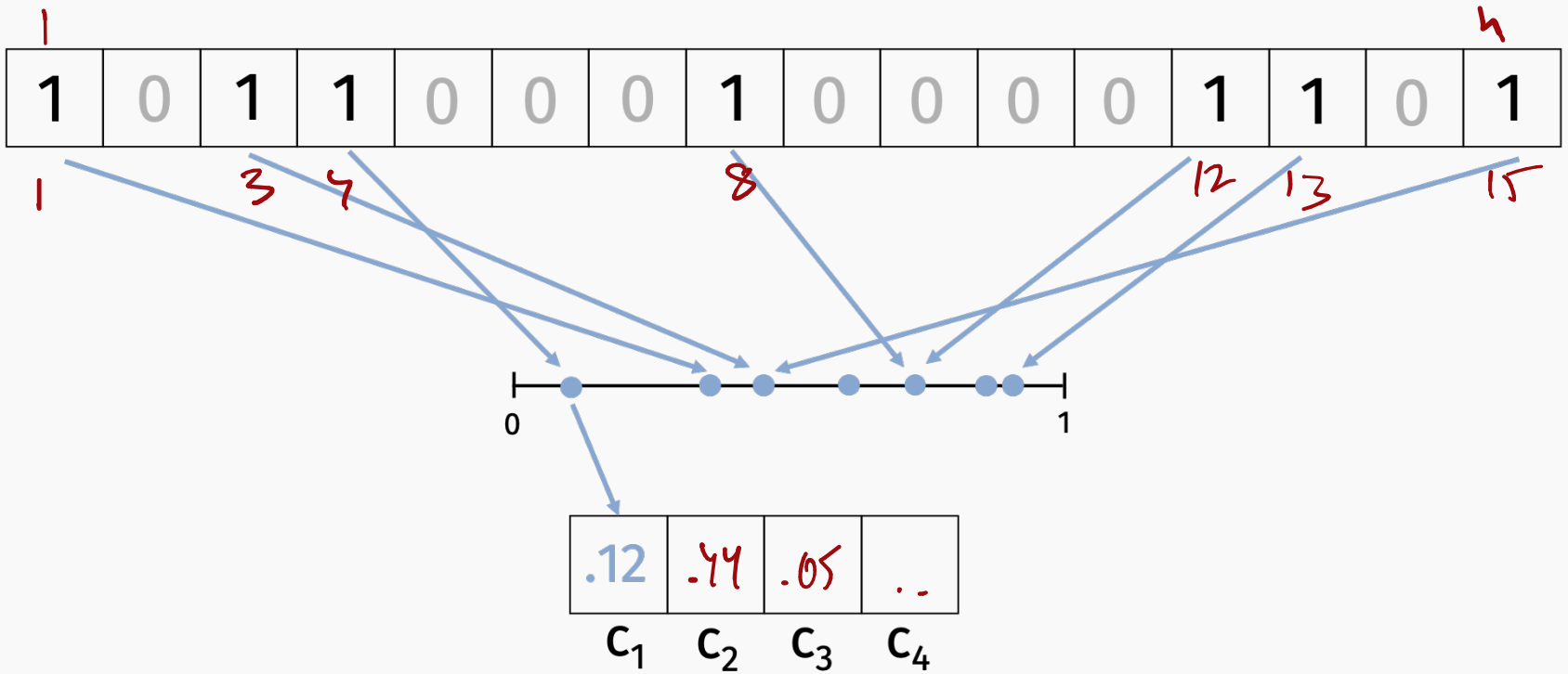
Homomorphic Compression: Want to use $C(\mathbf{q}), C(\mathbf{y})$ to approximately compute the Jaccard similarity $J(\mathbf{q}, \mathbf{y})$.

$\pi_q \quad \pi_y$

MinHash (Broder, '97):

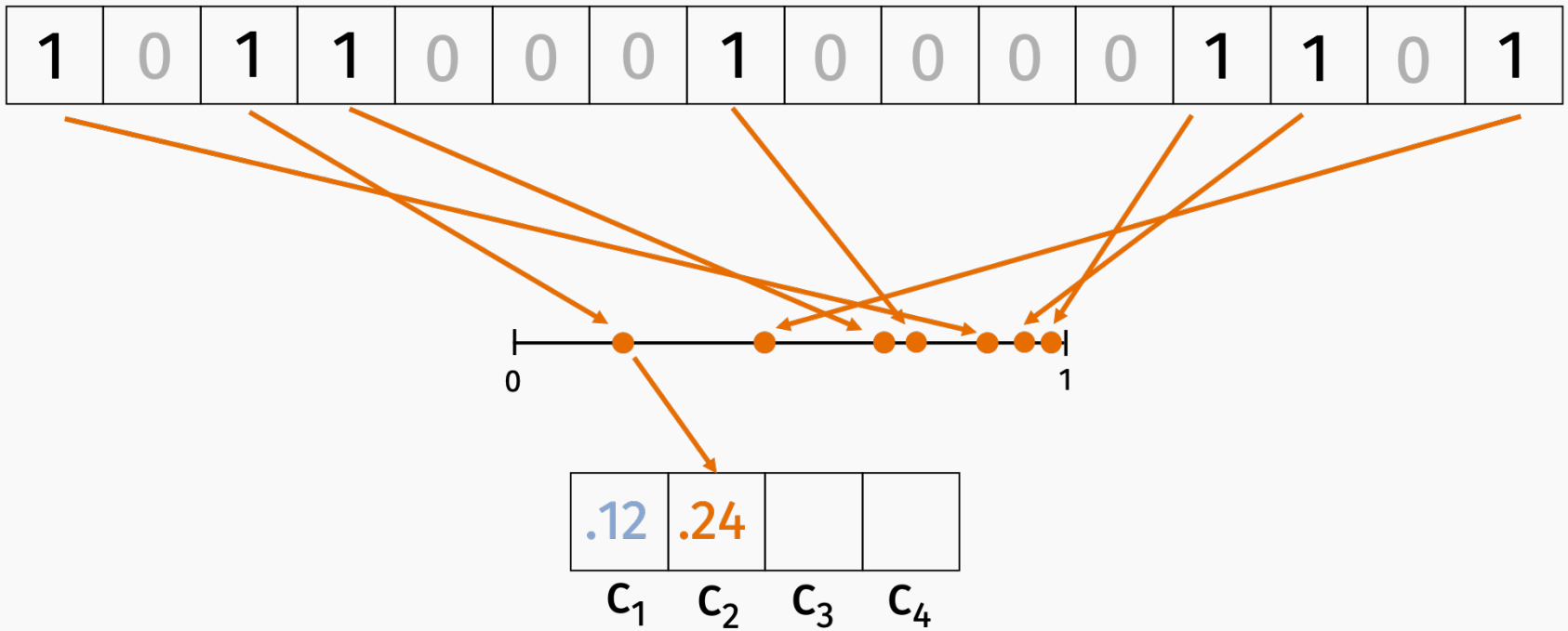
- Choose k random hash functions $h_1, \ldots, h_k : \{1, \ldots, n\} \to [0, 1]$.
- For $i \in 1, \ldots, k$,
  - Let $c_i = \min_{j, q_j = 1} h_i(j)$.
- $C(q) = [c_1, \ldots, c_k]$.

