When Hashes Met Wedges: A Distributed Algorithm for Finding High Similarity Vectors

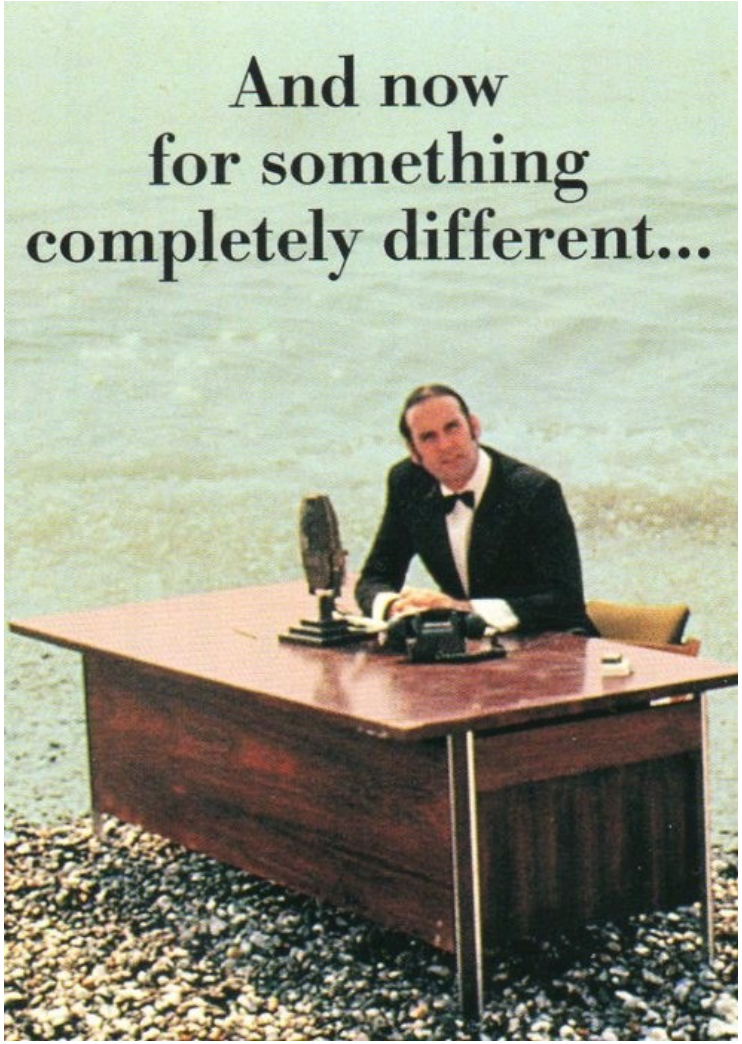
Aneesh Sharma
Google

C. Seshadhri
UC Santa Cruz

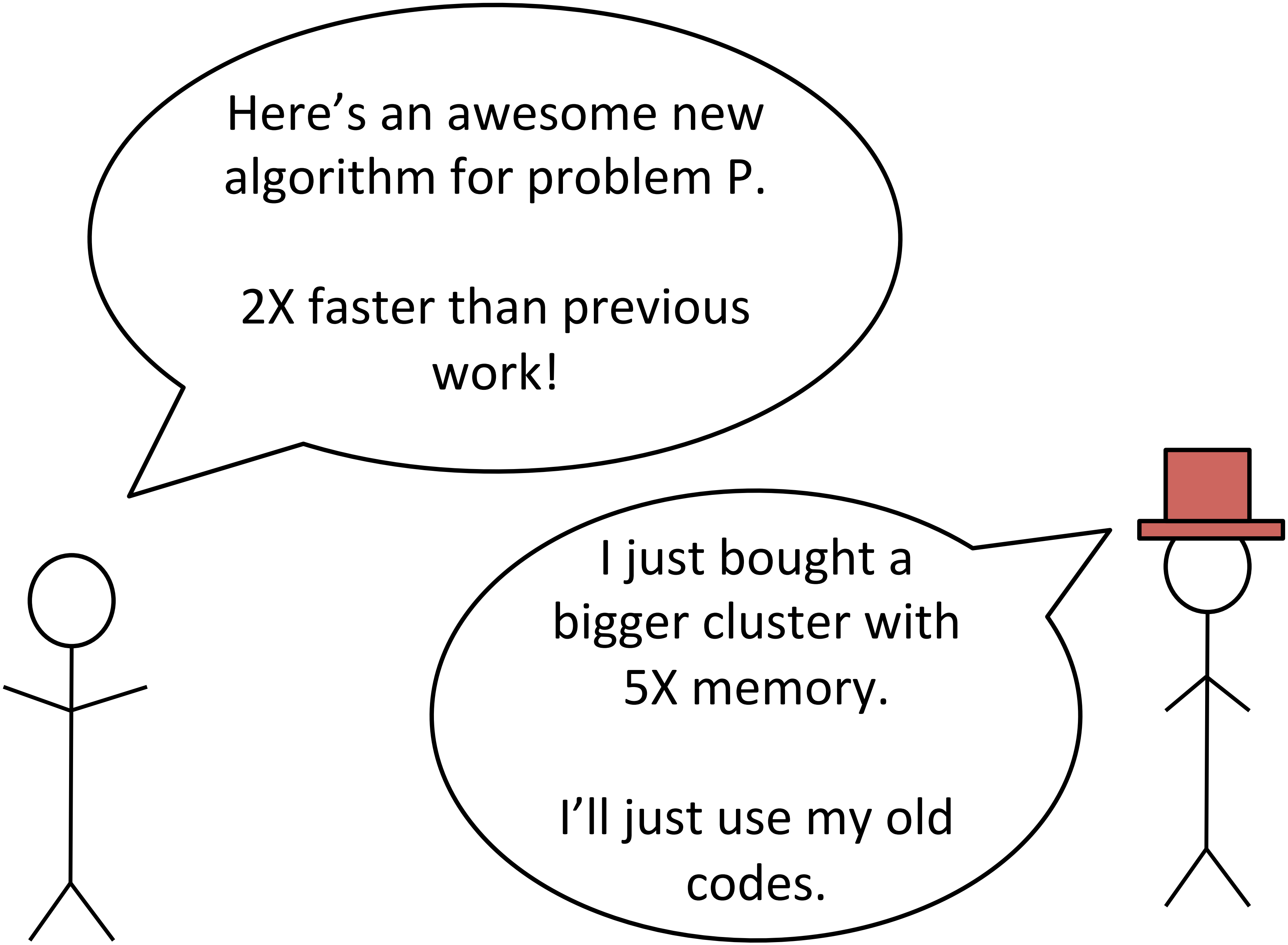
Ashish Goel
Stanford U

This is based on a true story

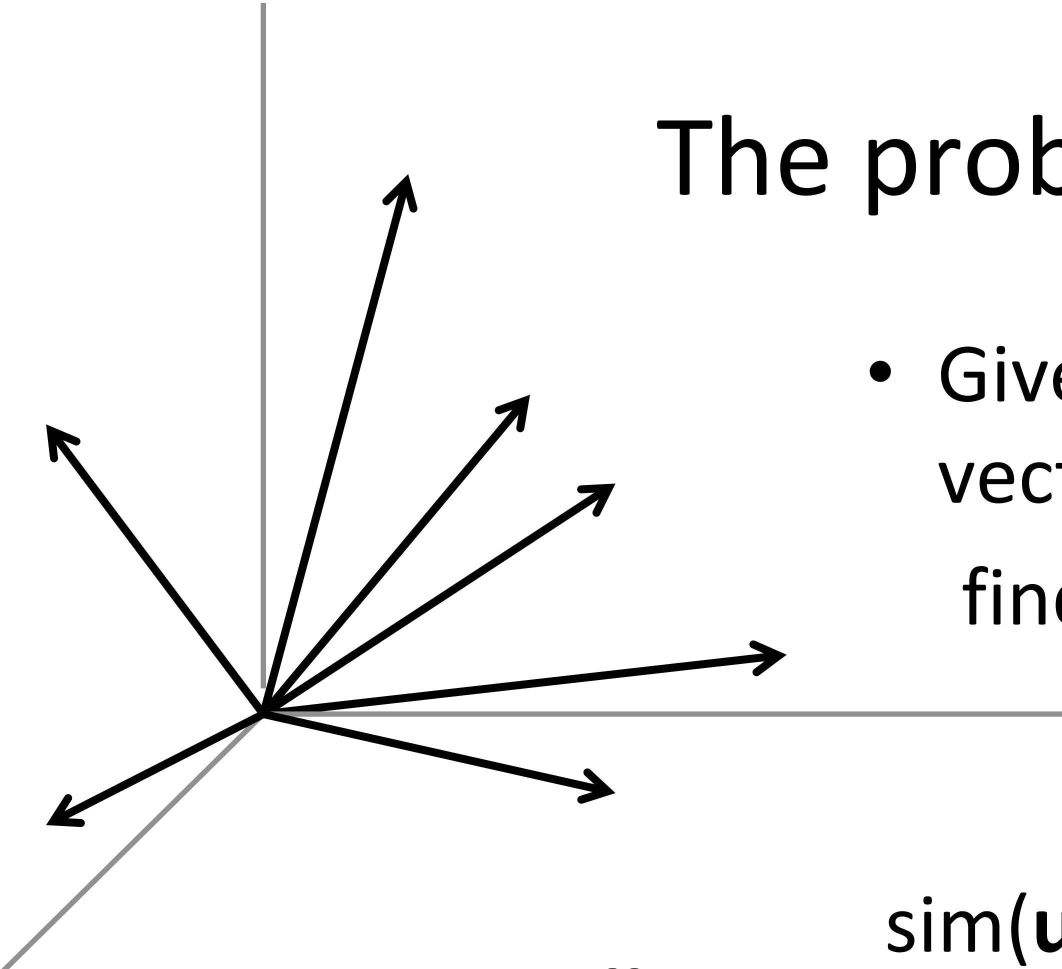
The application is real



Big data ➔Theory ➔Practice


Here's an awesome new algorithm for problem P.
2X faster than previous work!
I just bought a bigger cluster with 5X memory.
I'll just use my old codes.

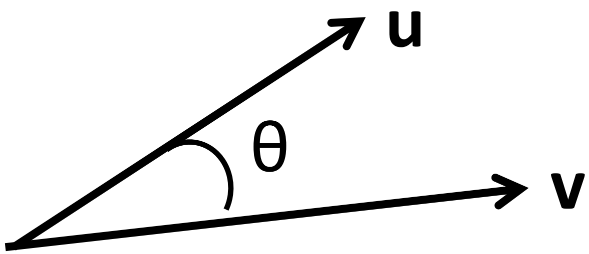
The problem



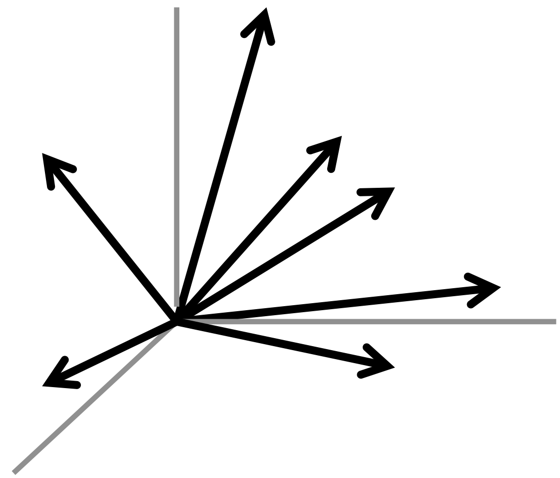
- Given n non-negative vectors in R^d, find all "similar" pairs

$\mathrm{sim}(\mathbf{u},\mathbf{v}) = \cos\theta$

(or $\mathrm{sim}(\mathbf{u},\mathbf{v}) = \mathbf{u}.\mathbf{v}$)

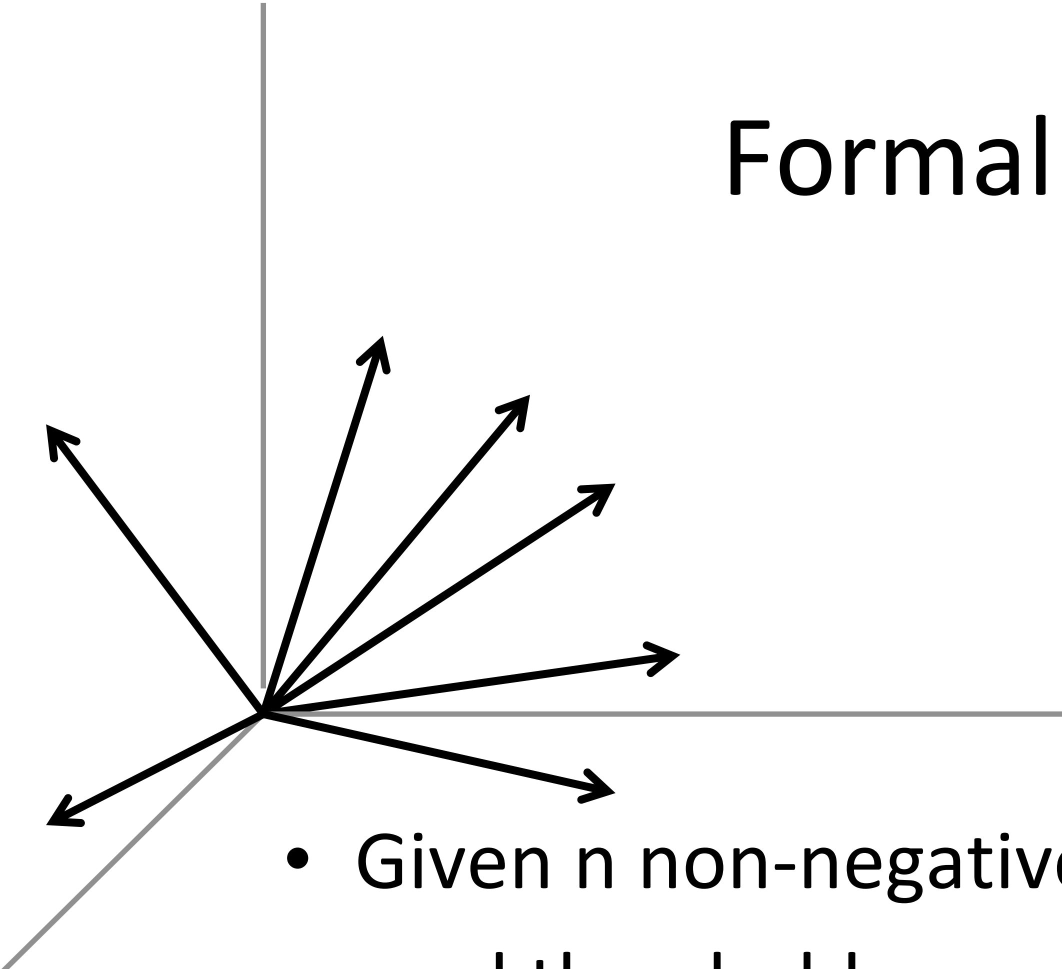


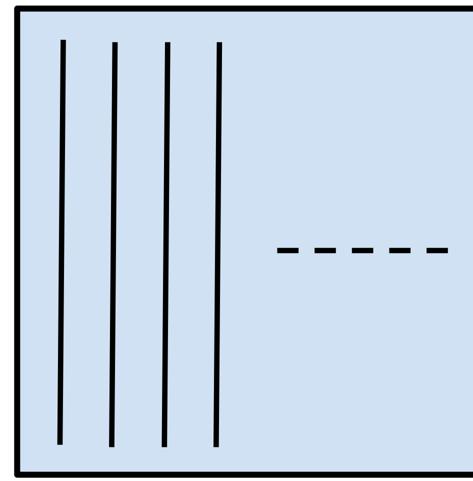
Why?



