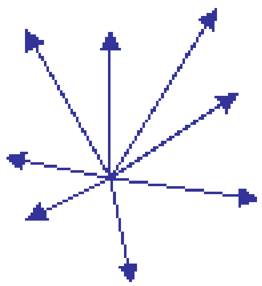
The Fast Johnson-Lindenstrauss Transform and Applications

Nir Ailon
Institute for Advanced Study
nailon@math.ias.edu

Joint with Bernard Chazelle

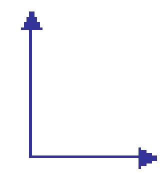
Dimension Reduction

- Algorithmic metric embedding technique



$$(R^d, L_p) \rightarrow (R^k, L_q)$$

$$k \ll d$$



- Distances after map $\approx_\varepsilon$ distances before map
- Useful when time/space depends heavily on d

- **Removes redundancy from data**

Dimension Reduction Applications

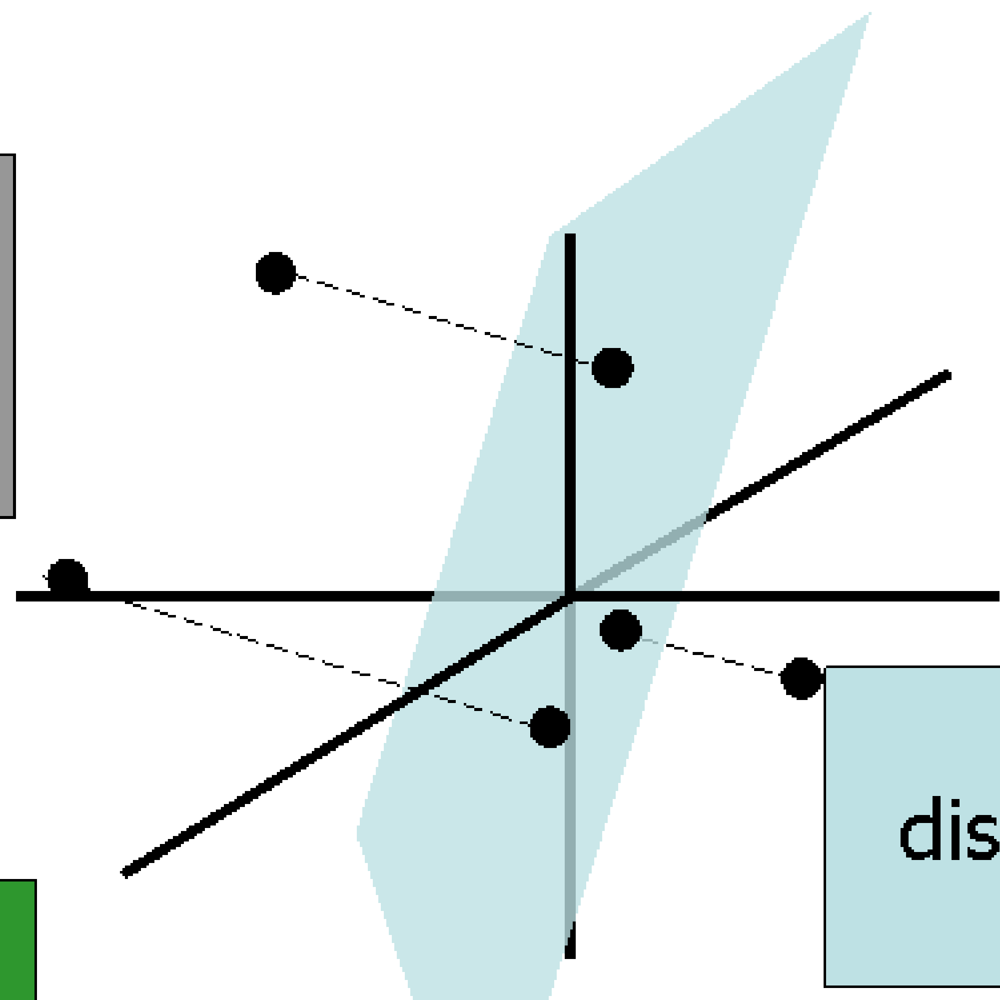
- **Approximate nearest neighbor** [KOR00, IM98]…
- Text analysis [PRTV98]
- Clustering [BOR99, S00]
- Streaming [I00]
- Linear algebra [DKM05, DMM06]
  - Matrix multiplication
  - SVD computation
  - L_2 regression
- VLSI layout Design [V98]
- Learning [AV99, D99, V98] . . .