$h_1(1), h_1(3), h_1(4) \ldots h_1(15)$



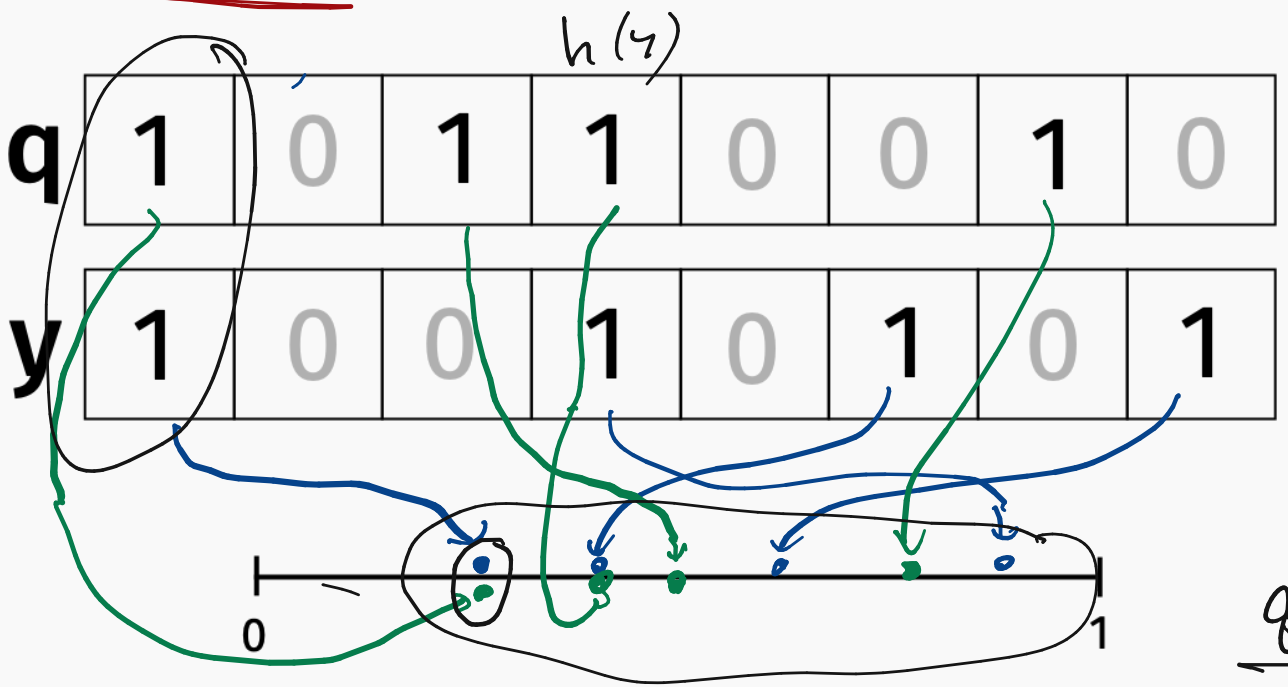
MINHASH

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



For all i,

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

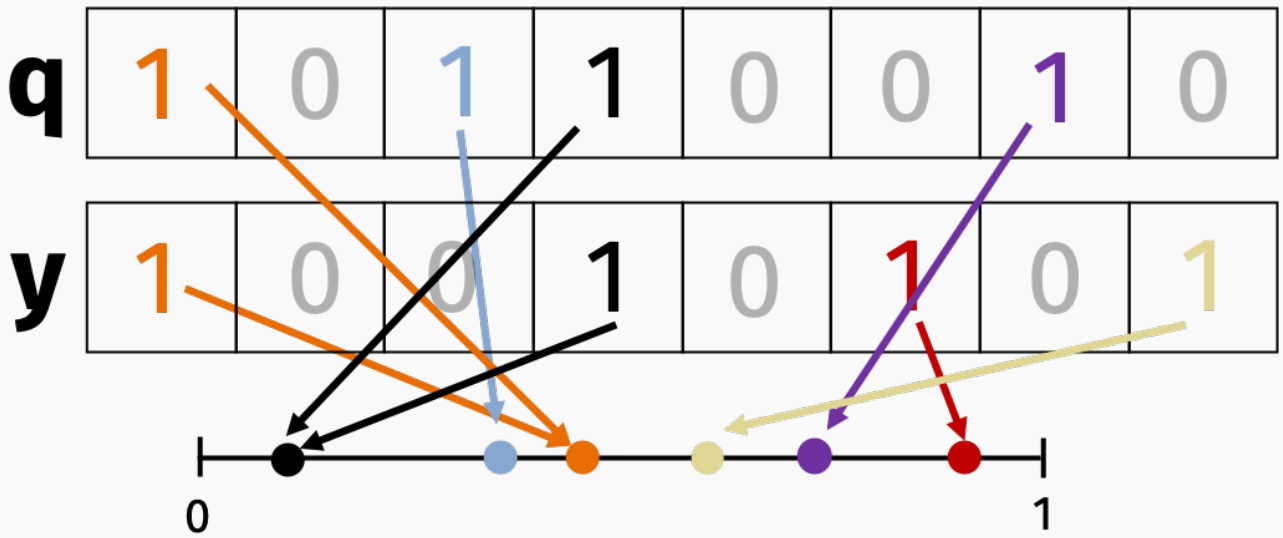


$\frac{q \cap y}{q \cup y} = J(q,y)$

Proof:

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

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 <u>both</u> $\mathbf{q}$ 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 $\mathbf{q}$ and $\mathbf{y}$.

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:

$$\mathbb{E}\tilde{J} = \frac{1}{k} \sum_{i=1}^{k} \mathbb{E}\,\mathbb{1}[\ldots] = \frac{1}{k} \sum_{i=1}^{k} J(\mathbf{q}, \mathbf{y}) = J(\mathbf{q}, \mathbf{y})$$

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

1 wp J

0 wp $1-J$

$$\text{Var}[\tilde{J}] = \frac{1}{k^2} \sum_{i=1}^{k} \text{Var}\left[\mathbb{1}[c_i(q) = c_i(y)]\right] = \frac{1}{k^2} \sum_{i=1}^{k} J - J^2$$

$$\leq \frac{1}{k} \cdot J \leq \frac{1}{k}$$

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

$\frac{1}{\sqrt{k}}$

$$\Pr\left[|\tilde{J} - J| > \epsilon\right] \leq \delta \qquad \alpha = \frac{1}{\sqrt{\delta}}$$

$$\Pr\left[|\tilde{J} - J| \geq q\sigma\right] \leq \frac{1}{q^2}$$

$$\Pr\left[|\tilde{J} - J| \geq \underbrace{\frac{1}{\sqrt{\delta}} \cdot \frac{1}{\sqrt{k}}}_{\epsilon}\right] \leq \delta$$