- Fundamental for link prediction and recommendation
- [Goel et al 13][Gupta et al 13] Key feature in Who To Follow engine at Twitter
  - Common representations are non-negative

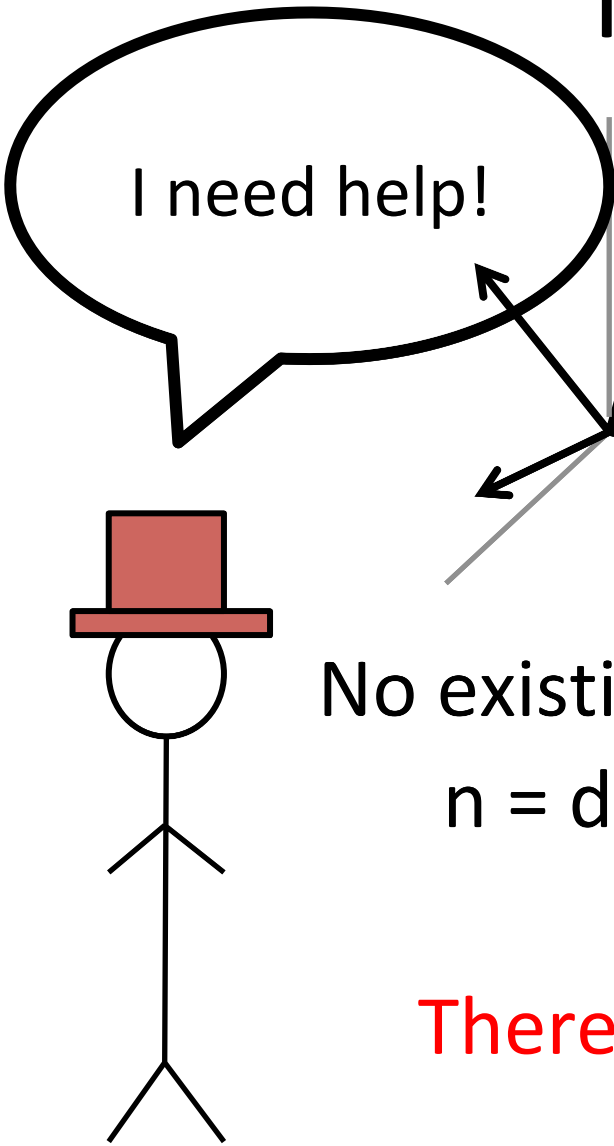
Formally…

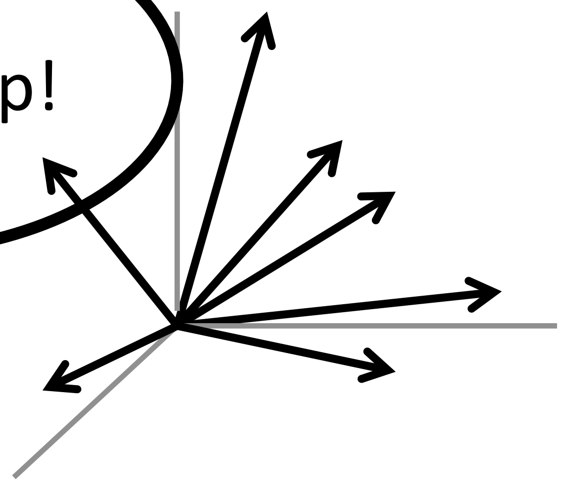


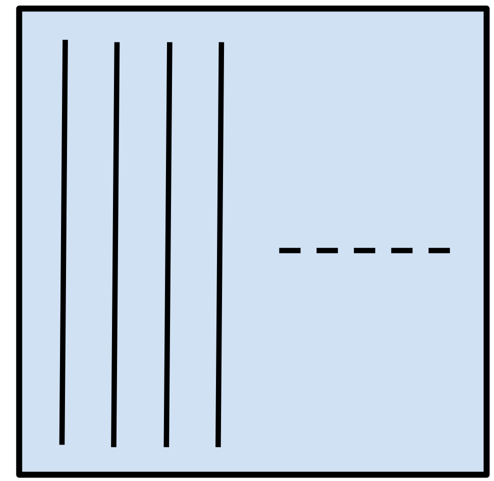


- Given n non-negative unit vectors in R^d
  and threshold τ,
  find all pairs (**u**,**v**) such that **u**.**v** > τ
- In A^TA, find all entries > τ

The challenge



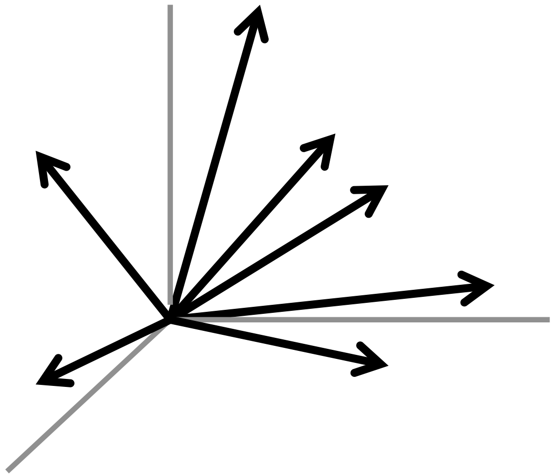




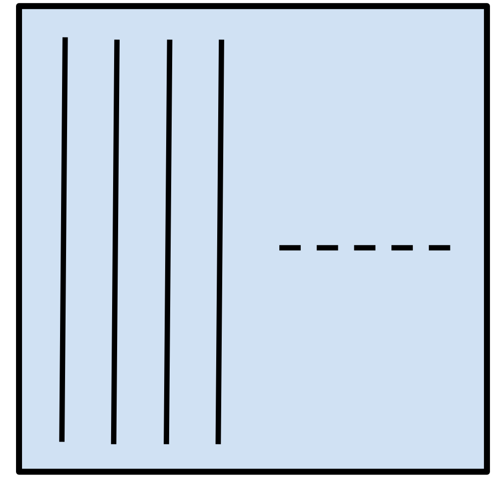
No existing algorithm works when

n = d = 1B and nnz(A) > 10B

There is no systems solution



WHIMP

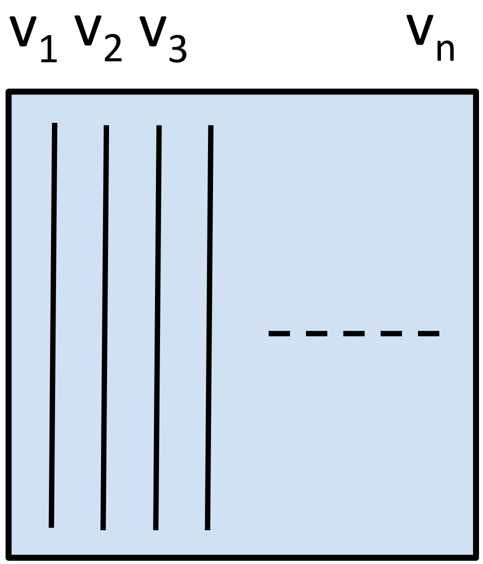


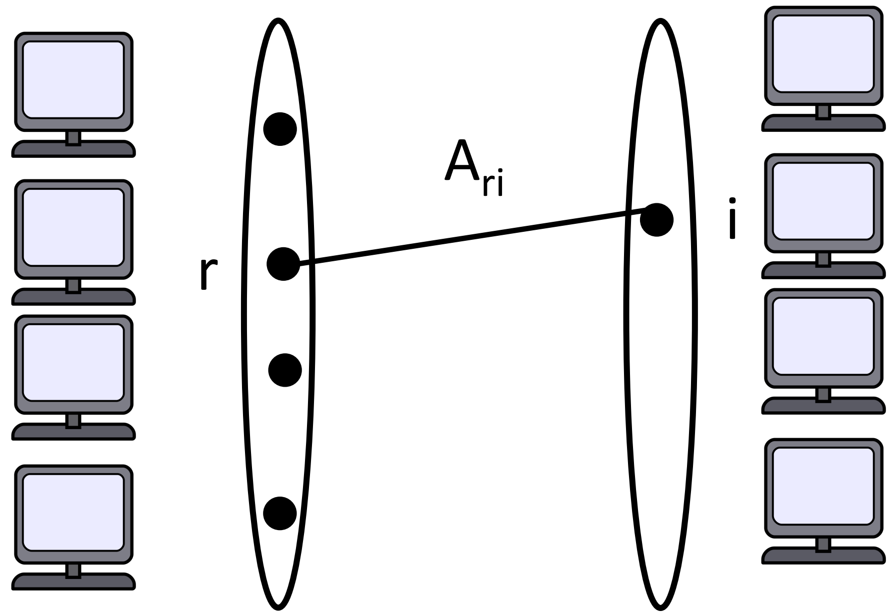
- WHIMP (Wedges and Hashes In Matrix Prod.)

Distributed (MR) algorithm for finding similar vectors

  – Theoretically "near-optimal" total shuffle/comm
  – Practically viable. Works on nnz(A) = O(100B) without killing cluster

The distributed framework





- Synchronous communication along edges (can be simulated in MR)
- Total communication is shuffle cost

Previous art

- Exact matrix mult: [BLAS, Csparse]
- Approx matrix mult, using low rank approximation: [Drineas-Kannan-Mahoney 06] [Sarlos 06][Belabbas-Wolfe 08]
- Random projections, (Asym) LSH [Indyk-Motwani99] [Charikar03] [Andoni-Indyk 06] [Shrivastava-Li15] [Andoni-Indyk-Laarhoven-Razenshteyn-Schmidt15]
- Path sampling: [Cohen-Lewis99] [Schank-Wagner 06][S-Pinar-Kolda 13] [Kolda-Pinar-Plantenga-S-Task 14] [Zadeh-Goel 15] [Ballard-Kolda-Pinar-S 15]

Pre ious a

- Exact m rix mult e]
- Approx
  urlos
  0
- Ran
  tava-Li15]
- F h-Goe [Ballard
  et

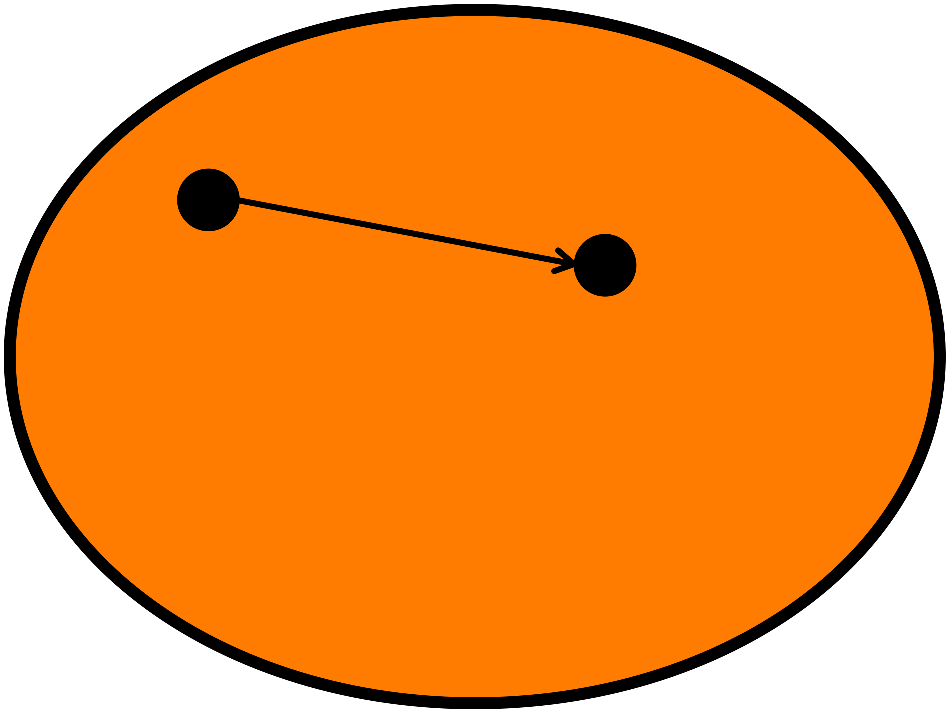
Too much communication!

Philosophers and psychiatrists should explain why it is that we mathematicians are in the habit of **systematically erasing our footsteps**…

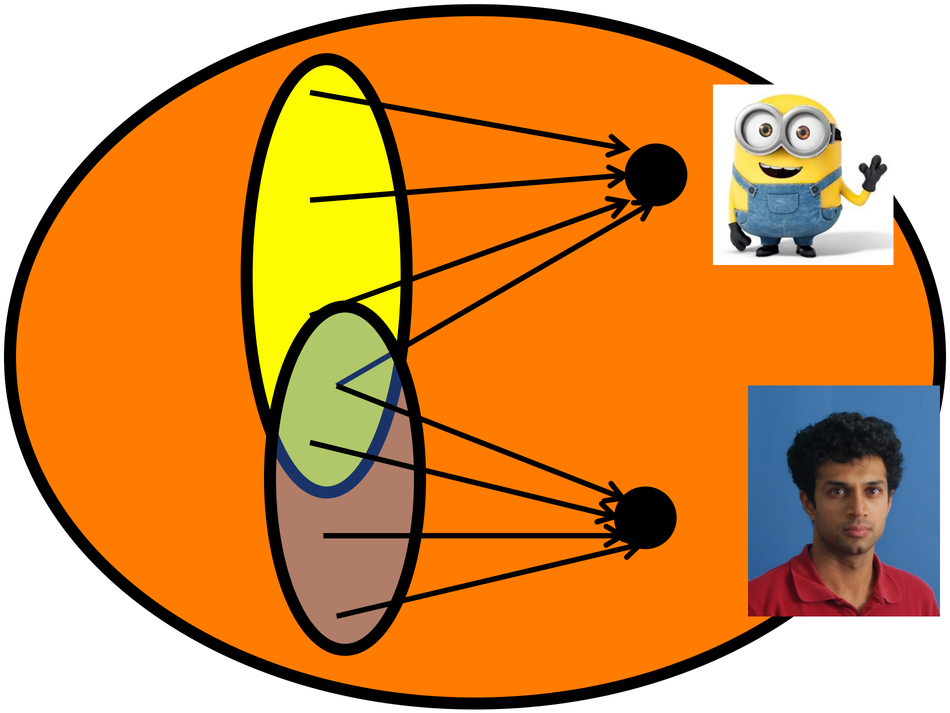
- Gian-Carlo Rota

I’ll tell you about an erased footstep.