History of Johnson-Lindenstrauss Dimension Reduction (1984-2005)

n points in
d dimensions

PROJECT

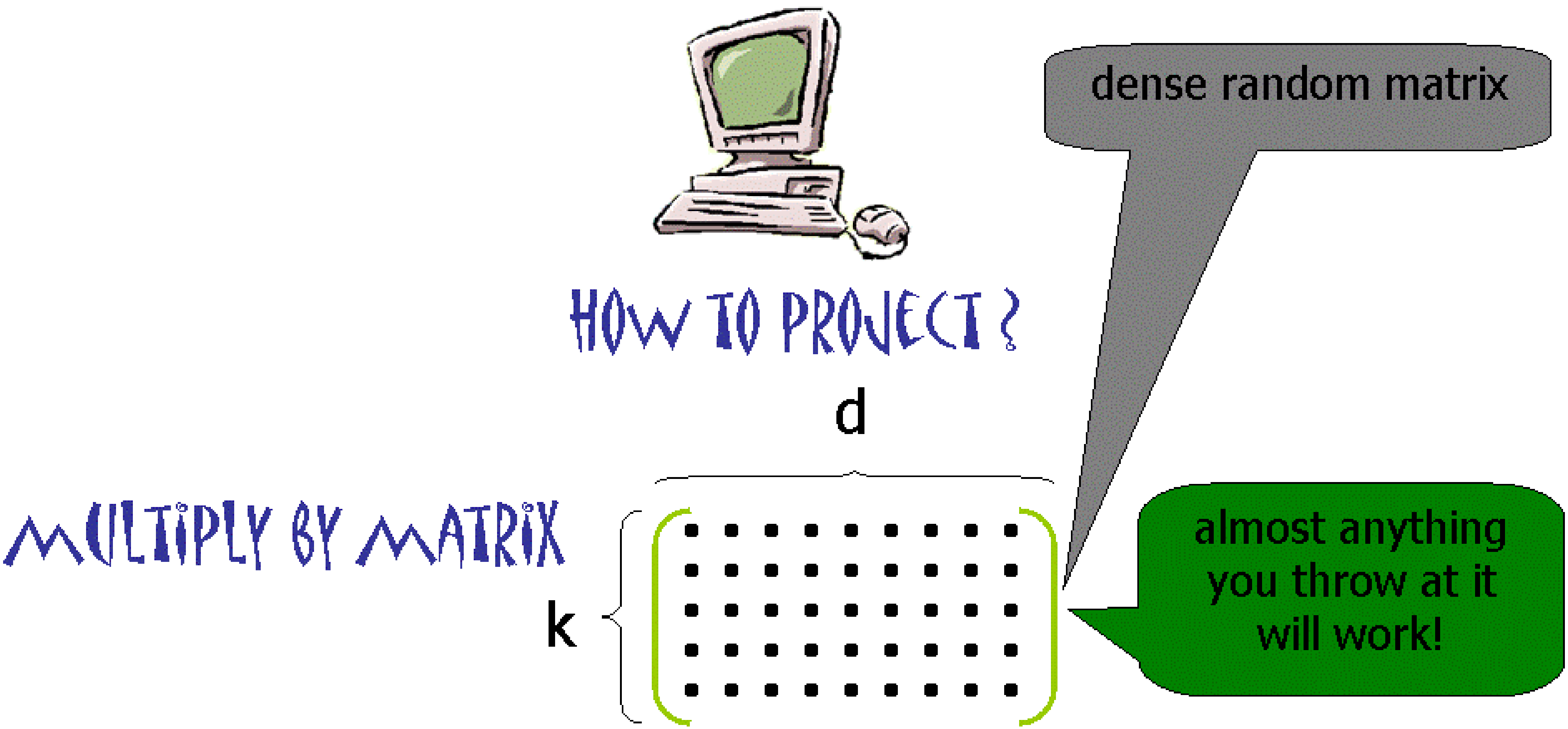
n points in
k dimensions



distances ε-preserved

choose random subspace
dimension $k=O(\varepsilon^{-2}\log n)$

History of Johnson-Lindenstrauss Dimension Reduction (1984-2005)



History of Johnson-Lindenstrauss Dimension Reduction (1984-2005)

orthogonal unit vector rows [JL84]

non-orthogonal unit vectors [DG99]

gaussians [IM98]

±1's [A03]

it's all the same asymptotically

running time= Ω(kd)