$$\boxed{k = \frac{1/\delta}{\epsilon^2}}$$

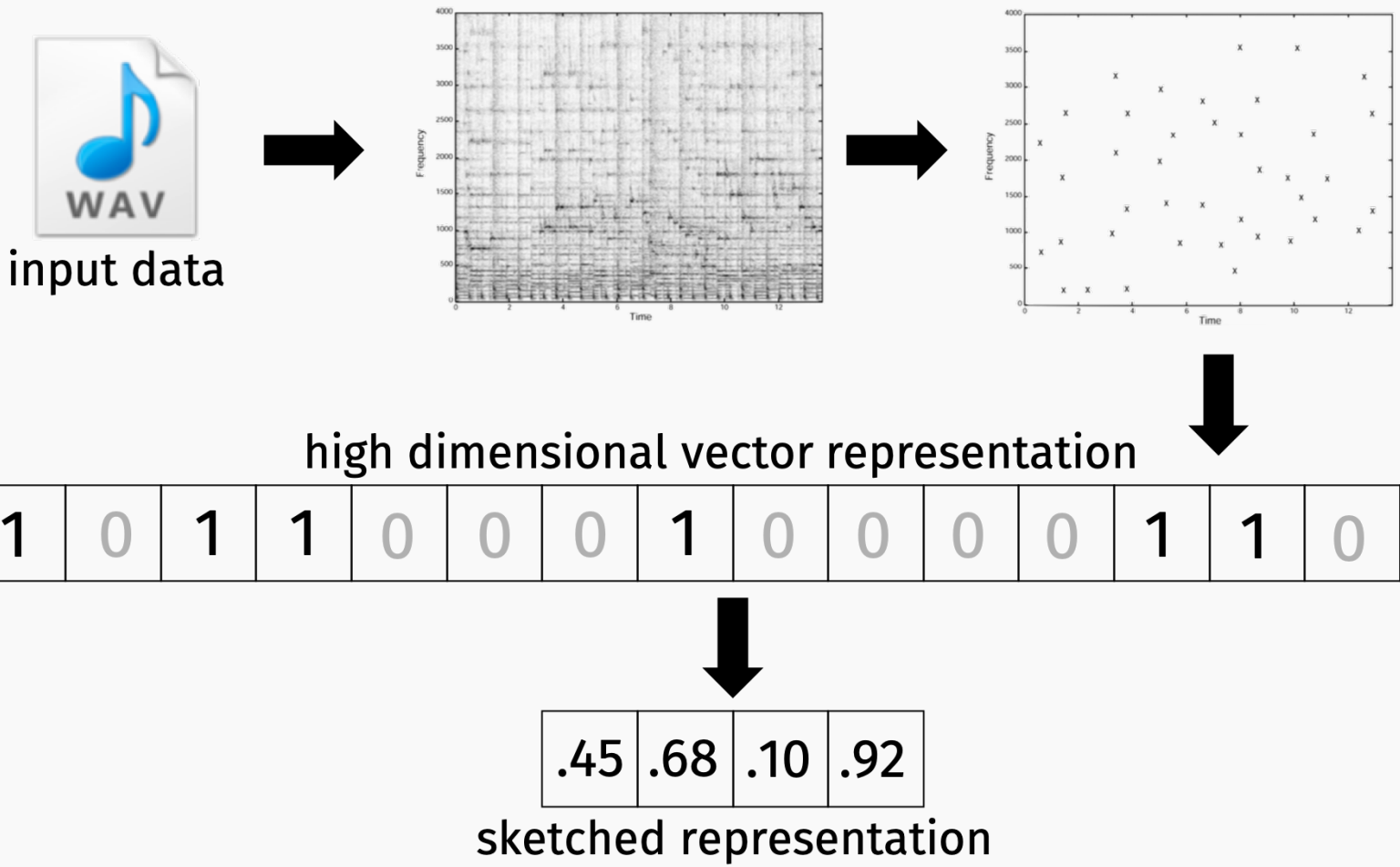
Chebyshev inequality: As long as $k = O\left(\frac{1}{\epsilon^2 \delta}\right)$, then with prob. $1 - \delta$,

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

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



BREAK

NEAR NEIGHBOR SEARCH

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.

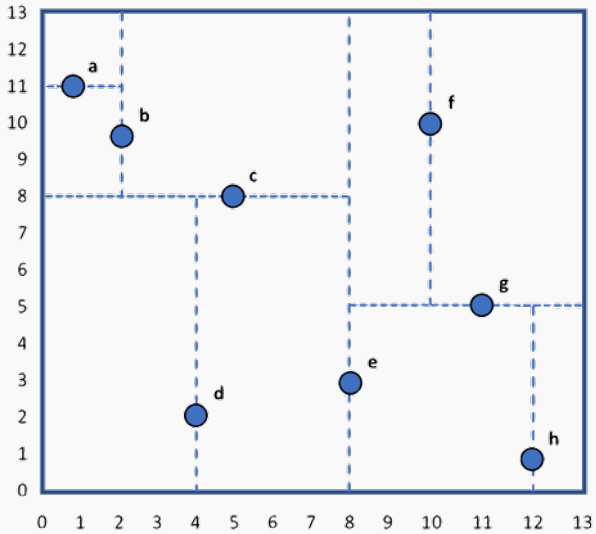
How does similarity sketching help in these applications?

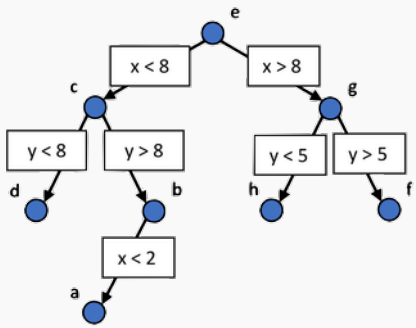
- Improves runtime of "linear scan" from $O(nd)$ to $O(nk)$.
- Improves space complexity from $O(nd)$ to $O(nk)$. This can be super important – e.g. if it means the linear scan only accesses vectors in fast memory.

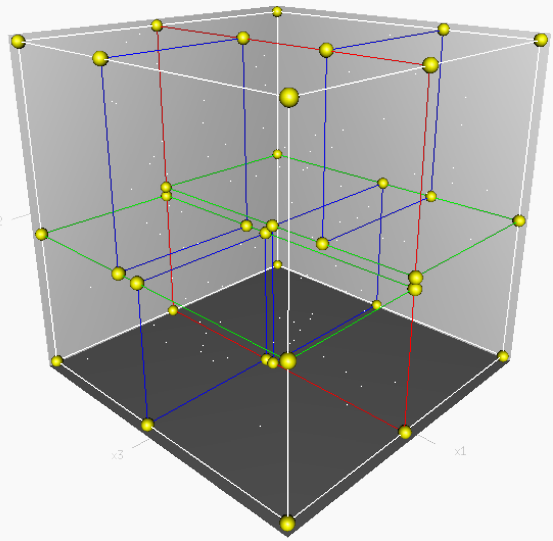
BEYOND A LINEAR SCAN

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]
  - Probably the most practical. This is most similar to the custom method e.g. Shazam uses.

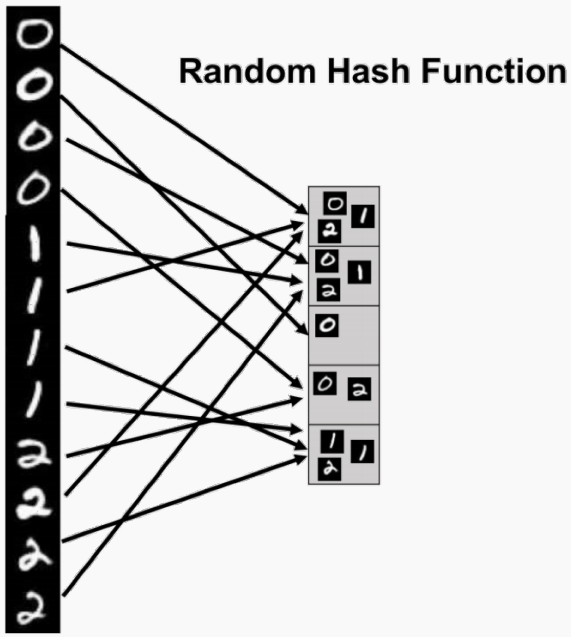
Key Insight: Trade worse space-complexity for better time-complexity.

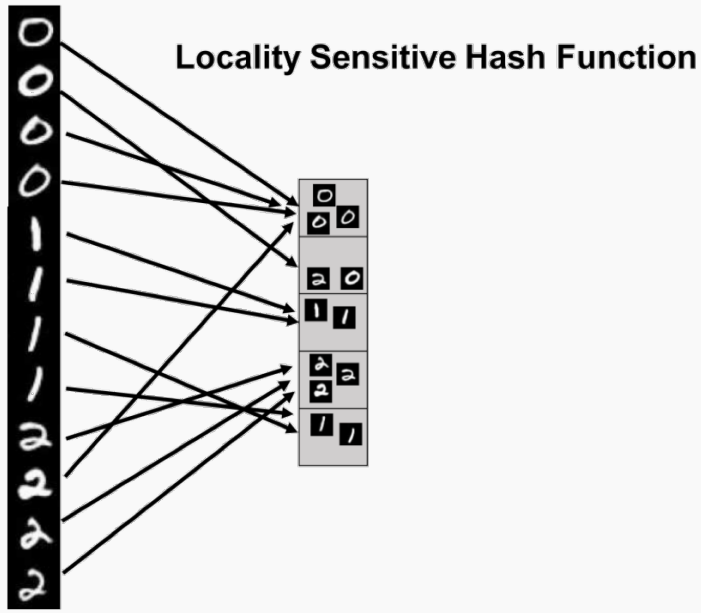
LOCALITY SENSITIVE HASH FUNCTIONS

Let $h : \mathbb{R}^d \to \{1, \ldots, m\}$ 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.

