Ideas:
- Hashing (statistics)
- Concentration of measure
- Fast approximate linear algebra: SVD, Nyström approximation
- Fast algorithms: sparse matrices
- Error correcting codes

Analysis of data stream:
Example: network traffic monitoring, hot spots, denial of service attacks, usage-based pricing.
Problem: mid 90's: flow between two hubs ≤ 1 million/hr.

count packets: unrealistic!
DRAM too expensive. gain in speed ≈ 8%/yr.
gain in links ≈ 100%/yr.

Solution: random sampling of packets.
identify: large consumer with very few measure anomalies

→ track specific packets identified by specific header.

Example: denial of service:
- TCP packets
- Specific destination ID

anomalies: > 1% link capacity over a measurement interval 15.1min/1hr.
Random Projections Applied to Nearest Neighbor Search in High Dimension.

Introduction:

Database of \( N = 50 \) million digital fingerprints.

Query: find the closest match to a given fingerprint.

Each fingerprint: \( 10^5 \) features from a \( 10^6 \) image.

Match: \( 75\% \) feature overlap.

Problem: 50 million comparison in \( R^{10^6} \).

Idea 1: each fingerprint = vector in \( R^{10^6} \).

Explore \( R^{10^6} \).

Problem: search in high dimension is costly:

- Linear storage (as opposed to exponential), query time: \( n N \)

\[ N = \min(2^n, nN) \]

- Exponential storage: \( N^n \) space, query: \( n \log(N) \).

\( N = 5 \times 10^7 \)

\( n = 10^5 \)

All these numbers are large.

Idea 2: only a very small combination of the features is used.

Map the 50 million only to a small-tuple.
Define a hash function:

\[ h : \mathbb{R}^n \rightarrow \mathbb{R}^k \quad \text{for } k < n. \]

Problem: We may have \( h(x_i) = h(x_j) \) for two vectors \( x_i \neq x_j \); collision.

A good hash function should minimize collisions:

- If \( \| x_i - x_j \| \leq \delta \), then \( P(\hat{h}(x) = h(x_j)) \geq P_1 \)
- If \( \| x_i - x_j \| \geq c\delta \), then \( P(\hat{h}(x) = h(x_j)) \leq P_2 \)

Assume \( P_1 > P_2 \).

Algorithm for finding nearest neighbors:

1. Choose \( L \) functions \( g_e \) for \( e = 1 \ldots L \), \( g_e = (h_1^e, h_2^e, \ldots, h_k^e) \)
   
   \( h_\cdot^e \): random hash function from a family \( \mathcal{H} \)

2. Construct the corresponding hash tables

   \[ i = 1 \ldots N, \quad l = 1 \ldots L \]
   
   \[ G(i, l) = g_e(x_i) = (h_1^e(x_i), \ldots, h_k^e(x_i)) \]

Query point \( y \)

1. For \( l = 1 \ldots L \)
   
   - Find all \( x_i \) that have the same \( g_e(y) \).
   
   \[ g_e(y) = \begin{bmatrix} h_1^e(y) = h_1^e(x_i) \\ \vdots \\ h_k^e(y) = h_k^e(x_i) \end{bmatrix} \]

   - \( g_e(x_i) \) (bucket)
Note: There exists a fast algorithm to explore all the non-empty entries of the hash table. Also, the table does not use more than $NL$ memory.

- for all $x_i$ s.t. $g_e(x_i) = g_e(y)$ compute $||x_i - y||$.
- accept $x_i$ if this distance is small enough (less than $d$).

end for

Two strategies: stop after finding enough neighbors
- continue until all buckets are exhausted.

The algorithm returns the $c$-approximate $\ell$-near neighbors randomized.

Definition: Given a set $\mathcal{S}$ of points in $\mathbb{R}^n$ and $\ell > 0$, $\varepsilon > 0$, construct a data structure / an algorithm such that $\forall q \in \mathcal{S}$, if there exists $p \in \mathcal{P}$ with $||p - q|| < \varepsilon$, then the algorithm returns a $c\mathcal{S}$-near neighbor, $q'$ (maybe $q' = q$ but not necessarily).

Remarks: the number of hash functions $\ell$ ($\approx$ number of projections) $\nu \sim Np$, $p = \frac{\ln Vp}{\ln Vp}$, poly-storable.
\[ p < 1 \quad \text{because} \quad p_1 > p_2 \]

Example of binary vectors: \( p = \frac{1}{c} \quad c = 2 \)

Example: In Johnson-Lindenstrauss

\[ k \text{ is superconstant} \quad n \log n \]

Choice of the parameters:

- \( k \): if the hash functions are independent, and
- for each \( a \), \( P(h_a(x_i) = h_a(x_j)) \geq p_2 \) if \( \| x_i - x_j \| \leq \delta \)

Then:

\[ P(g_a^e(x_i) = g_a^e(x_j)) \geq p_1^k \quad \text{if} \quad \| x_i - x_j \| \leq \delta \]

Consider \( p \) in a \( \delta \)-neighbor of the query \( q \).

What is the probability that \( \exists \) at least one bucket \( e \) such that \( g^e(p) = g^e(q) \). The complement event is

\[ \forall e = 1 \ldots L, \quad g^e(p) \neq g^e(q) \]. Because the \( g^e \) are independent.

\[ P(\forall e = 1 \ldots L, \quad g^e(p) \neq g^e(q)) = \prod_{e=1}^{L} P(\neg g^e(p) = g^e(q)) \]

\[ \leq (1 - p_1^k)^L \]

Since \( P(g^e(p) = g^e(q)) \leq 1 - p_1^k \)

Therefore \( P(\exists e \in \{1 \ldots L\}, \quad g^e(p) = g^e(q)) \geq 1 - (1 - p_1^k)^L \)

If we choose \( \varepsilon = (1 - p_1^k)^L \), then with probability \( 1 - \varepsilon \) we can retrieve \( p \) as a neighbor.
To ensure that \((1 - p_k)^L\) is small we should choose \(L\) sufficiently large: 
\[
L = \log_{1-p_k} \varepsilon
\]

Also if \(k\) is large, we have less collisions: \(P_2^k\), but \(L\) needs to grow. \(\varepsilon\) small \(1 - p_k \approx 1\) as \(k\) so \(L\) needs to be large.