Optimality

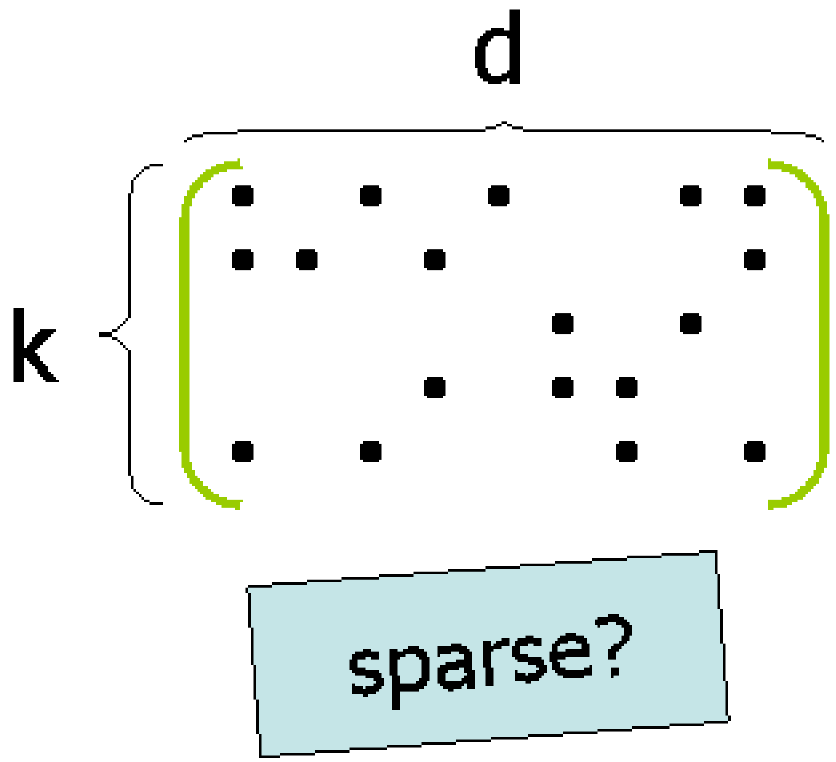
cannot have target dimension

$$k = o\left(\frac{\log n}{\varepsilon^2 \log \varepsilon^{-1}}\right)$$

how many linear operations/random bits necessary?