The Twitter problem



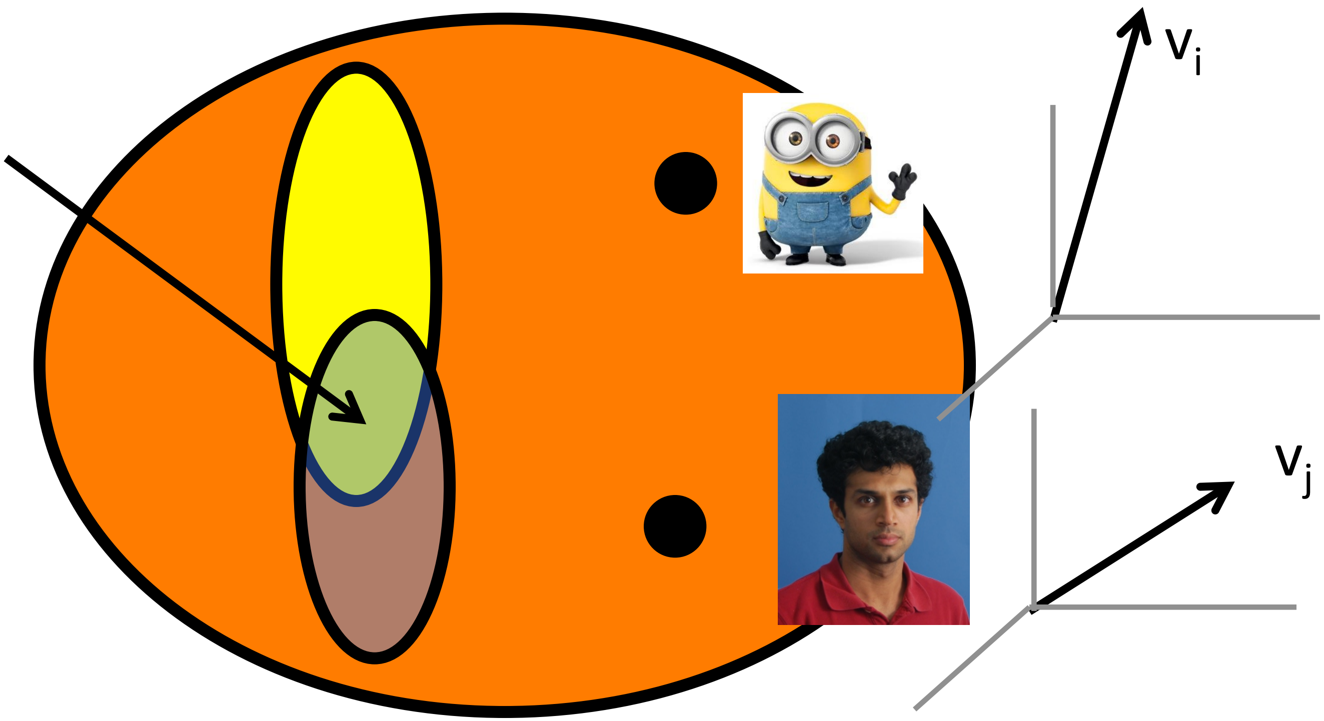
The Twitter problem



- Users with large intersections of followers tend to be “similar”

The Twitter problem

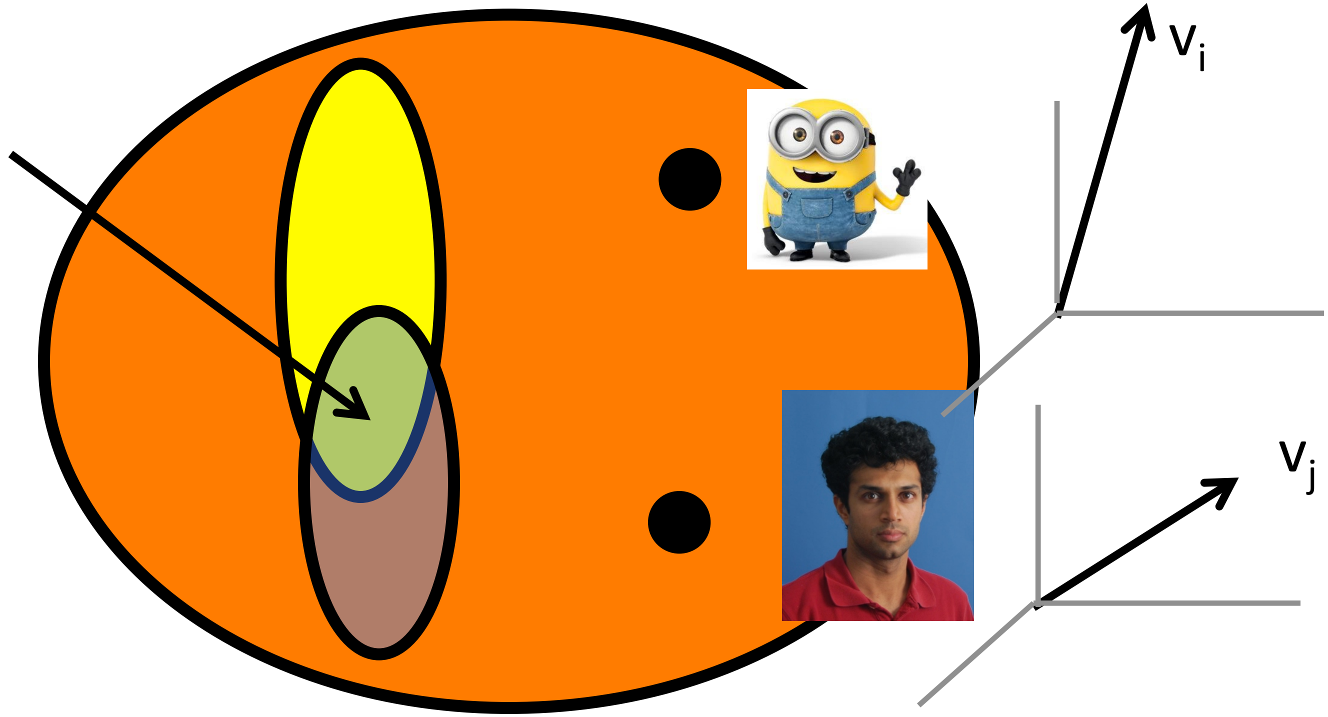
$$\frac{|S \cap T|}{\sqrt{|S||T|}}$$



- Cosine similarity is “normalized intersection”

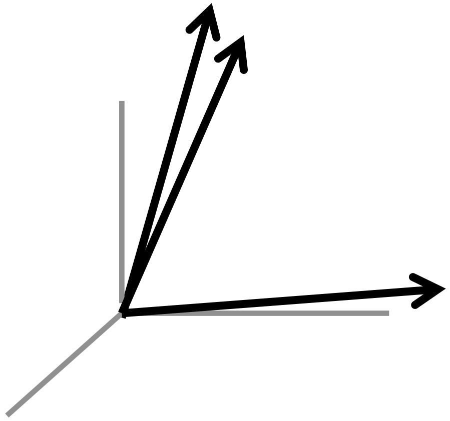
The Twitter problem

$$\frac{|S \cap T|}{\sqrt{|S||T|}}$$



- Domain studies show similarities of 0.15 – 0.2 matter
- 15% of my followers follow you. We need to know

The similarity threshold



- Most literature on low dimensional projections/hashing/nearest neighbor for on sim > 0.8
- In recommendations, similarities around 0.1-0.3 matter

Real recommendations

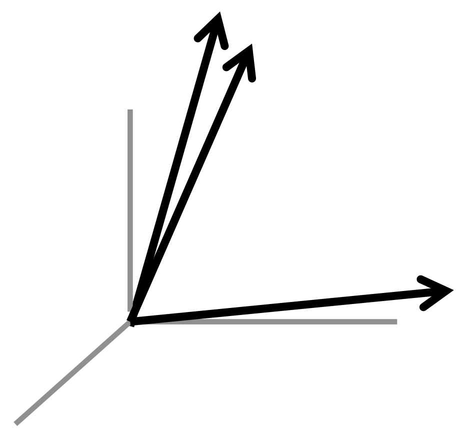
Users similar to @www2016ca

Rank	Twitter @handle	Score
1	@WSDMSocial	0.268
2	@WWWfirenze	0.213
3	@SIGIR2016	0.190
4	@ecir2016	0.175
5	@WSDM2015	0.155

Users similar to @duncanjwatts

Rank	Twitter @handle	Score
1	@ladamic	0.287
2	@davidlazer	0.286
3	@barabasi	0.284
4	@jure	0.218
5	@net_science	0.200

The quadratic bottleneck



- To find similarities of τ, you need $1/\tau^2$ work or communication (or pain)
- A well-engineered solution for $\tau = 0.9$ fails miserably for $\tau = 0.2$ (20X more pain)

Our real contribution

- Theorem: To find similarities of τ, WHIMP requires communication/shuffle

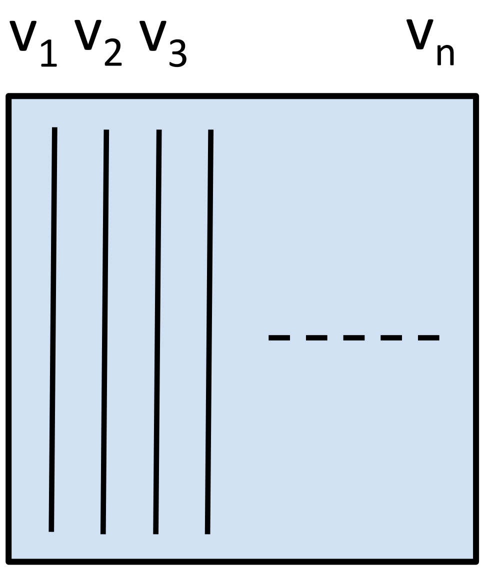
lower bound on output

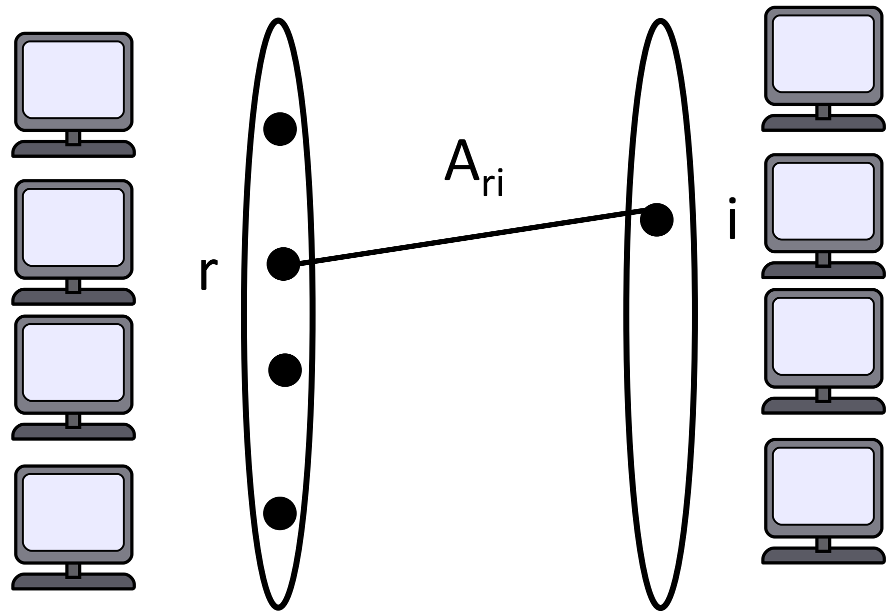
$$(\tau^{-1} \log n) \boxed{(\#\text{ pairs with sim} > \tau)}$$
$$+ (\tau^{-2} \log n)(\text{nnz}(A))$$

typically large

- In previous methods, the τ^{-1} and τ^{-2} terms multiply larger quantities

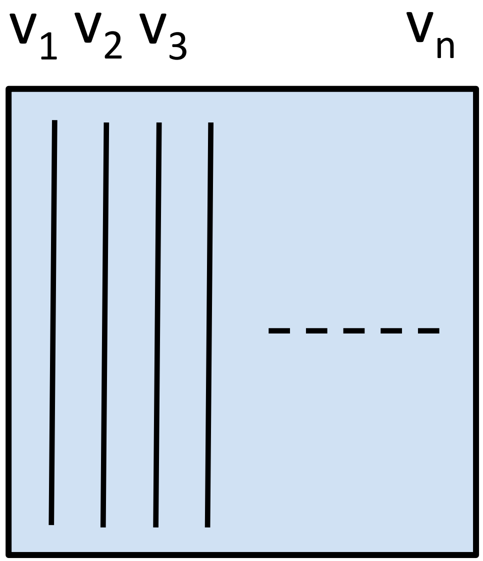
The distributed framework

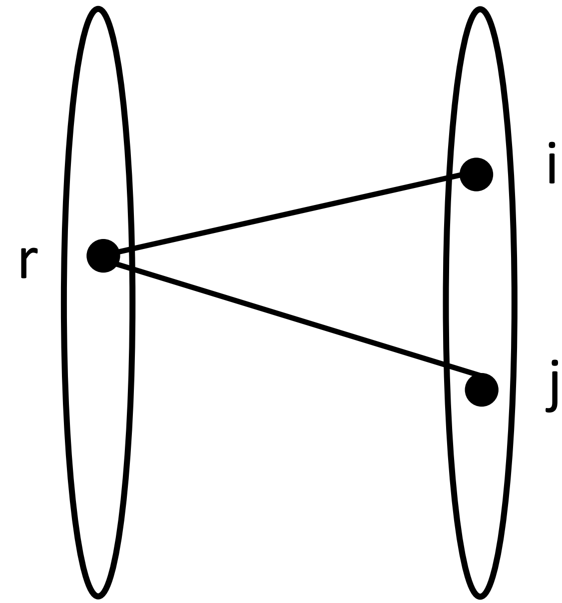




- Synchronous communication along edges (can be simulated in MR)
- Total communication is shuffle cost