Examples of good hash functions

All good hash functions work because of Johnson-Lindenstrauss

1. i) Choose a random vector on \(S^{n-1}, \mu\).
   ii) Project the points on the 1-dimensional line defined by \(\mu\).
   iii) Randomly choose an origin on the line and divide into segments of size \(w\).
   iv) The hash function the index of the segment:

\[
\hat{h}_{\mu, b}(x) = \lfloor \langle x, \mu \rangle - b \rfloor / w \rfloor
\]

where \(b \sim \text{Uniform}(0, w)\).

2. Project the points on a randomly chosen subspace of dimension \(k\)

Projector: \(Q = \text{PHD}\)

\[D: n \times n, \text{ diagonal with entries } \pm 1 \text{ with probability } \frac{1}{2}\]
$H$: $n \times n$ Hadamard

$H_{ij} = \frac{1}{\sqrt{n}} (-1)^{\langle i-1, j-1 \rangle}$

$\langle i-1, j-1 \rangle$: dot product of $i-1$, $j-1$ expanded in binary.

$H$: DFT on the group $(\mathbb{Z}/2\mathbb{Z})^n$

Applying $H$ can be done in $n \log(n)$ computations, even if $H$ is $m^2$.

$P$: $P_{ij} = 0$ with probability $1 - q$

$q = \min(1, \frac{\log^2 n}{n})$

$\sim N(0, 1/q)$

$k \times n$: very sparse matrix.

Approximate nearest neighbor algorithm (Ailon, Chazelle 06)

query time: $O(n \log n + \varepsilon^2 \log^2 N)$

Storage $O(1/\varepsilon^2) (\varepsilon^2 \log N) \log(n)$

$c = 1 + \varepsilon$.

$H = \frac{1}{\sqrt{2}} \begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix}$ Haar.

$H_{2k+1} = \frac{1}{\sqrt{2}} \begin{bmatrix} H_{2k} & H_{2k} \\ H_{2k} & -H_{2k} \end{bmatrix}$: same as Haar wavelet packets.
Example of hash function

\[ \{0,1\}^n \rightarrow \{0,1\}^{p_i} \]

\[ p = p_1 \ldots p_p, p_0 \rightarrow R_i(p) = p_i : \text{bit } i \]

Binary representation

\[ \mathcal{H} : \text{family of hash functions}, \]

choose \( \mathcal{H} \) at random.

\[ P_{\mathcal{H}}(R_i(p) = r(q)) = \text{measure of} \{ \mathcal{H} \in \mathcal{H} : \text{such} \ R(p) = R(q) \}. \]

Here:

\[ h(p) = h(q) \text{ if } p \text{ and } q \text{ have the same bit } i \]

We need a distance on \( \{0,1\}^d \):

\[ \text{Hamming distance} : d(p,q) = \# \text{ bits where } p_i \neq q_i. \]

So \( d(p,q) \leq R \Rightarrow P_{\mathcal{H}} = q_i : \text{for at least } n-R \text{ bits}. \]

If we pick a bit at random across all the \( d(p,q) \), the probability that

uniformly over the \( d(p,q) \),

the bit is the same for \( p \) and \( q \) is

\[ \geq \frac{n-R}{n} = 1 - \frac{R}{n} \]

So

\[ d(p,q) \leq R \Rightarrow P_{\mathcal{H}} \left( \exists \mathcal{H} : R(p) = R(q) \right) \geq 1 - \frac{R}{n}. \]

By the same argument

if \( d(p,q) \geq cR \Rightarrow P(h \in \mathcal{H} : R(p) = R(q)) \leq 1 - \frac{R}{n}. \]
If we choose $c > 1$ then $P_i < P_2$.  

**Example 2** (generalization of the line construction).

$\mathbb{R}^n$ with $l_1$ distance $l_1 (x - y) = \| x - y \|_1 = \sum |x_i - y_i|$

- Choose $\text{W} \gg \delta$
- Construct a tiling of the space with cells of width $\text{W}$ in any direction; randomly shift the tiling.

Each cell defines a bucket

Choose randomly: $S_1, S_2, \ldots, S_n \in [0, w)$ uniformly distributed.

$\{S_1, \ldots, S_n\} = \left( \left\lfloor \frac{x_1 - S_1}{w} \right\rfloor, \ldots, \left\lfloor \frac{x_n - S_n}{w} \right\rfloor \right)$

**Example 3**

$(\mathbb{R}^n, l_2)$ $x, y \in \mathbb{R}^n$  $\theta(x, y) = \cos \left( \frac{\langle x, y \rangle}{\| x \| \| y \|} \right)$

Pick $x \in S^{n-1}$ at random.

$h_x (p) = \text{sign} (\langle x, p \rangle)$.

$P( h_x (p) = h_x (q) ) = P( \mu \in S^{n-1}, h_x (p) = h_x (q) ) = 1 - \frac{\theta(x, y)}{\pi}$
Example 4.

$S^{n-1}$

- Regular polytope, inscribed into $S^{n-1}$
- Map $x \in S^{n-1}$ to the closed polytope vertex.
- Bucket = Voronoi cell of the polytope vertices by ray on the sphere.