…to reduce a single vector

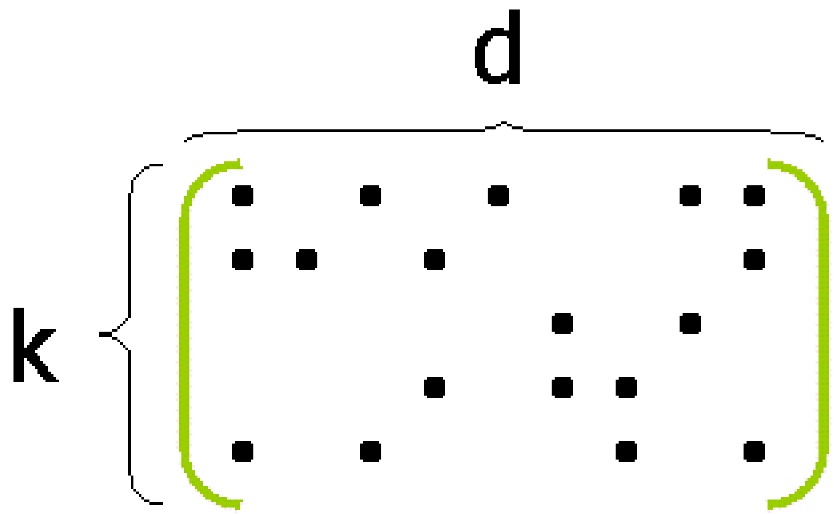
Sparse Projections



[Bingham, Mannila 2001]
Works in practice

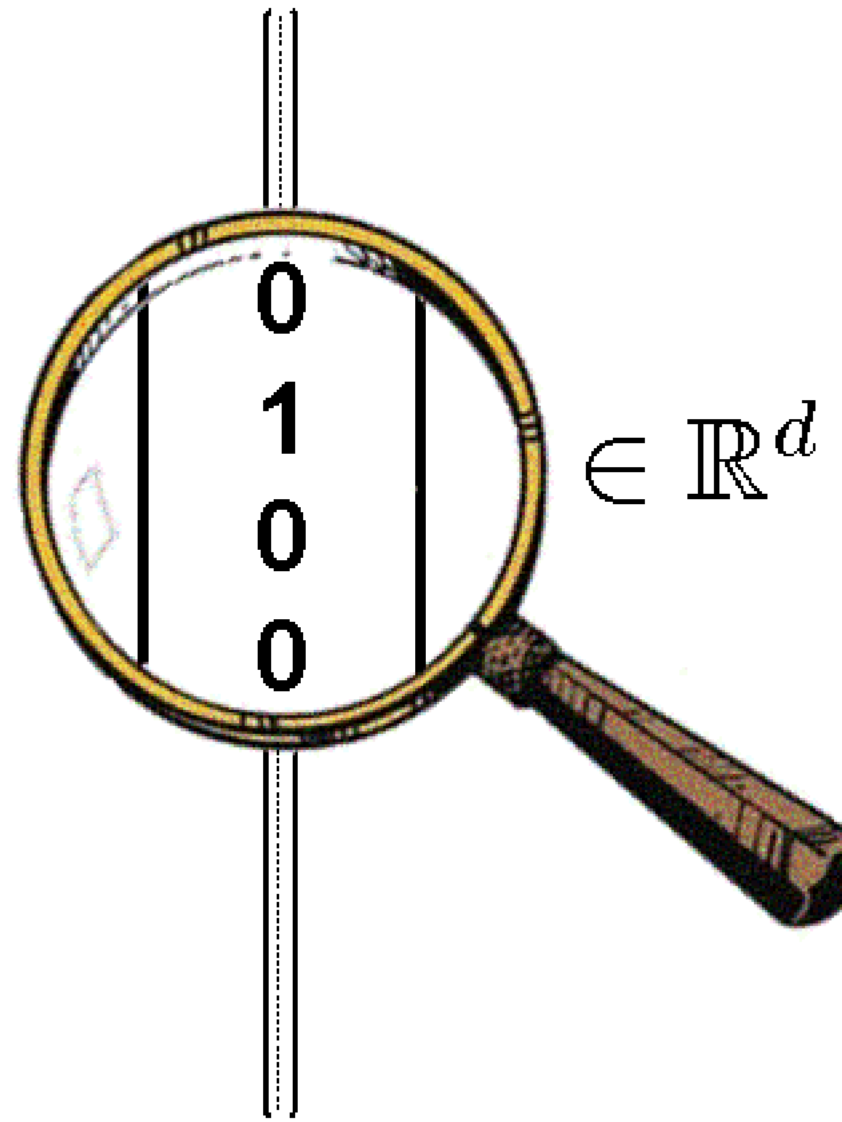
Works for “random” data

Sparse Projections



sparse?

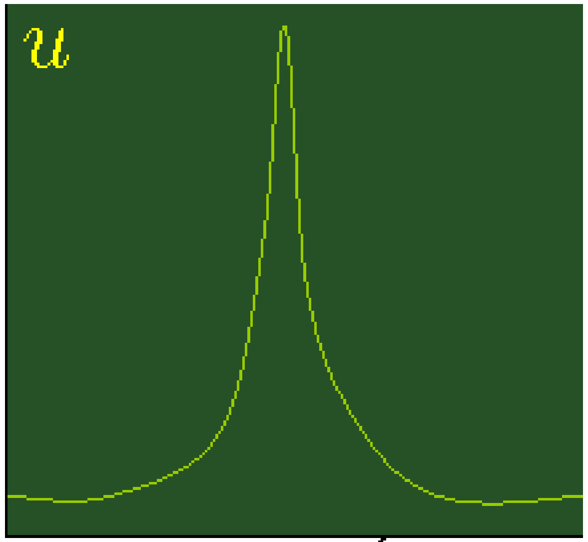
Not if I can help it!
If you don't know
where the 1 is hiding
you'll have to increase k
as matrix becomes
sparser...



$\in \mathbb{R}^d$

Harmonic Analysis

$u \in \mathbb{R}^d$. Some facts:

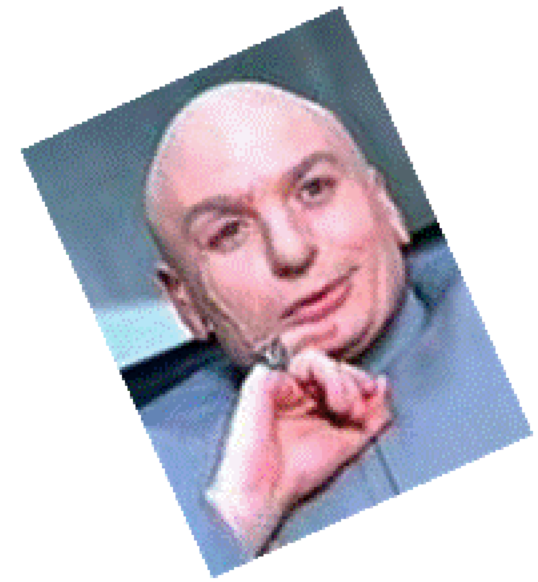


- $u \mapsto \hat{u}$ linear
- $\|\hat{u}\|_2 = \|u\|_2$
- u sparse $\Rightarrow \hat{u}$ dense (uncertainty)
- $\hat{u}$ computable in $O(d \log d)$ time (FFT/Walsh transform)