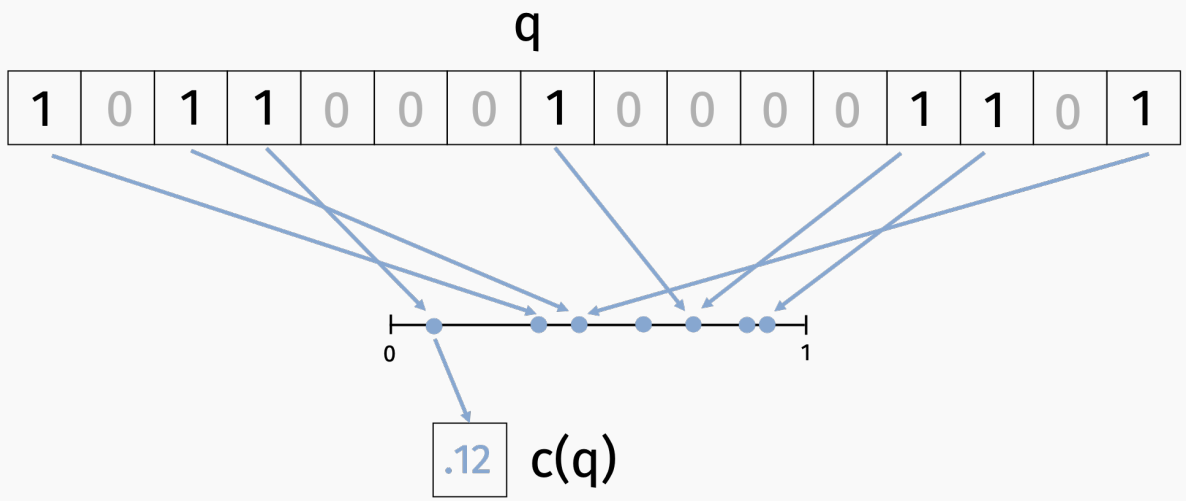




LOCALITY SENSITIVE HASH FUNCTIONS

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



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 uniform 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.

Pre-processing:

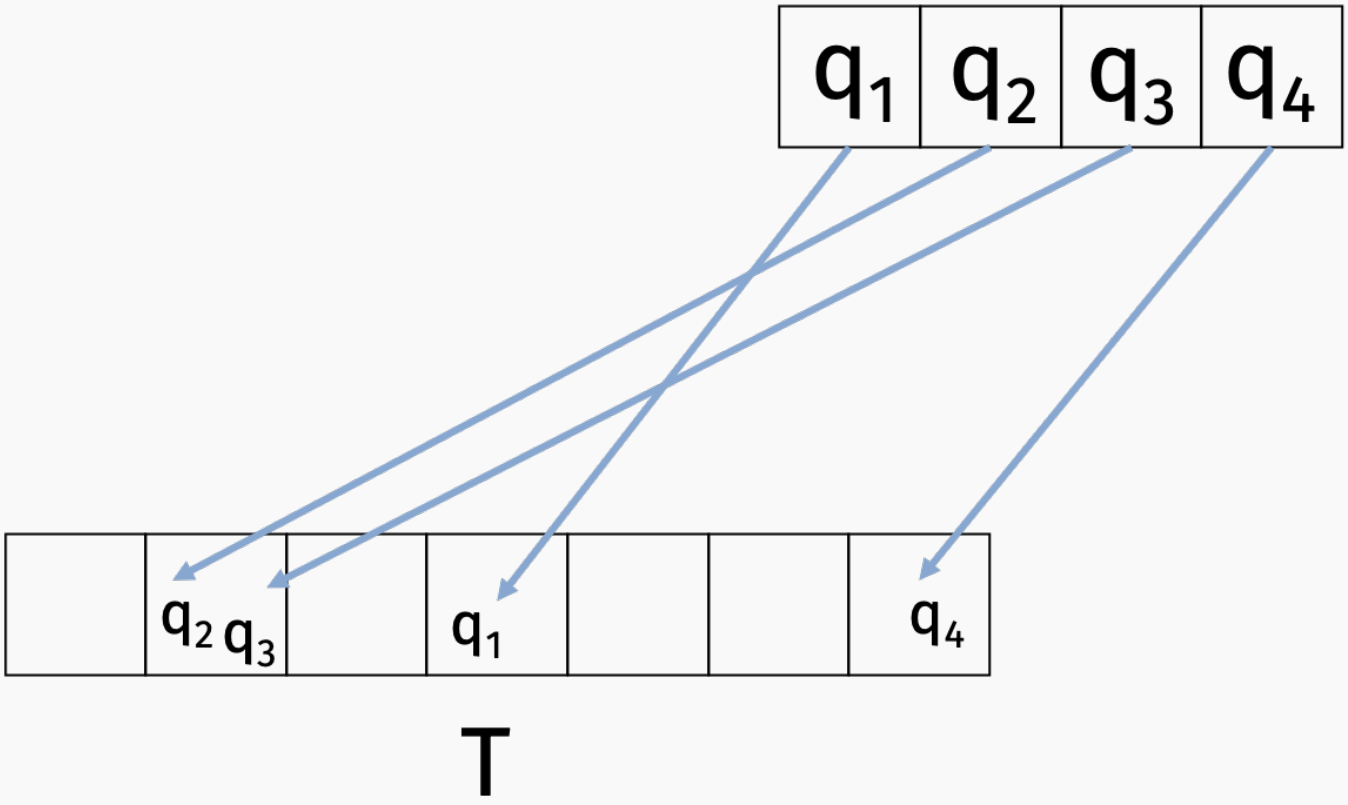
- Select random LSH function $h : \{0,1\}^d \to 1, \ldots, m$.
- Create table T with $m = O(n)$ slots.[1]
- For $i = 1, \ldots, n$, insert $\mathbf{q}_i$ into $T(h(\mathbf{q}_i))$.

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.

NEAR NEIGHBOR SEARCH



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 $\mathbf{y}$.