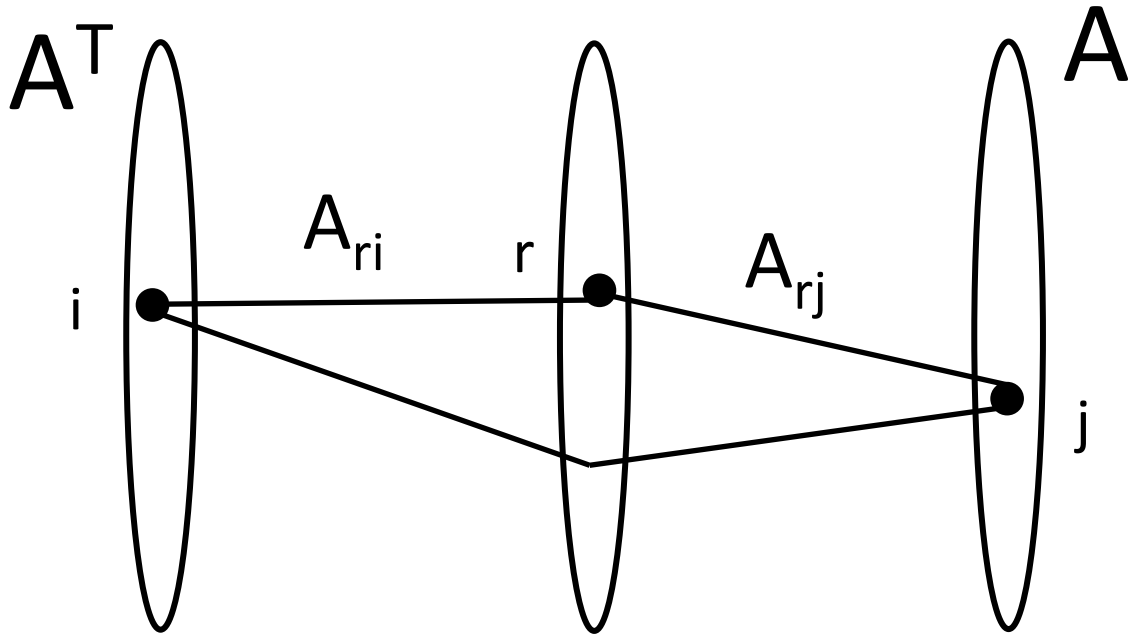
Wedge sampling





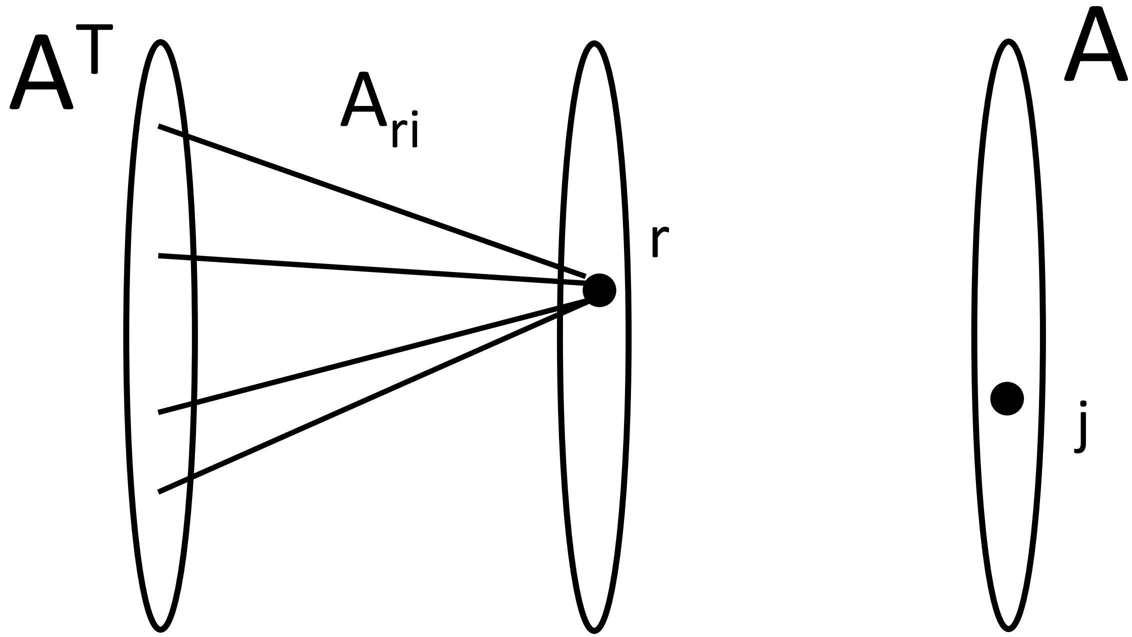
- [Cohen-Lewis 99], [Schank-Wagener 06], [S-Pinar-Kolda 13], [Zadeh-Goel 16]
- nnz(A) time preprocessing
- In O(1) time, generates wedge (i, r, j)
- Pr[wedge with ends i,j] proportional to $v_i \cdot v_j$

Wedge sampling



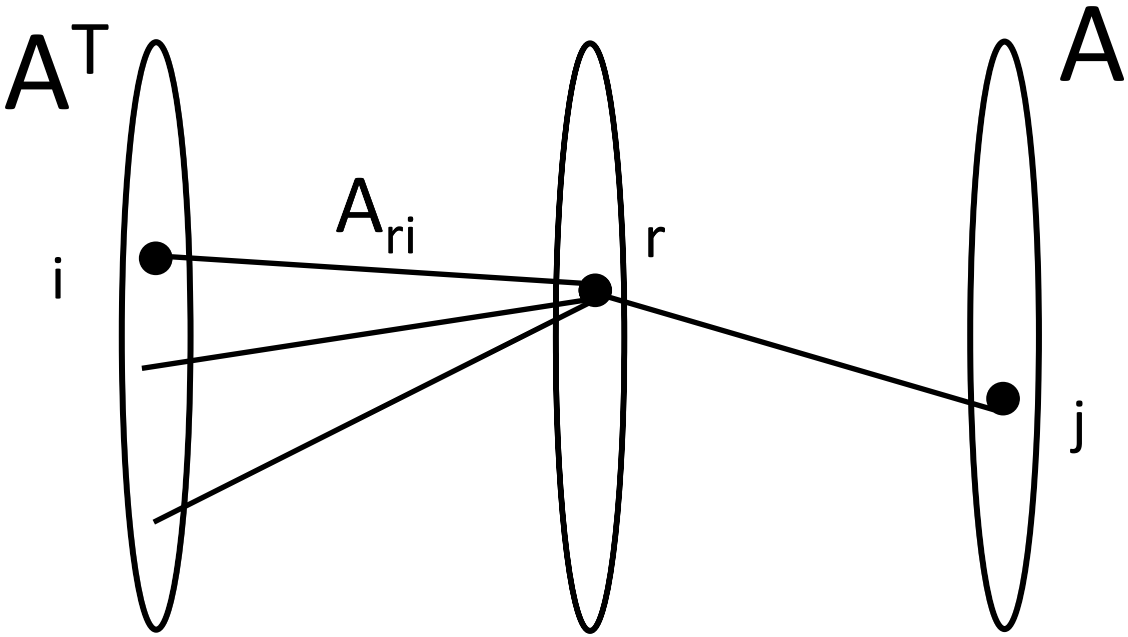
- Weight of path (i,r,j) = $A_{ri}\ A_{rj}$
- Sum over paths from i to j = $\Sigma_r\ A_{ri}\ A_{rj} = v_i \cdot v_j$
- Sample path proportional to weight; probability of getting (i,j) prop. to $v_i \cdot v_j$
  - Non-negativity used!

Cohen-Lewis trick



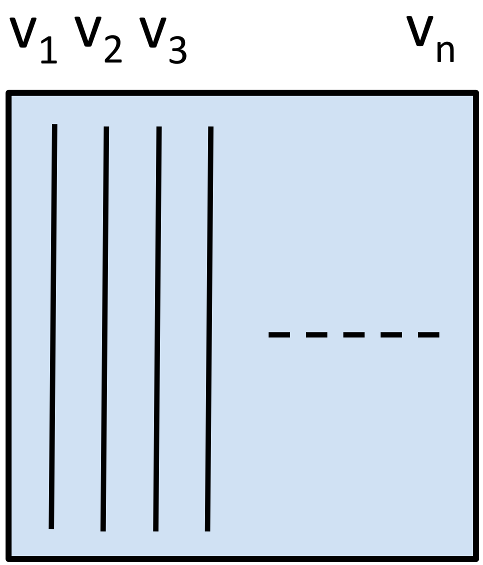
- Preprocess to compute $w_r = \Sigma_i A_{ri}$
- Build data structure to sample r prop. to w_r

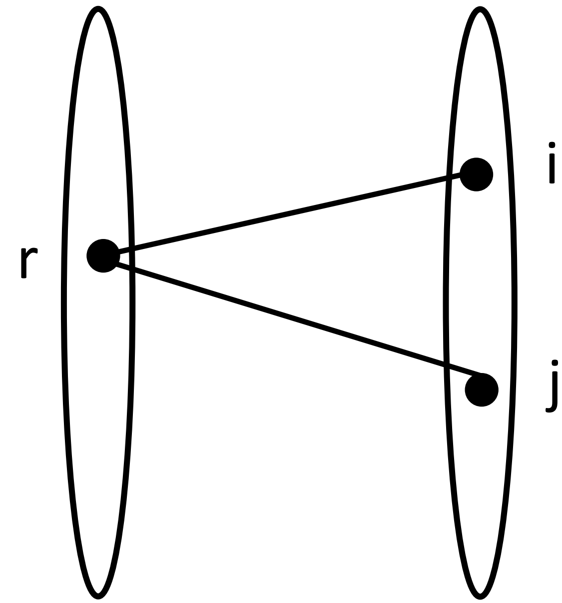
Cohen-Lewis trick



- Preprocess to compute $w_r = \Sigma_i A_{ri}$
- Build data structure to sample r prop. to w_r
- Pick i w.p. A_{ri}/w_r , and repeat to get j
- Output (i,j)

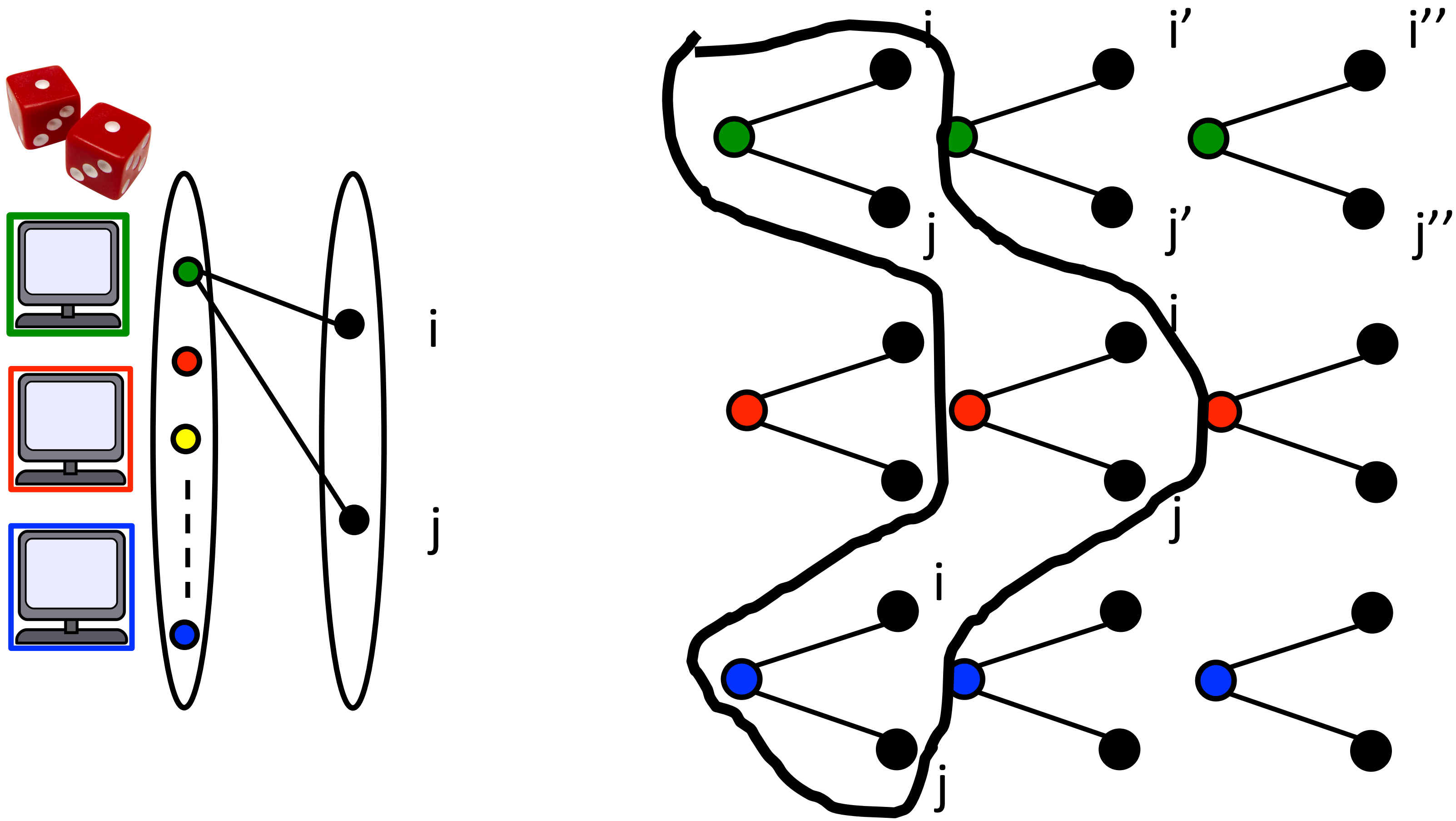
Wedge sampling





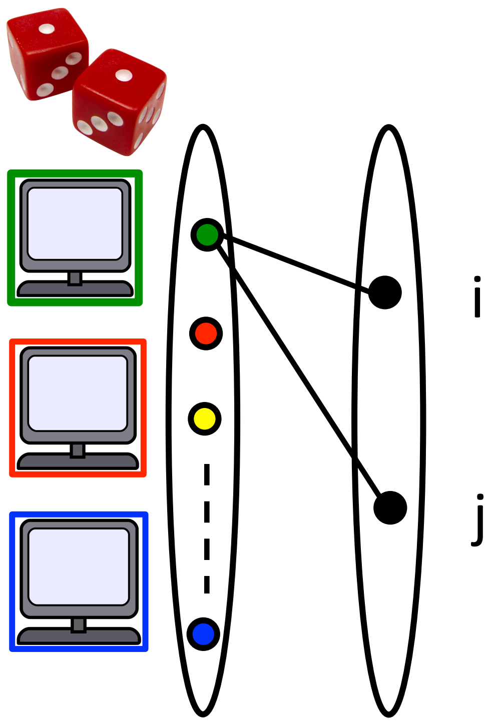
- [Cohen-Lewis 99], [Schank-Wagener 06], [S-Pinar-Kolda 13], [Zadeh-Goel 16]
- nnz(A) time preprocessing
- In O(1) time, generates wedge (i, r, j)
- Pr[wedge with ends i,j] proportional to $v_i \cdot v_j$