Must Add Randomness

Intuition fails because

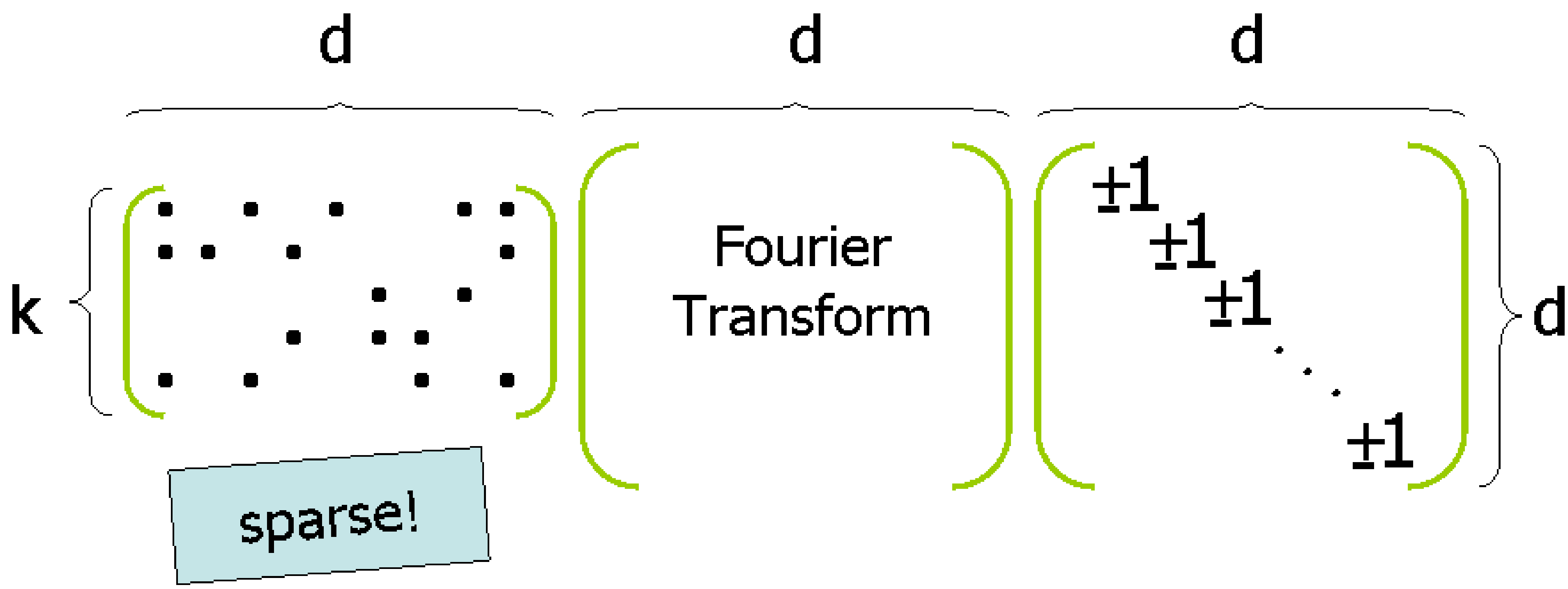
$u \mapsto \hat{u}$ rigid transformation



Solution: take $\widehat{Du}$ where

$$D = \begin{pmatrix} \pm 1 & & & \\ & \pm 1 & & \\ & & \ddots & \\ & & & \pm 1 \end{pmatrix}$$

Fast J-L Transform (FJLT)



	$L_2 \to L_1$	$L_2 \to L_2$
naive	$\Omega\left(\frac{d \log n}{\varepsilon^2}\right)$	$\Omega\left(\frac{d \log n}{\varepsilon^2}\right)$
FJLT	$O\left(d \log d + \frac{\log^2 n}{\varepsilon^3}\right)$	$O\left(d \log d + \frac{\log^3 n}{\varepsilon^2}\right)$
when FJLT better?	$n = e^{O(d\varepsilon)}$	$n = e^{O(\sqrt{d})}$

subsumed by $k = \frac{c \log n}{\varepsilon^2} = O(d)$

Improved to d log log n

Proof: Worst Case is Hidden Coordinate Set

$$\widehat{Du} = (\underbrace{c, \ldots, c}_{\Theta(\frac{d}{\log n})}, 0, \ldots, 0)$$

I can no longer hide all information in one coordinate. I can only shuffle the coordinates.