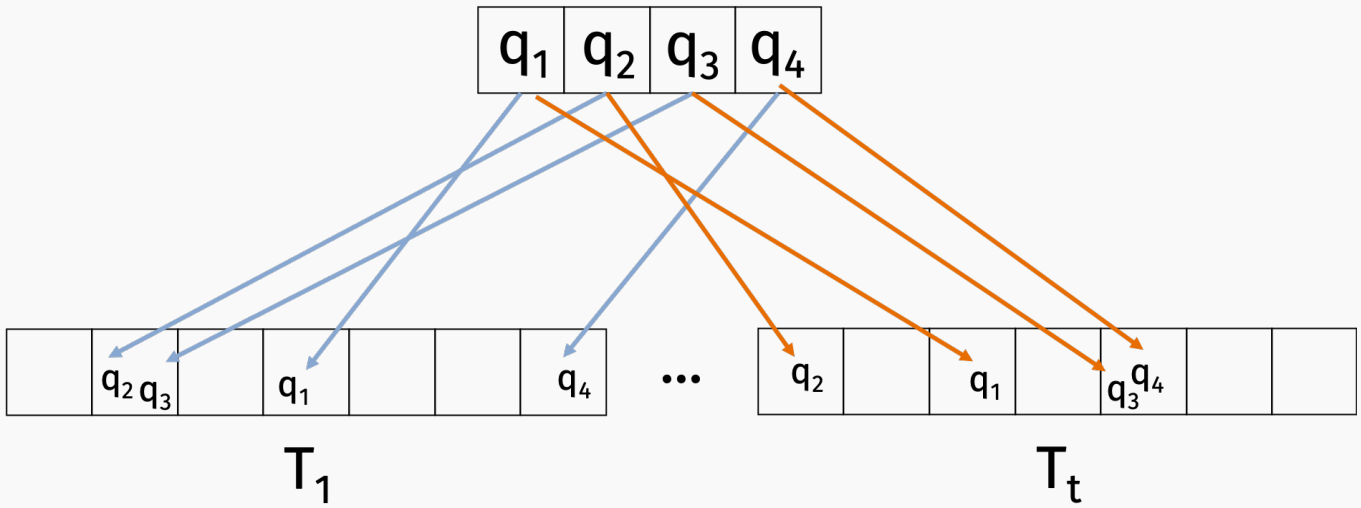
Note: The meaning of "close" and "not close" is application dependent. E.g. we might specify that we want to find anything with Jaccard similarity $> .4$, but not with Jaccard similarity $< .2$.

REDUCING FALSE NEGATIVE RATE

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

What's the probability we do not find q?

REDUCING FALSE NEGATIVE RATE



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 $T_j(h_j(\mathbf{q}_i))$.

REDUCING FALSE NEGATIVE RATE

±10
99%

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?

WHAT HAPPENS TO FALSE POSITIVES?

89%

Suppose there is some other database point $\mathbf{z}$ 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 $\mathbf{y}$ hashes into at least one bucket containing $\mathbf{z}$.

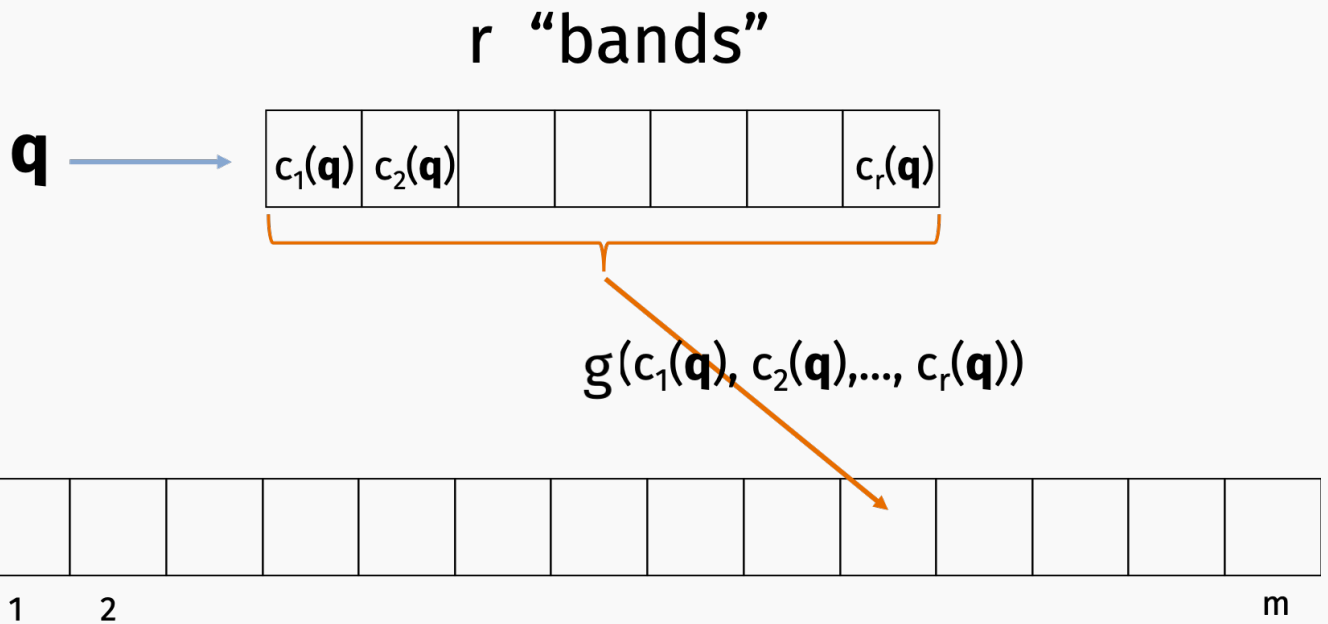
In the new scheme with $t = 10$ tables?

REDUCING FALSE POSITIVES

Change our locality sensitive hash function.

Tunable LSH for Jaccard similarity:

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



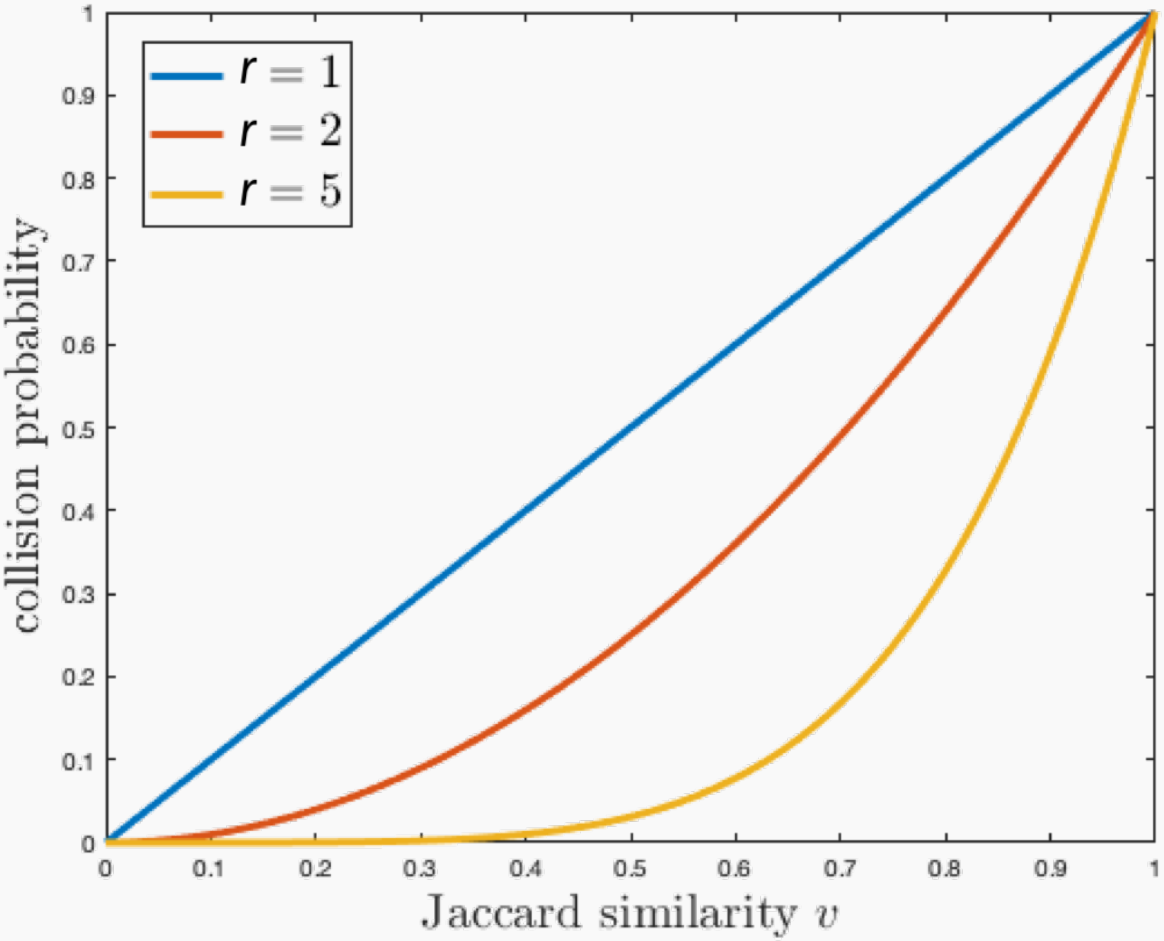
REDUCING FALSE POSITIVES

Tunable LSH for Jaccard similarity:

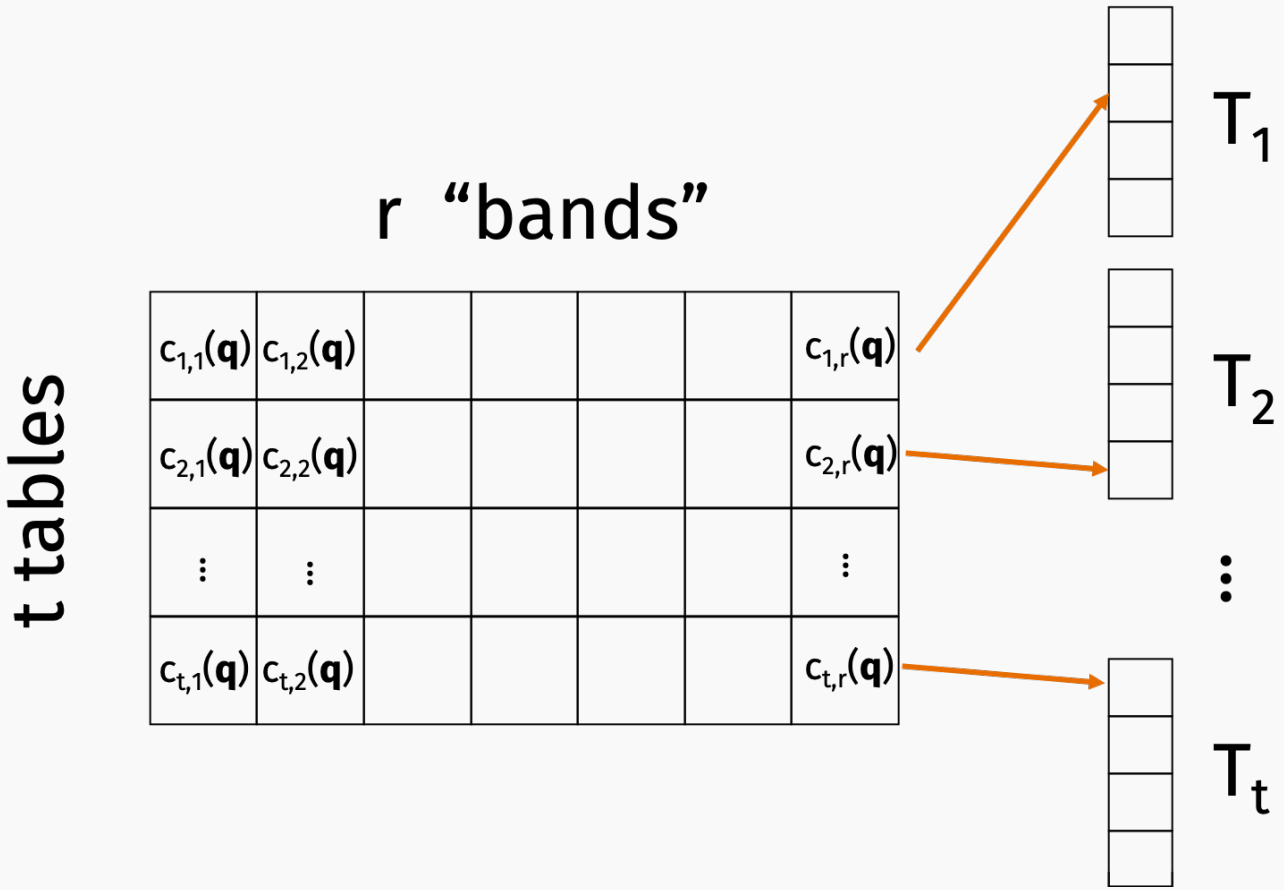
- Choose parameter $r \in \mathbb{Z}^+$.
- Let $c_1, \ldots, c_r : \{0,1\}^d \to [0,1]$ be random MinHash.
- Let $g : [0,1]^r \to \{1, \ldots, m\}$ be a uniform random hash function.
- Let $h(\mathbf{x}) = g(c_1(\mathbf{x}), \ldots, c_r(\mathbf{x}))$.

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

TUNABLE LSH


r = 1
r = 2
r = 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

Full LSH cheme has two parameters to tune:



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

False Negatives

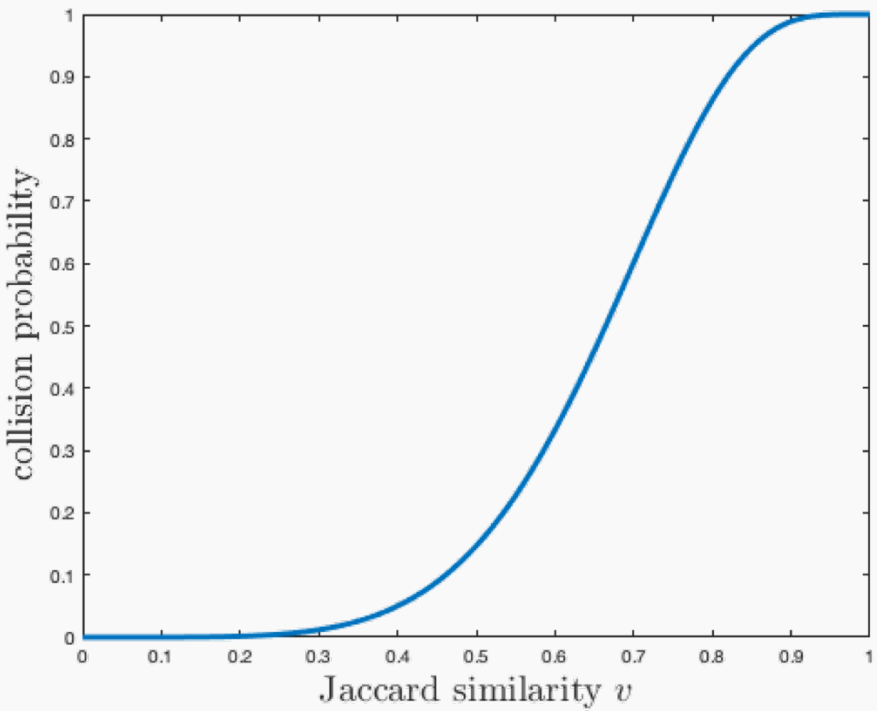
False Positives

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

False Negatives

False Positives

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

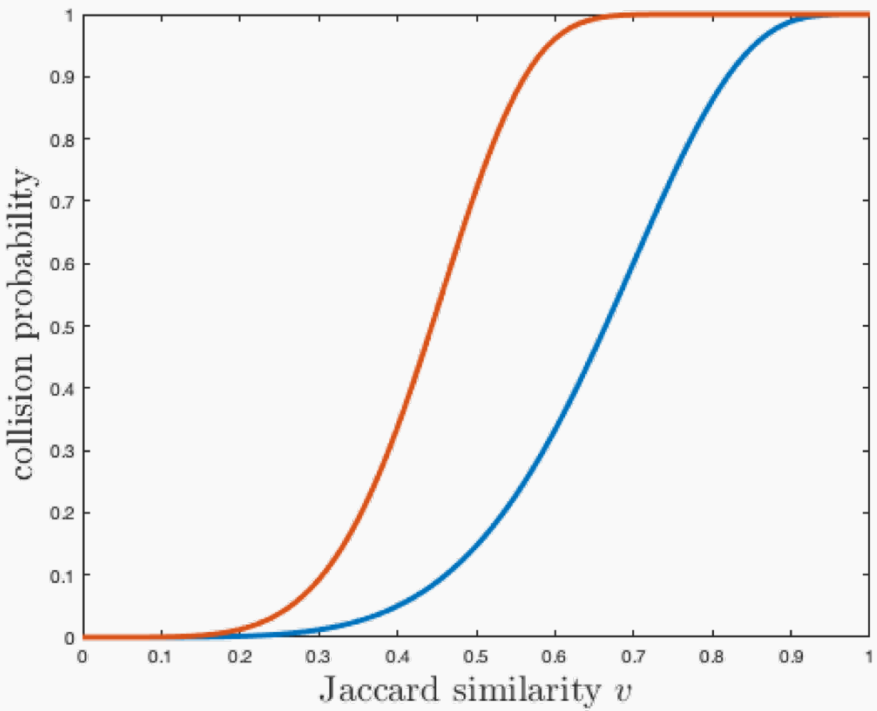


$r = 5, t = 5$

S-CURVE TUNING

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

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

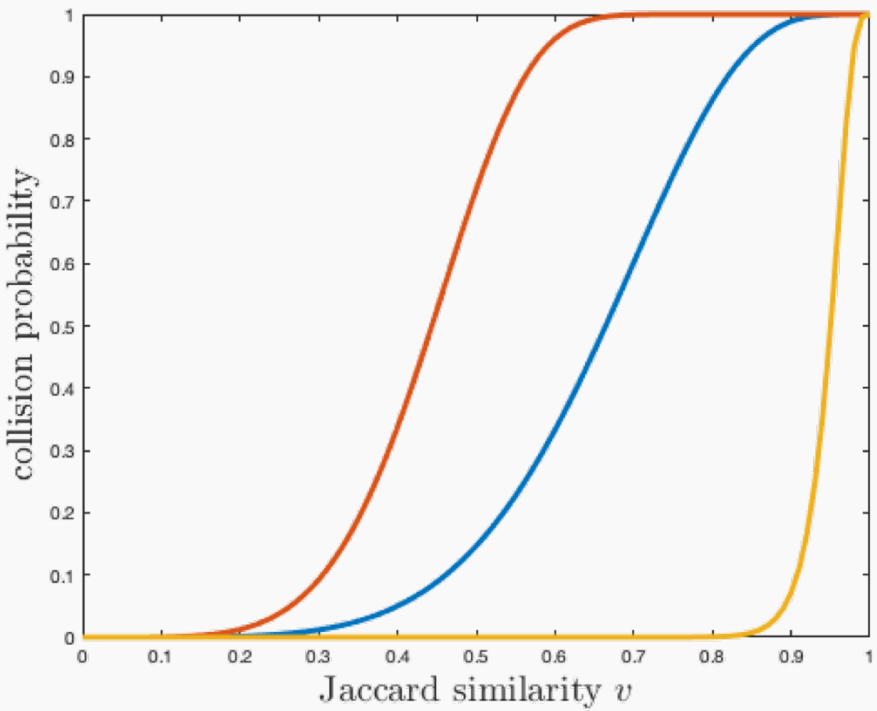


$$r = 5, t = 40$$

S-CURVE TUNING

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

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

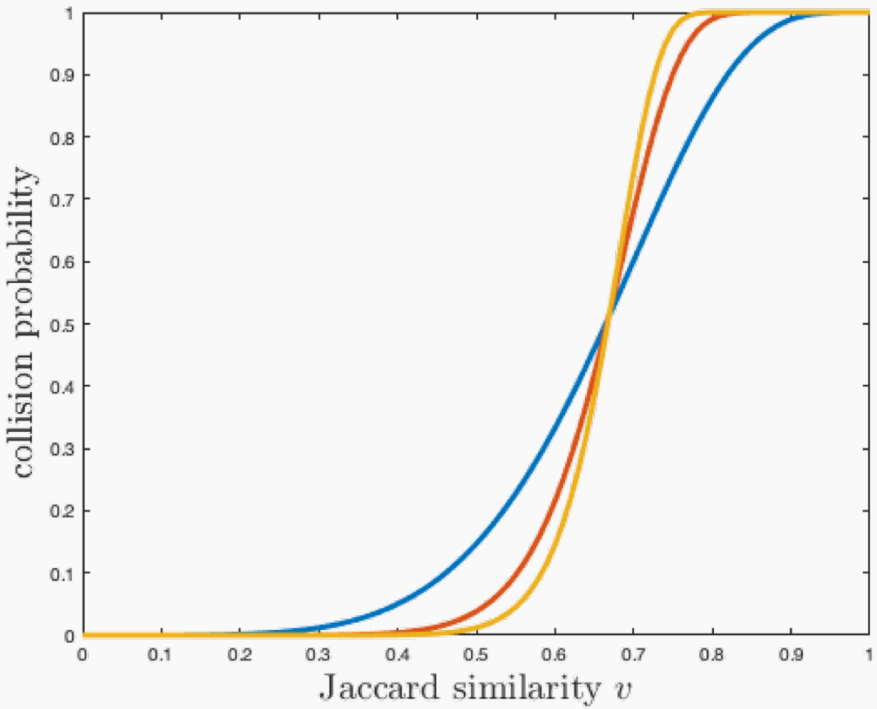


$$r = 40, t = 5$$

S-CURVE TUNING

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:

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

FIXED THRESHOLD

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 $\mathbf{y}$.

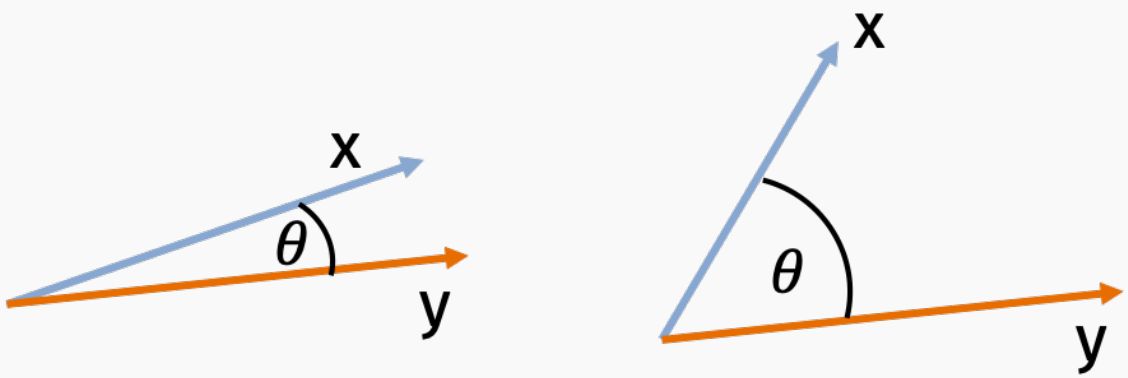
Theorem (Indyk, Motwani, 1998)

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



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

Cosine similarity is natural "inverse" for Euclidean distance.