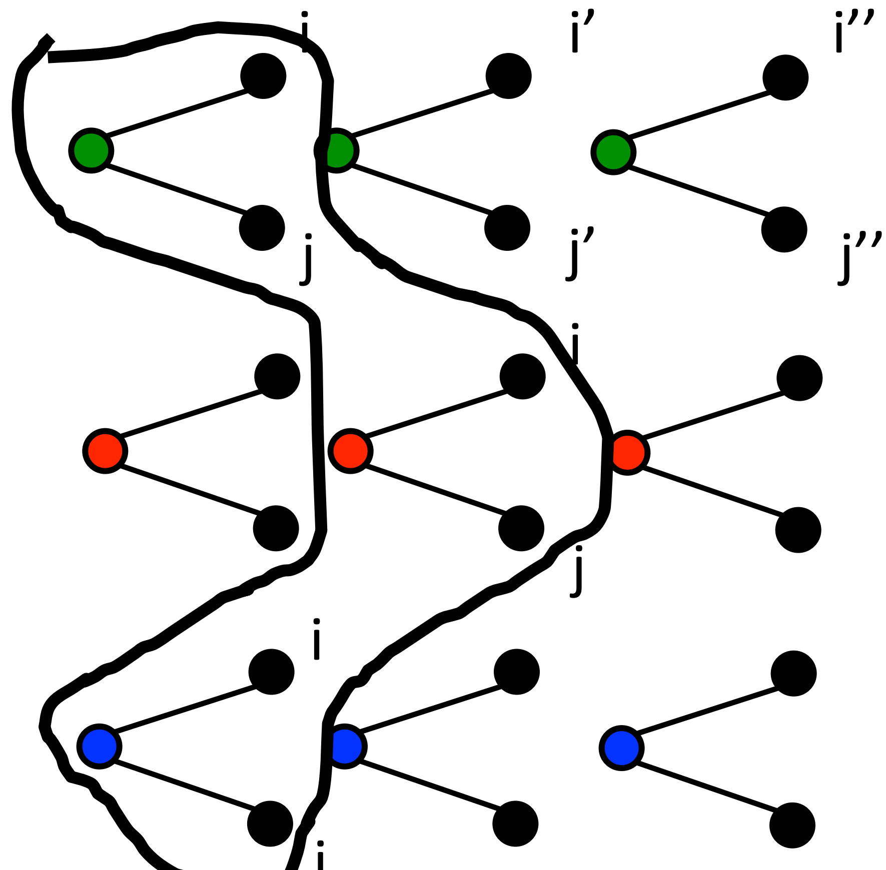
Distributed wedge sampling



- [Zadeh-Goel 15] DISCO: Frequent “candidates” tend to be large entries of product matrix
- Requires shuffle/communication of all wedges

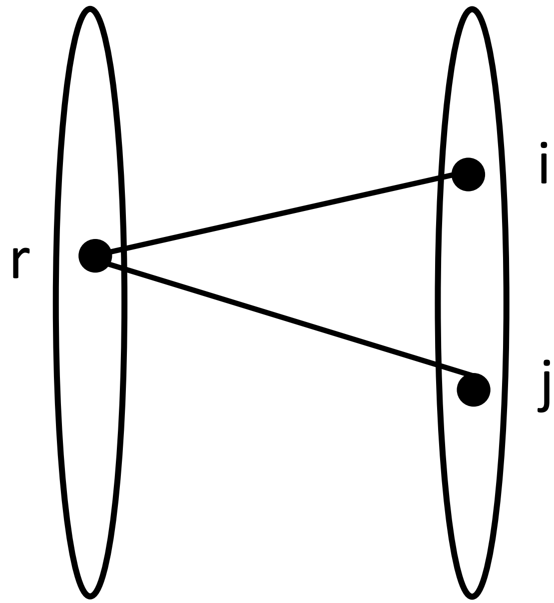
Distributed wedge sampling





So how may wedges do we need to catch all $v_i \,.\, v_j > \tau$?

How many wedges?



Pr[wedge with i,j]
proportional to $v_i \, . \, v_j$

$$\mathbf{Pr}[\text{wedge with } (i,j)] = \frac{v_i \cdot v_j}{\boxed{\sum_{i,j} v_i \cdot v_j}} = \frac{v_i \cdot v_j}{\|A^T A\|_1}$$

Sum of all
dot products/similarities

How many samples?

$$\mathbf{Pr}[\text{wedge with } (i,j)] \quad = \frac{v_i \cdot v_j}{\|A^T A\|_1}$$

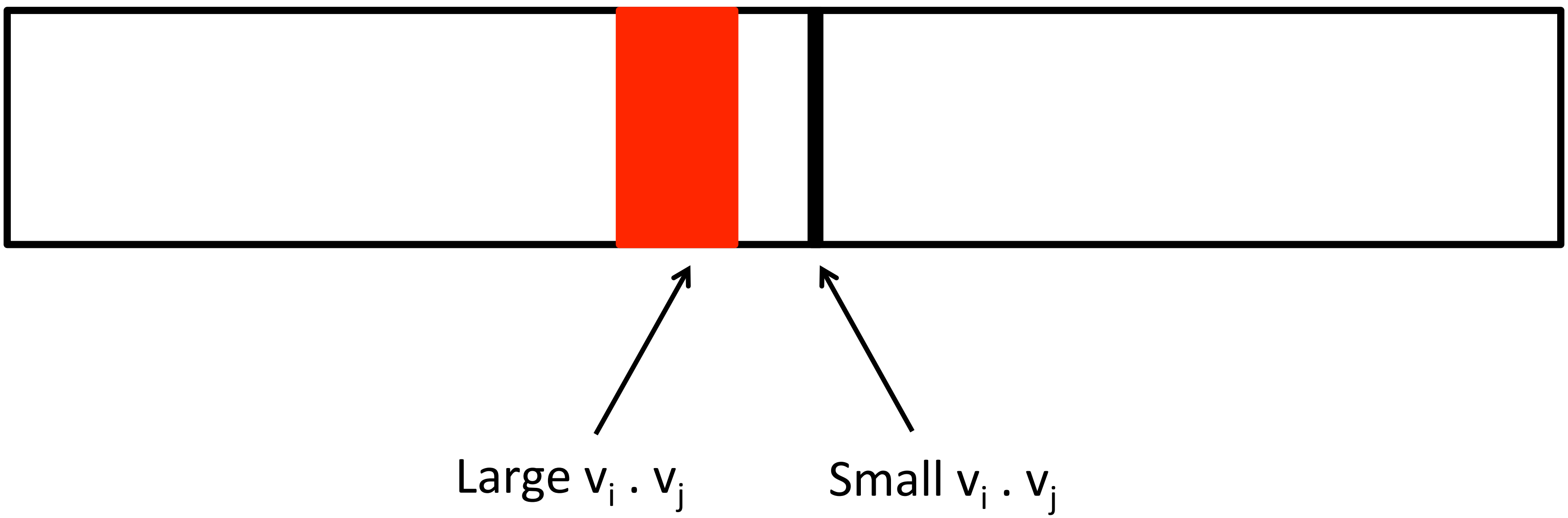
Suppose $v_i . v_j = \tau$

$$\#samples \approx \frac{10\|A^T A\|_1}{\tau}$$

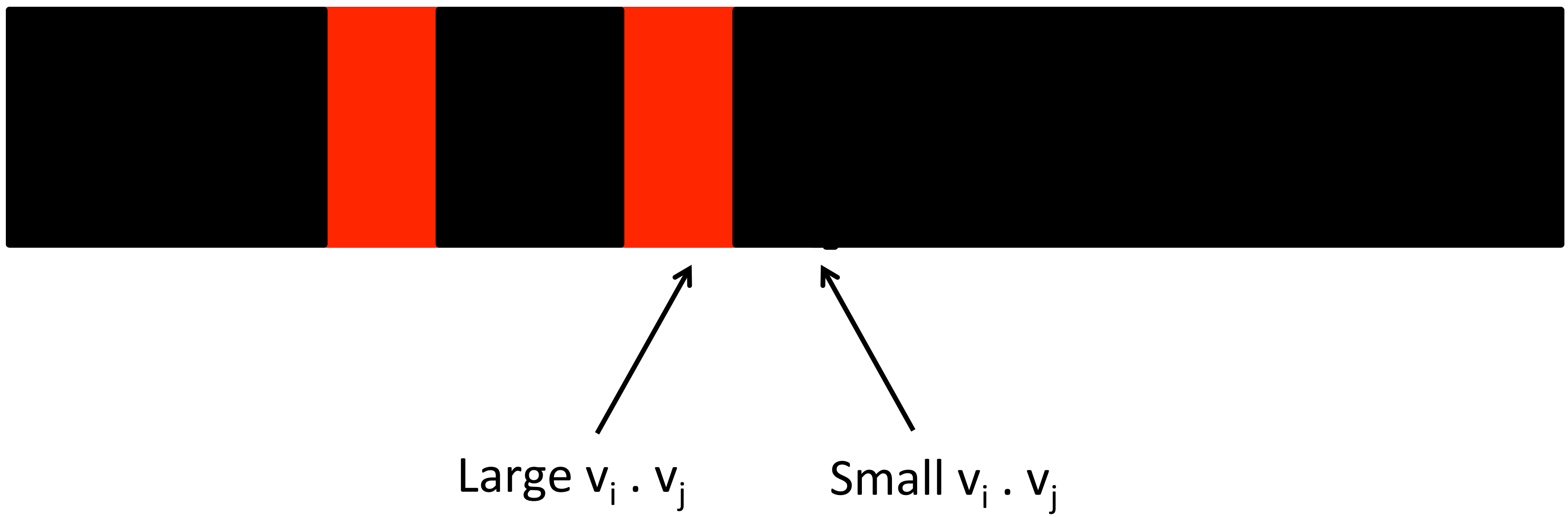
We only want large entries in $A^T A$

But # wedge samples is linear in $|A^T A|$

Signal vs noise



Signal vs noise



- Too many small entries “drown” out the few large entries
- Most of the communication is noise

How many samples?

$$\mathbf{Pr}[\text{wedge with } (i,j)] \quad = \frac{v_i \cdot v_j}{\|A^T A\|_1}$$

Suppose $v_i . v_j = \tau$

$$\#samples \approx \frac{10\|A^T A\|_1}{\tau}$$

Some numbers

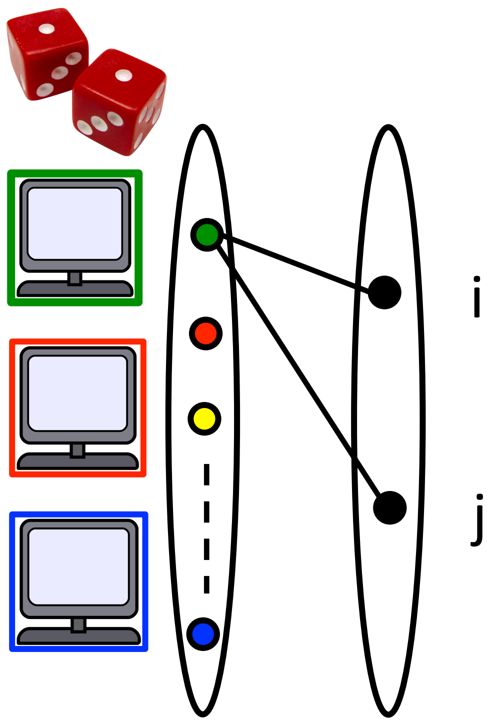
Dataset	Dimensions $n = d$	Size (nnz)	$\|A^T A\|_1$	TB shuffle
friendster	65M	1.6B	7.2E9	26.2
clueweb	978M	42B	6.8E10	247.4
eu	1.1B	84B	1.9E11	691.2
flock	-	$O(100B)$	5.1E12	18553.7