Proof of FJLT ($L_2 \to L_1$)

Assume $\widehat{Du} = (\underbrace{c, \ldots, c}_{d/\log n}, 0, \ldots, 0)$ with $c = \sqrt{\log n / d}$

$\Rightarrow \Phi_i \widehat{Du} \approx \sum_{j=1}^{d/\log n} g_j b_j c$ where $\begin{array}{l} g_j \sim N(0,1) \\ b_j \sim \text{Bernoulli}(s) \end{array}$

$\Rightarrow (\Phi \widehat{Du} | \sum b_j = b) \sim N(0, bc^2)$

$\Rightarrow E[|\Phi_i \widehat{Du}|] = E_b[|N(0, bc^2)|] \approx E_b[c\sqrt{b}] = cE[\sqrt{\text{Bin}(d/\log n, s)}]$

Want $E[\sqrt{\text{Bin}}] \approx_\varepsilon \sqrt{E[\text{Bin}]} \Rightarrow$ must have $E[\text{Bin}] = \Omega(1/\varepsilon)$

$\Rightarrow sd/\log n \approx \varepsilon^{-1} \Rightarrow s \approx \frac{\log n}{d\varepsilon}$

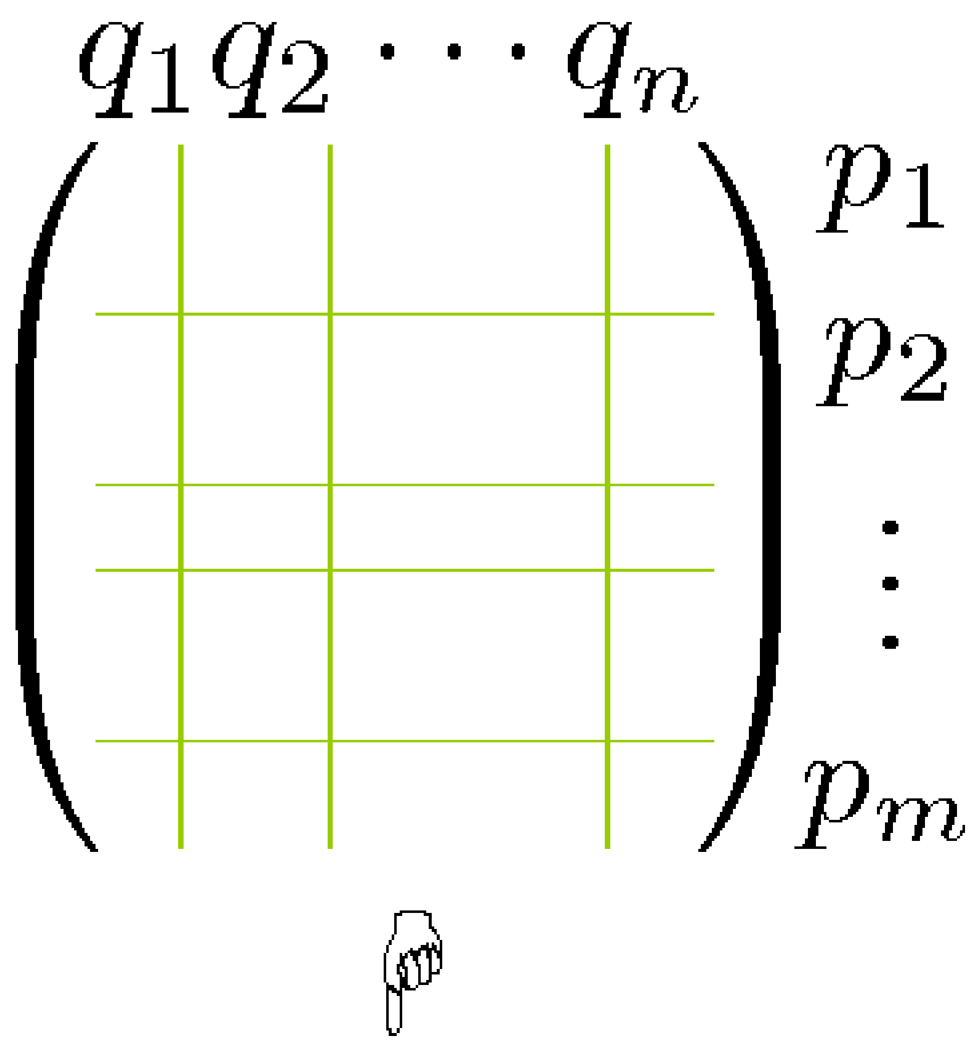
Applications

Nearest Neighbor Searching

	$L_2 \to L_1$	$L_2 \to L_2$
naive	$\Omega\left(\frac{d \log n}{\varepsilon^2}\right)$	$\Omega\left(\frac{d \log n}{\varepsilon^2}\right)$
FJLT	$O\left(d \log\log n + \frac{\log^2 n}{\varepsilon^3}\right)$	$O\left(d \log\log n + \frac{\log^3 n}{\varepsilon^2}\right)$
when FJLT better?	$n = e^{O(d\varepsilon)}$	$n = e^{O(\sqrt{d})}$