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

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

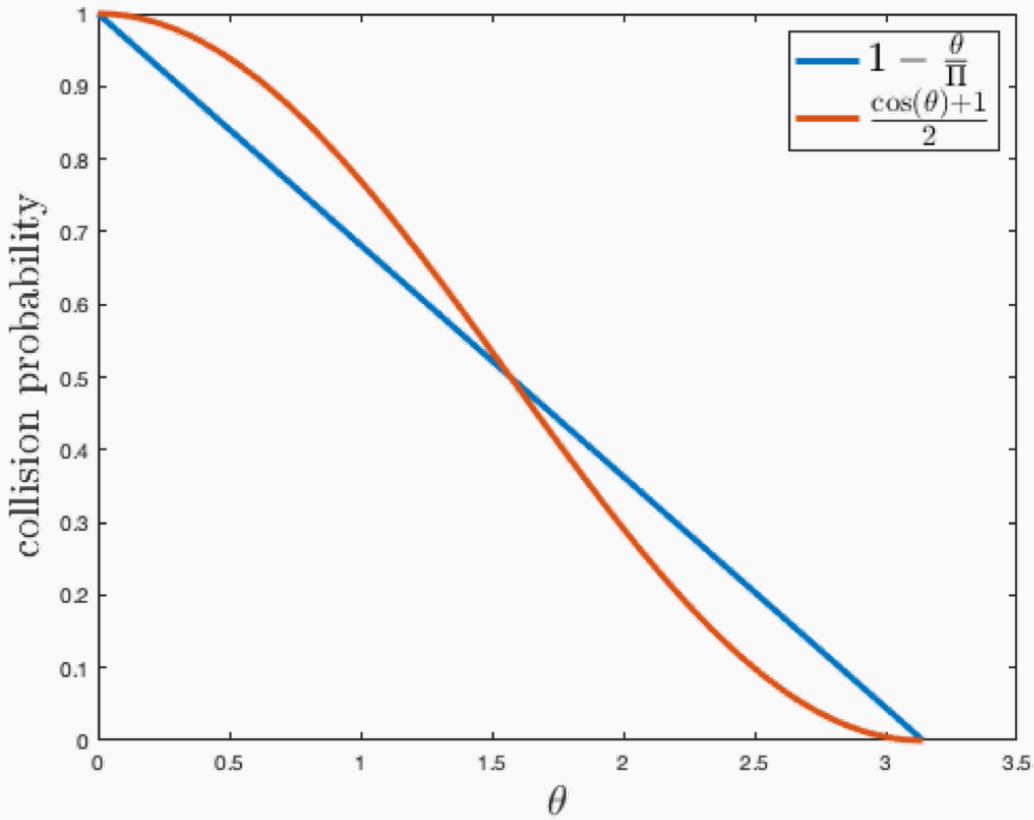
Locality sensitive hash for **cosine similarity**:

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

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

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

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



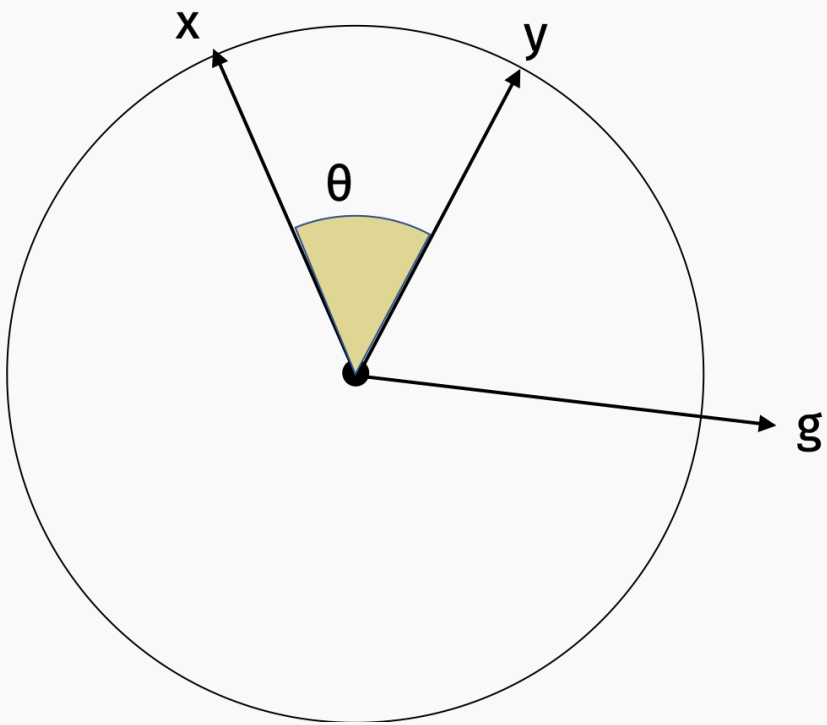
SimHash can be tuned, just like our MinHash based LSH function for Jaccard similarity:

- 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 $h(\mathbf{x}) = f([\mathsf{sign}(\langle \mathbf{g}_1, \mathbf{x} \rangle), \ldots, \mathsf{sign}(\langle \mathbf{g}_r, \mathbf{x} \rangle)])$.

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

To prove:

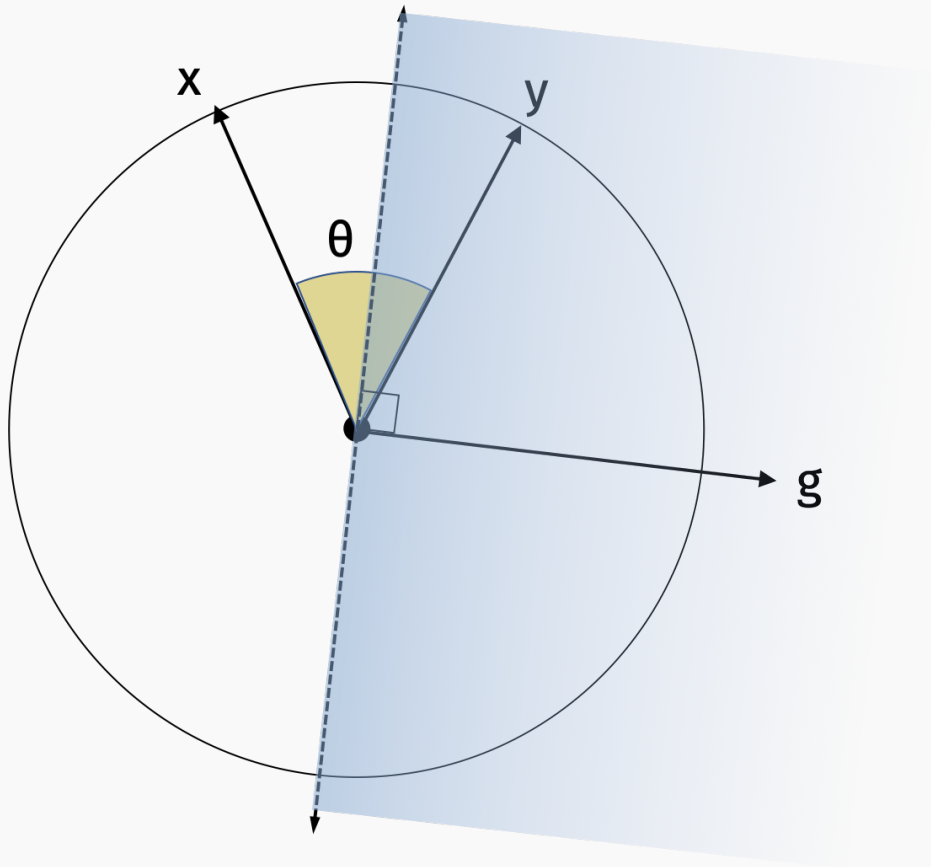
$\Pr[h(\mathbf{x}) == h(\mathbf{y})] = 1 - \frac{\theta}{\pi}$, where $h(\mathbf{x}) = f(\text{sign}(\langle \mathbf{g}, \mathbf{x} \rangle))$.



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

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

SIMHASH ANALYSIS



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