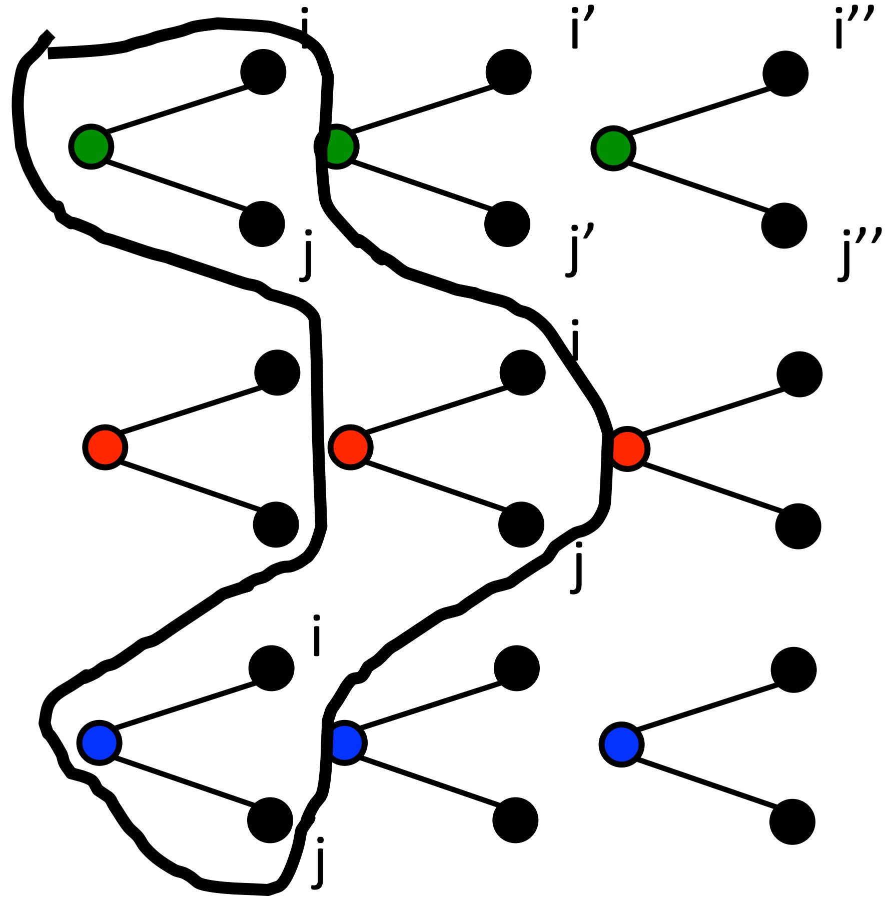
Shuffle = $(10|A^TA|/0.2)$ X 16 bytes

Single round of MR can handle < 150TB

No systems solution for flock

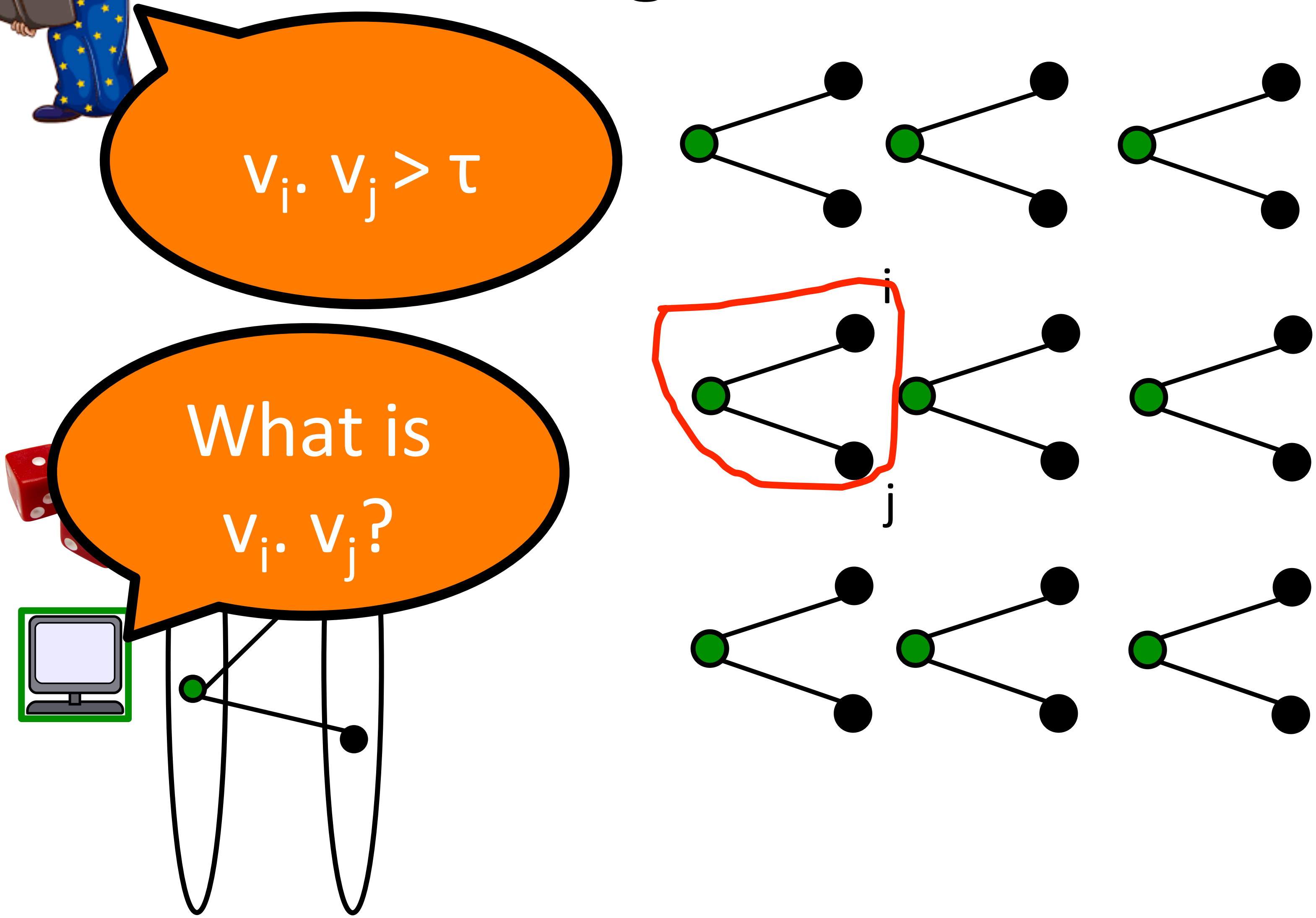
Wedge sampling



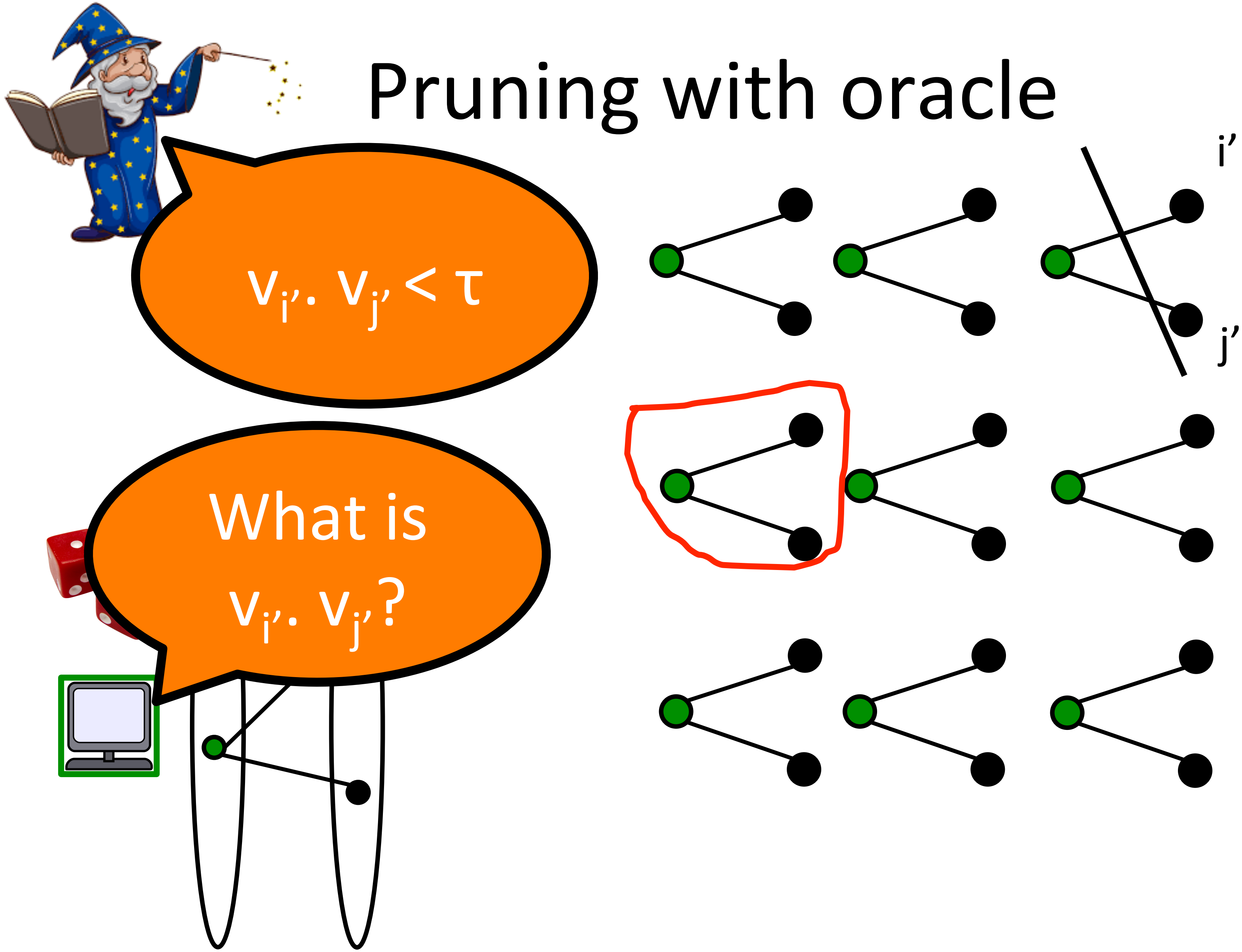


- [Zadeh-Goel 15] DISCO: Frequent “candidates” tend to be large entries of product matrix
- Requires shuffle/communication of all wedges

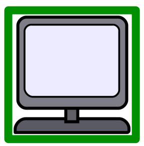
Pruning with oracle

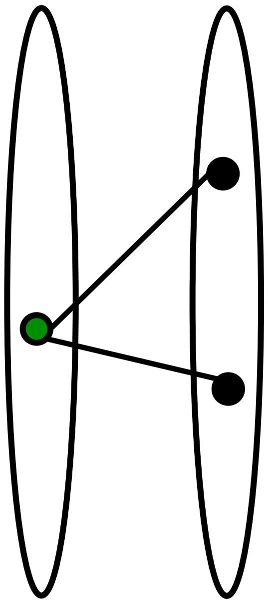


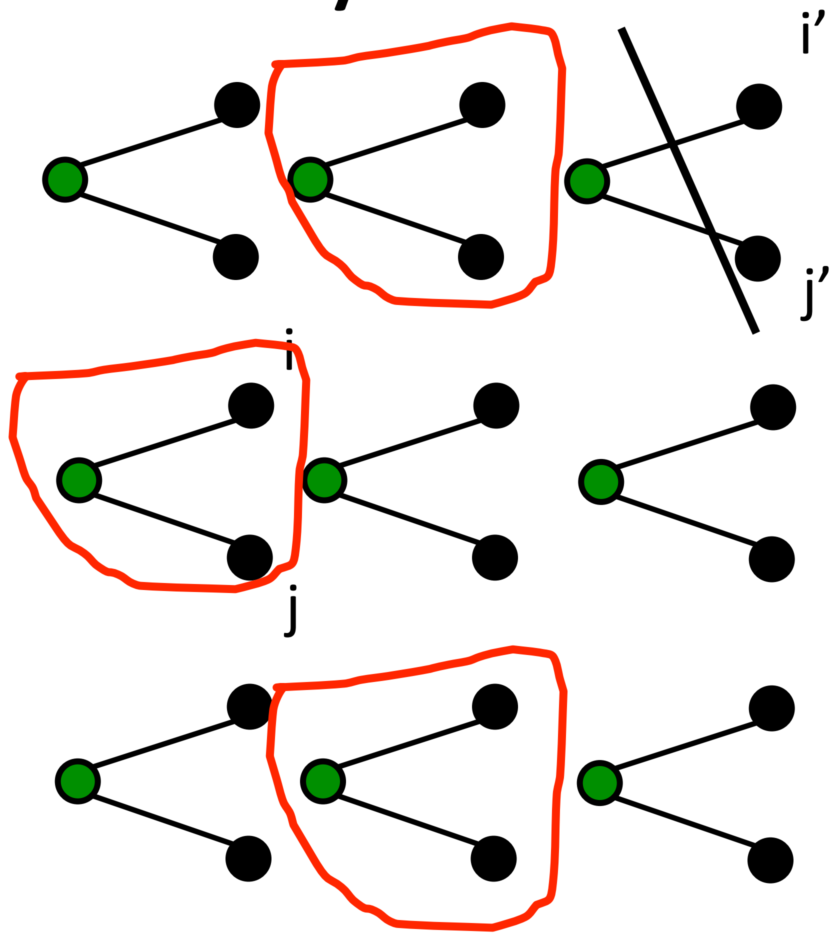
Pruning with oracle



Eventually



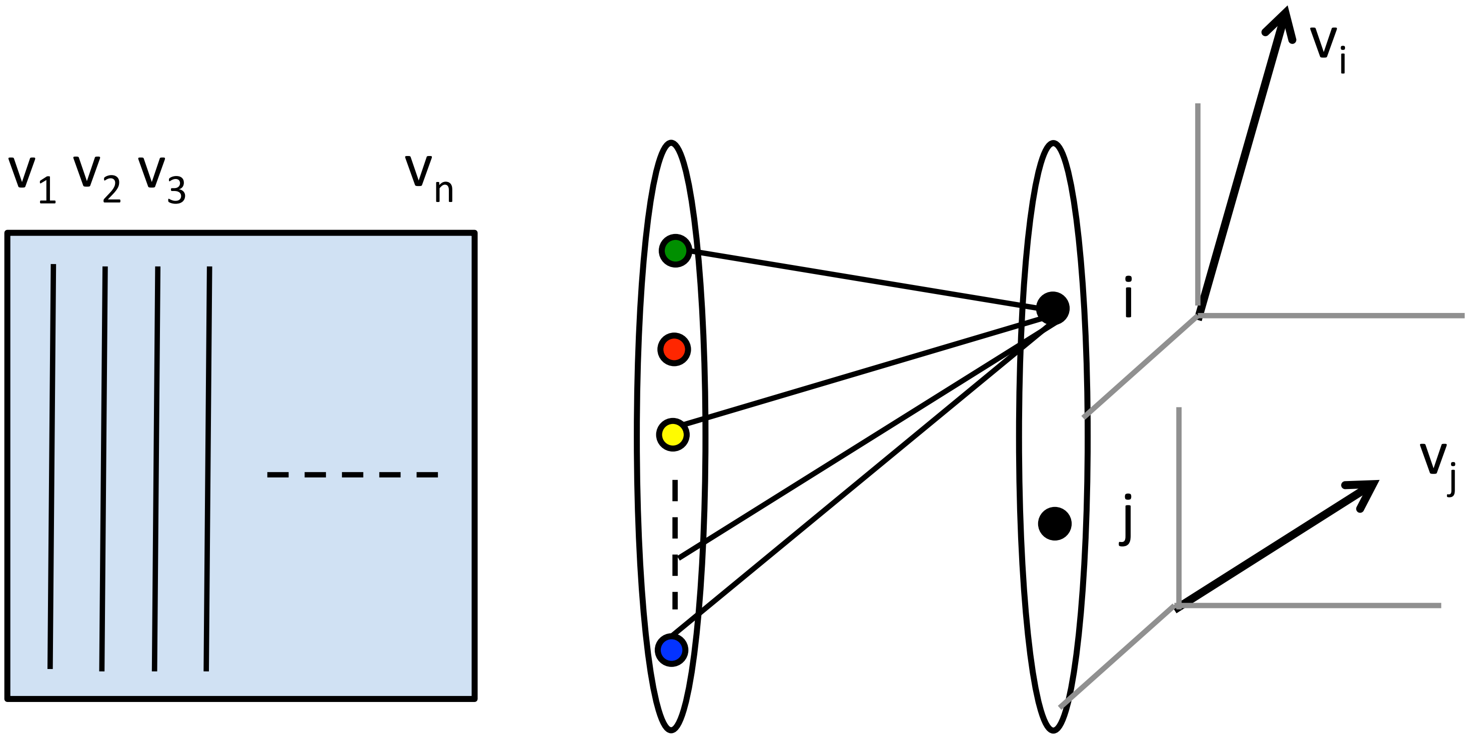




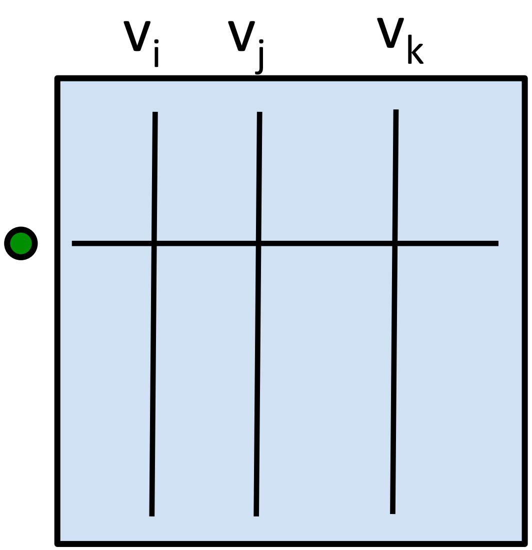
Only wedges with $v_i \cdot v_j > \tau$ are actually communicated!