Scientific Computation

$A =$



$\tilde{A}$ = row/column sampling approximation

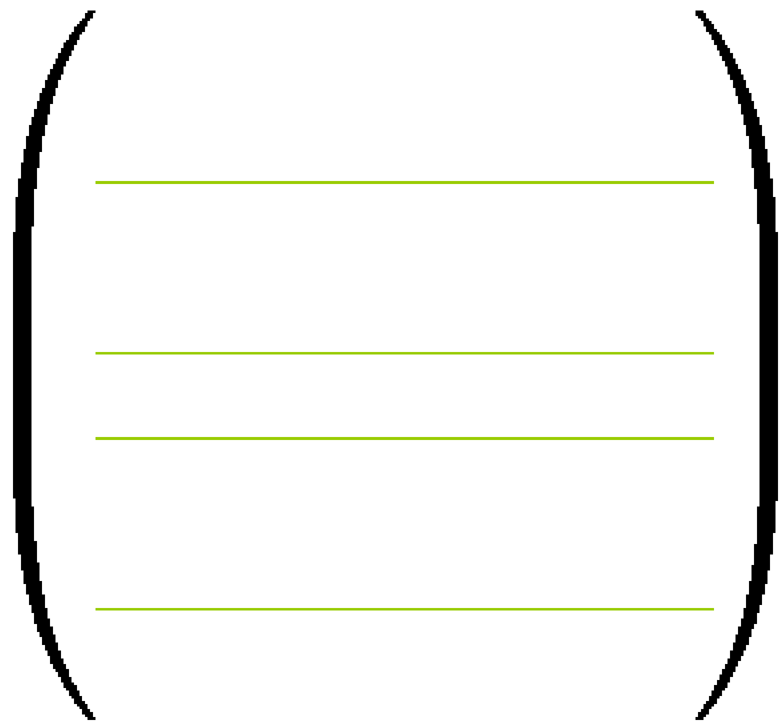
[Drineas, Kannan, Mahoney, Muthukrishnan . . .]

Scientific Computation

- Matrix-matrix multiplication
- Matrix-vector multiplication
- L_2 regression [S06, DS07]
- SVD computation

- Advantage:
  Sparsity of matrix preserved
- Disadvantage:
  Probabilities depend on matrices/vectors

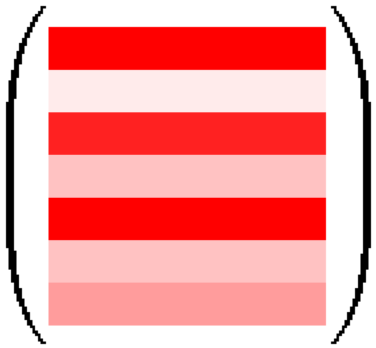
Scientific Computation



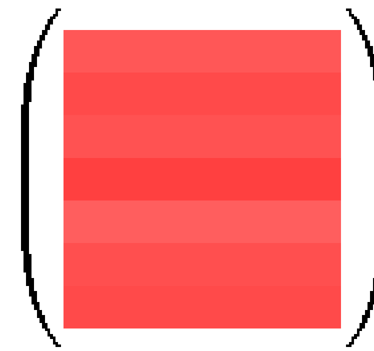
$$p_1 \sim \|A_1\|_2^2$$
$$p_2 \sim \|A_2\|_2^2$$
$$\vdots$$
$$p_m \sim \|A_m\|_2^2$$

- Streaming: requires 2 passes on large A
- Av_1, Av_2...: requires recomputing $p_1..p_m$

Scientific Computation



$$A \rightarrow \widehat{DA}$$



now sample..

- Spreads information "evenly"
- Can take $p_1..p_m$ uniform
- Naive: costs O(m log m) time
- Better: costs O(m log s) time
  s = number of rows to sample
- Requires one pass
- Does not depend on matrices/vectors

no cost in accuracy

little cost in time

sampling-friendly

Main Theoretical Open Problem

- What is fastest J-L transform?
- What is a J-L transform?

????

(to be continued…)

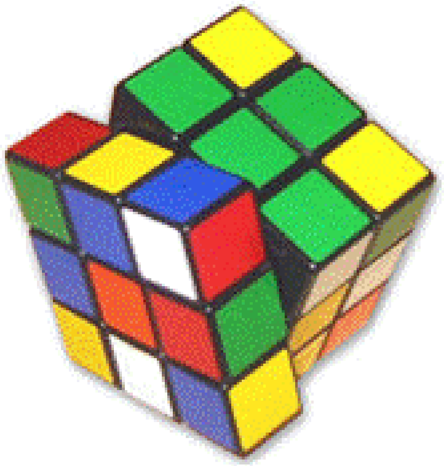
- What is most elegant J-L transform?