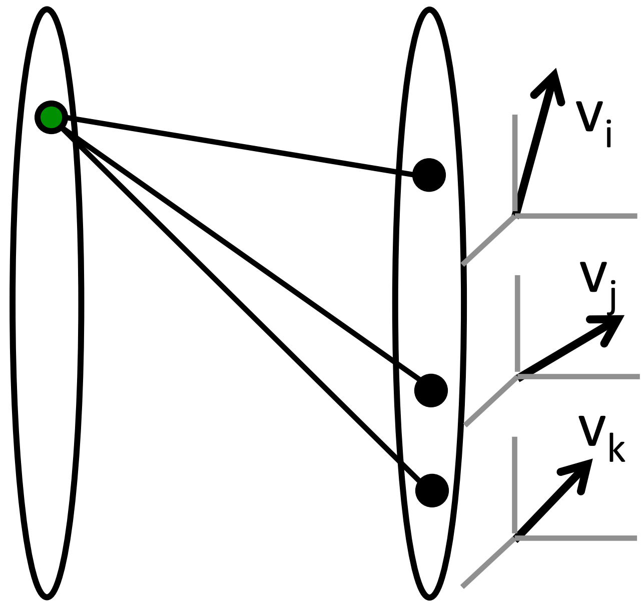
But isn’t designing the oracle the problem itself?

A reminder



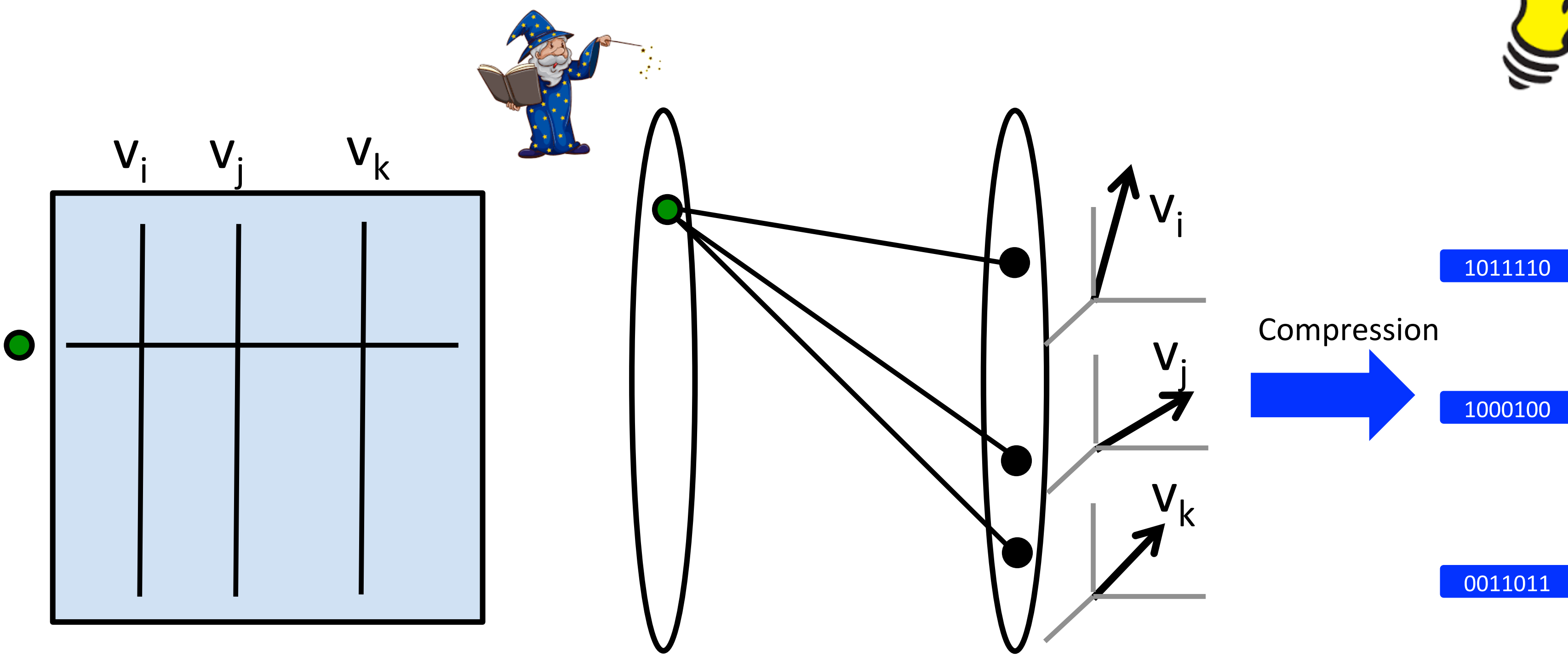
Something obvious





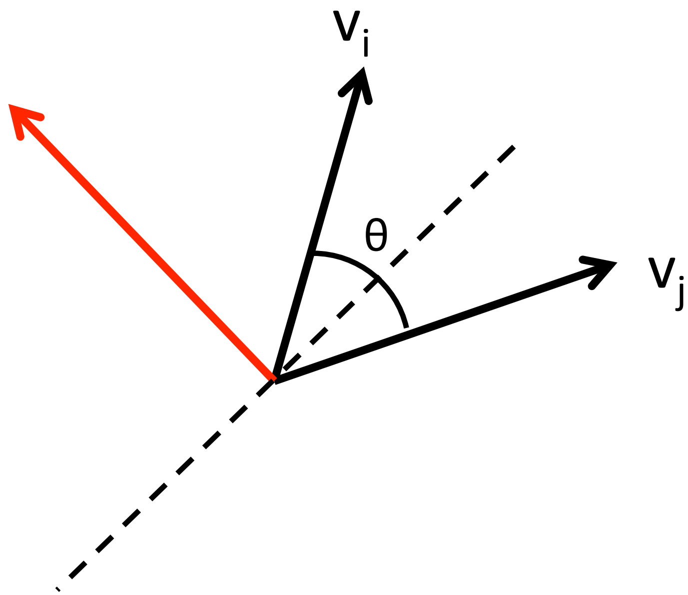
If the green node "knows" all the vectors, it can construct the oracle.
But that's just exact multiplication!

Something not obvious



Green node collects "sketches", and simulates oracle using them

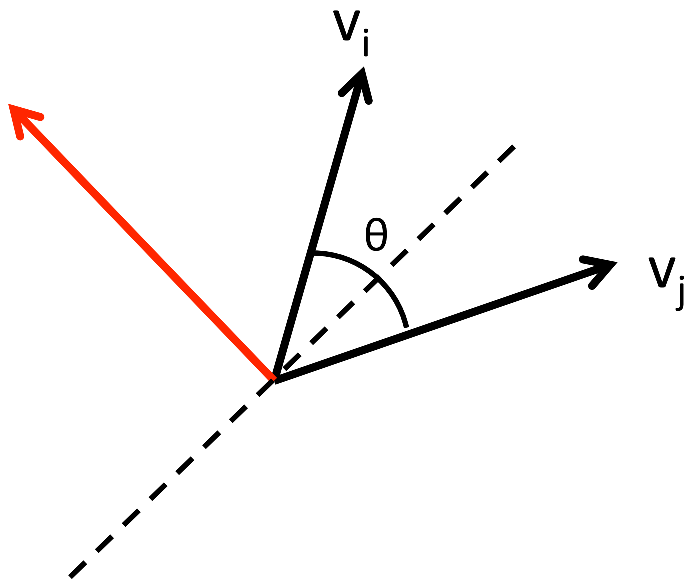
SimHash [Charikar 03]



$h(v_i) = 1$
$h(v_j) = 0$

- Single bit hash = sign of dot product
- $\Pr[h(v_i) = h(v_j)] = 1 - \theta/\pi$

The hashing scheme

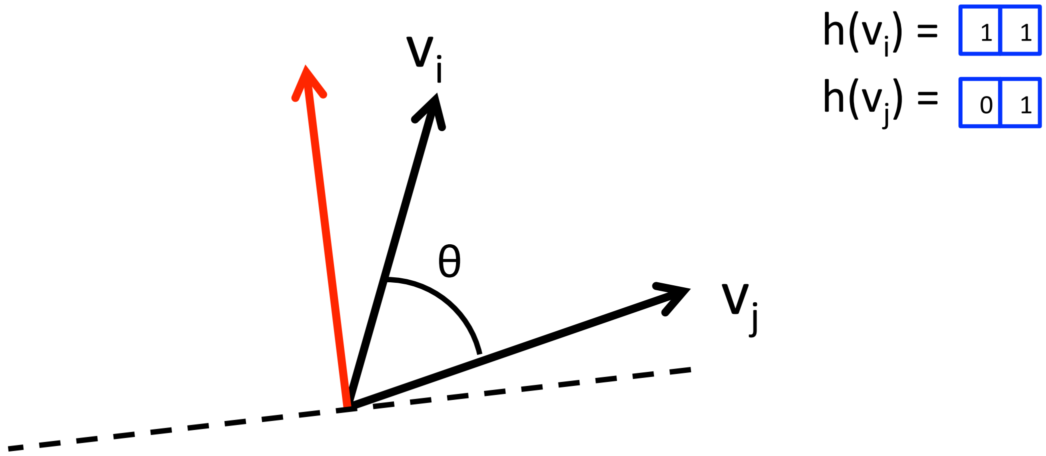


$h(v_i) = \boxed{1}$

$h(v_j) = \boxed{0}$

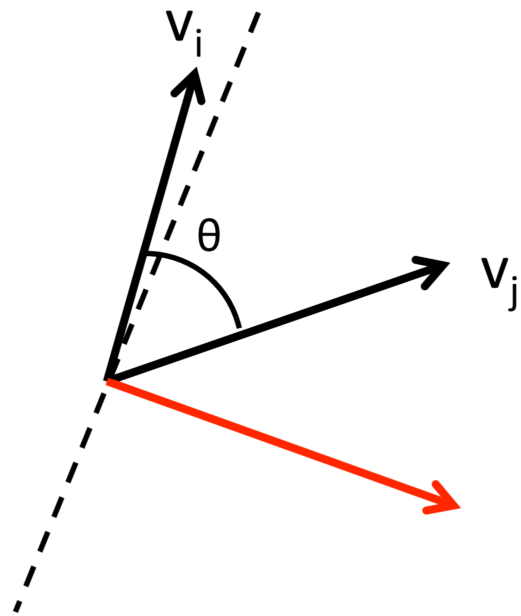
- Rinse and repeat k times

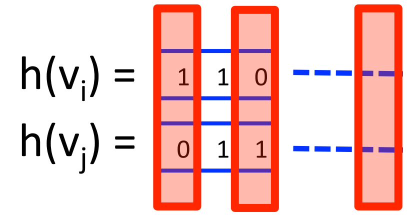
The hashing scheme



- Rinse and repeat k times

The hashing scheme

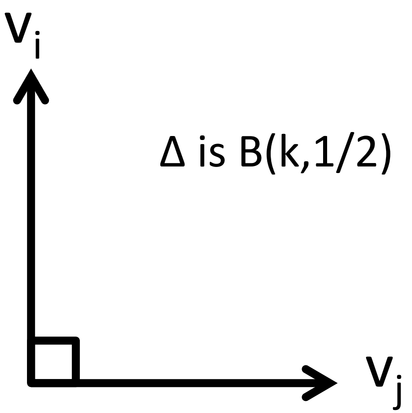




Hamming distance Δ
is measure of angle

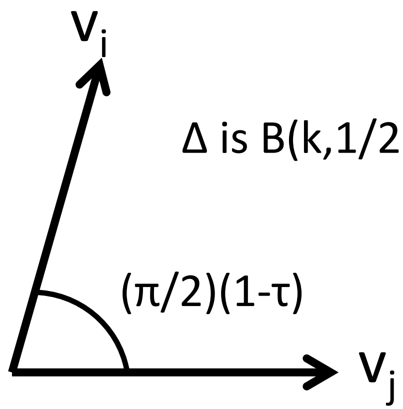
- Δ is binomial B(k, θ/π)
  - If v_i , v_j are orthogonal, Δ is B(k,1/2)
- (Roughly) Δ ≈ kθ/π
- cos(πΔ/k) ≈ cos(θ)