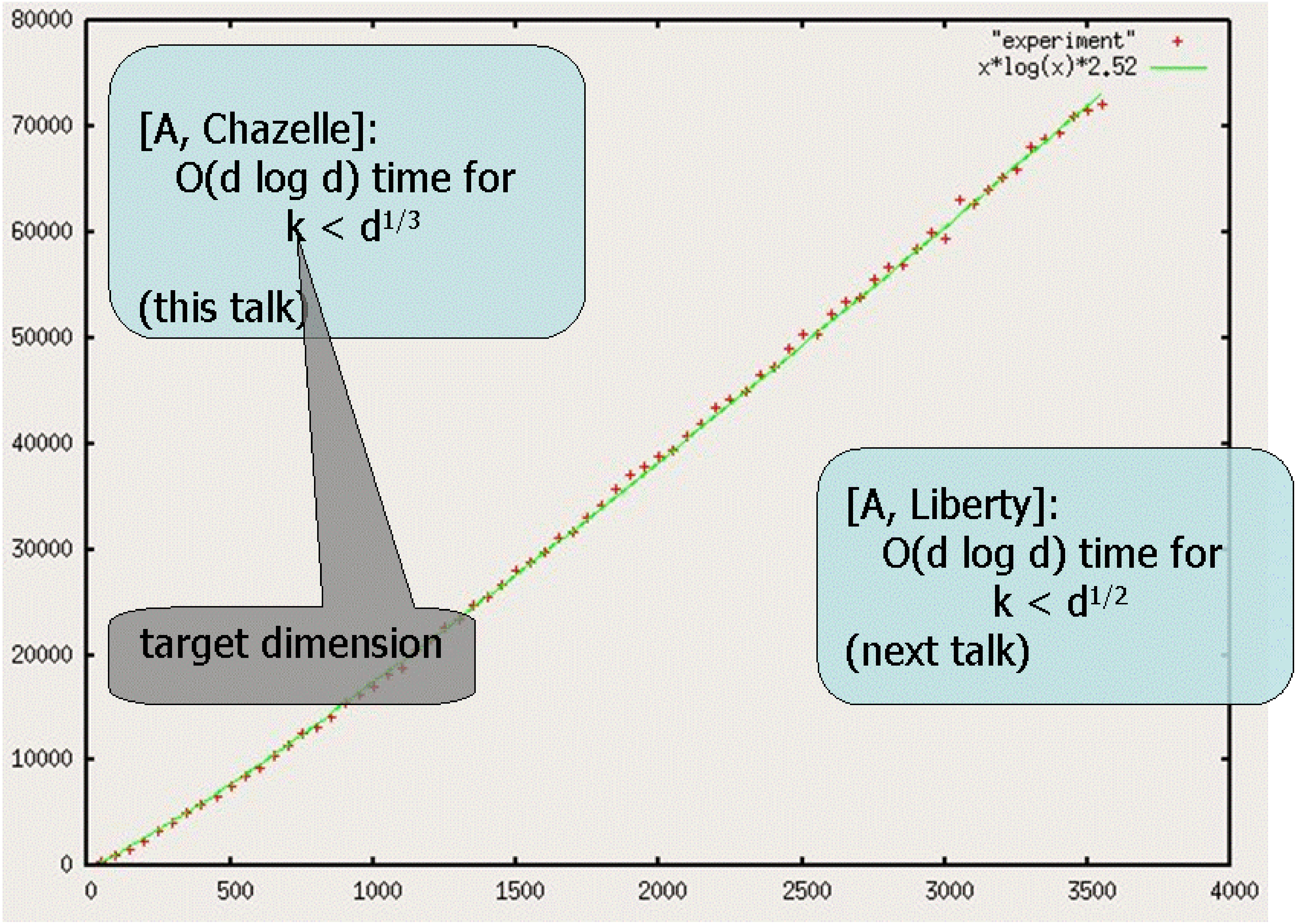
  repeat T times:

  pick random i, j

  pick random angle Θ

  rotate (x_i, x_j) by Θ





[A, Chazelle]:
O(d log d) time for
k < d^{1/3}
(this talk)
target dimension
[A, Liberty]:
O(d log d) time for
k < d^{1/2}
(next talk)
"experiment"
x*log(x)*2.52
80000
70000
60000
50000
40000
30000
20000
10000
0
0
500
1000
1500
2000
2500
3000
3500
4000