Choosing the hash length



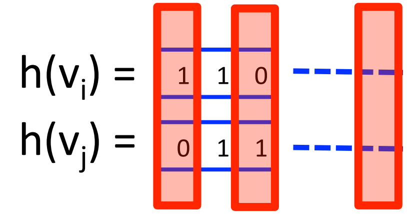
Δ is B(k,1/2)

cos = 0



Δ is B(k,1/2 − τ/2)

cos ≈ τ

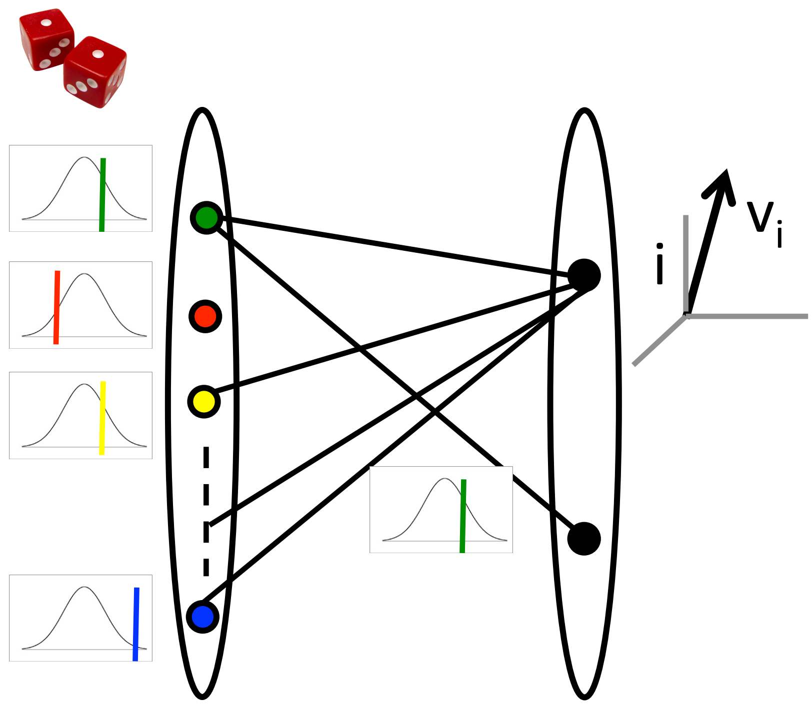


Hamming distance Δ
is measure of angle

- [Chernoff bound] Binomial tails
- Require $1/\tau^2$ flips to distinguish

- Need hash of length $1/\tau^2$ to determine similarities around τ

Generating SimHashes

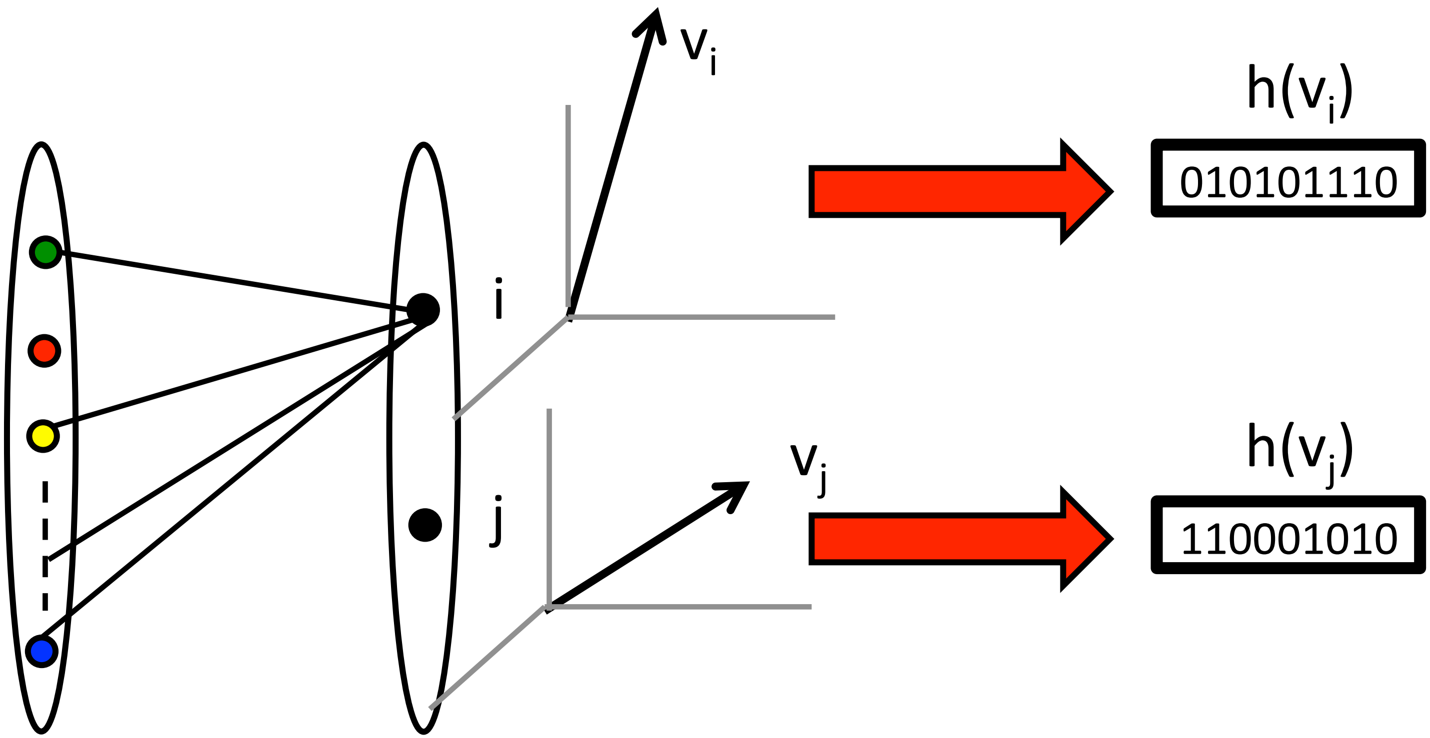


$$h_1(\vec{v_i}) = \vec{v_i} \cdot \vec{g} = \mathrm{sgn}(\sum_r M_{i,r} g_r)$$

- Sending independent Gaussian for each bit is expensive
- We use pseudorandom seeded Gaussians to reduce communication

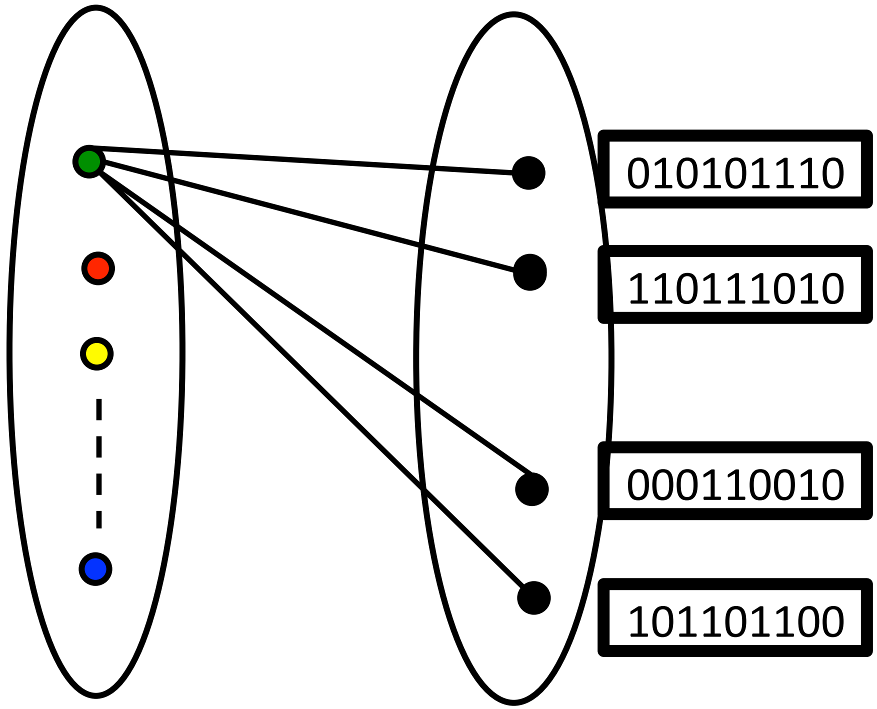
WHIMP = Wedge Sampling + SimHash (Hashes)

WHIMP, Round 1: Hashing



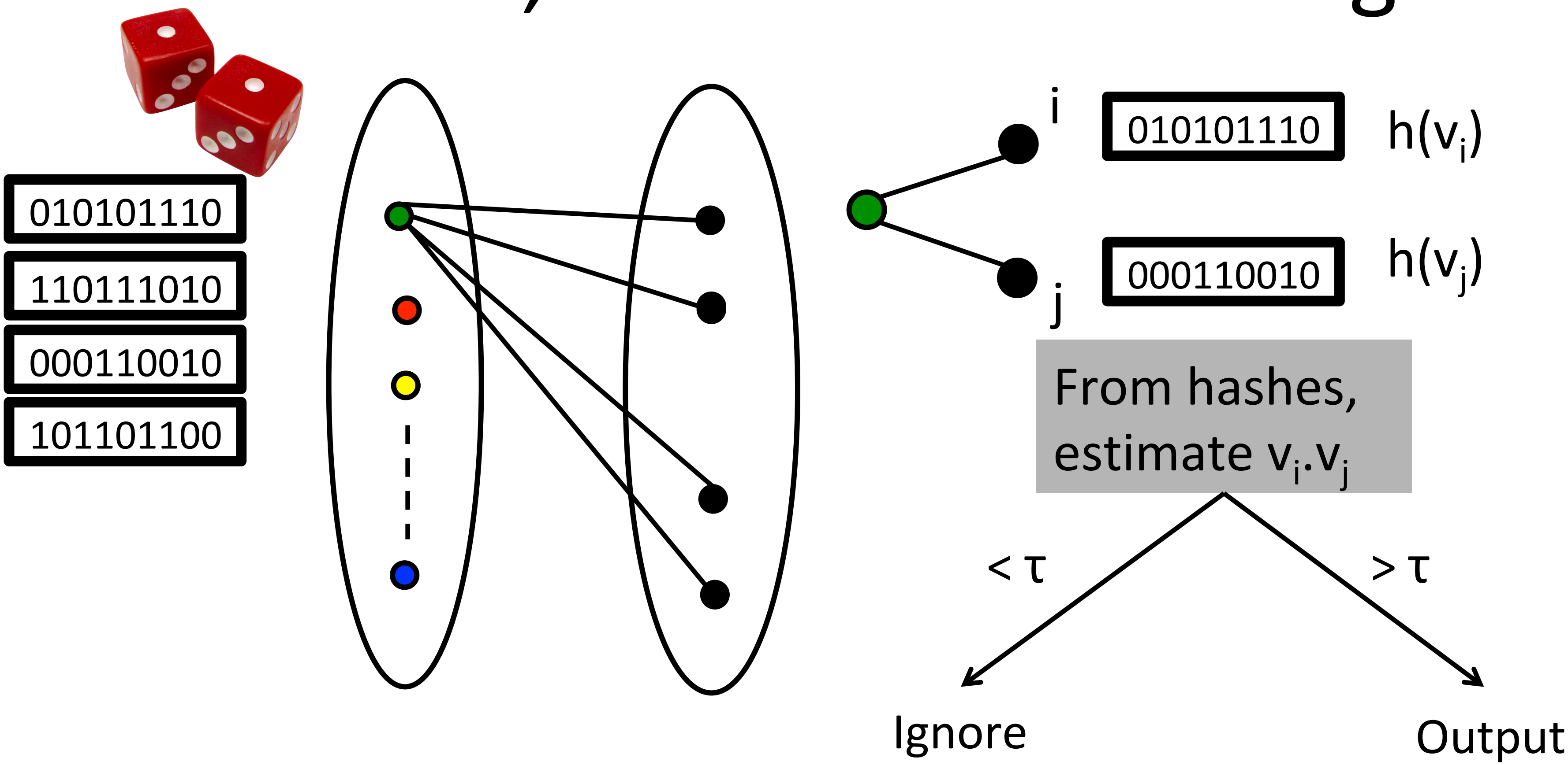
- Each processor on right computes $h(v_i)$
- Using pseudorandom generators, O(nnz(A)) communication

WHIMP, Round 2: Getting hashes



- Each vertex on left collects relevant hashes
- All edges send a hash
- Communication = $O(\tau^{-2}\ \mathrm{nnz}(A) \log n)$

WHIMP, Round 3: The Wedges



- Only output wedges that give similar vectors!
- Comm = (# τ-similar pairs) X (τ log n)

Some work required

THEOREM 4.1. *Given input matrices A, B and threshold τ, denote the set of index pairs output by WHIMP algorithm by S. Then, fixing parameters $\ell = \lceil c\tau^{-2}\log n \rceil$, $s = (c(\log n)/\tau)$, and $\sigma = \tau/2$ for a sufficiently large constant c, the WHIMP algorithm has the following properties with probability at least $1 - 1/n^2$:*

- *[Recall] If $(A^T B)_{a,b} \geq \tau$, (a, b) is output.*
- *[Precision] If (a, b) is output, $(A^T B)_{a,b} \geq \tau/4$.*
- *The total computation cost is $O(\tau^{-1}\|A^T B\|_1 \log n + \tau^{-2}(\mathrm{nnz}(A) + \mathrm{nnz}(B))\log n)$.*
- *The total communication cost is $O((\tau^{-1}\log n)\|[A^T B]_{\geq \tau/4}\|_1 + \tau^{-2}(\mathrm{nnz}(A) + \mathrm{nnz}(B))\log n + m + n)$.*

Similar pairs output

Hashes

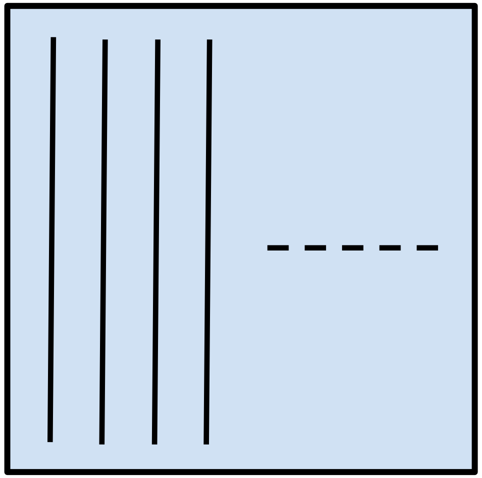
- Careful choice of parameters to get it to work in practice

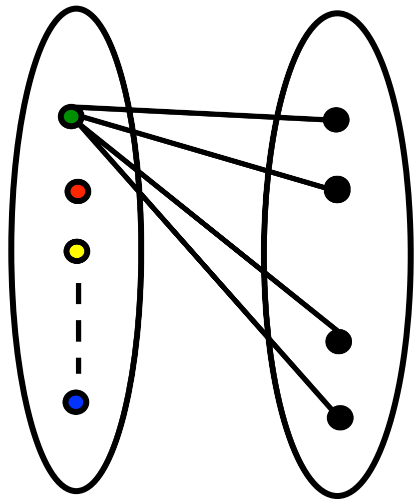
Evaluations

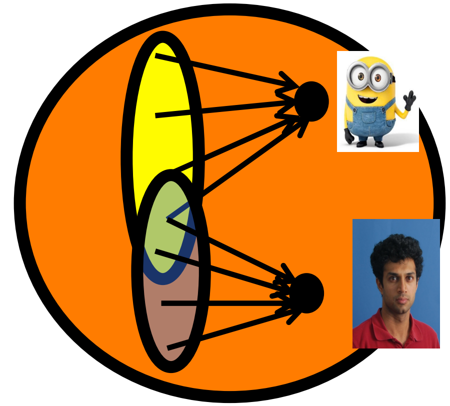
Dataset	Dimensions $n = d$	Size (nnz)	$\|A^T A\|_1$
friendster	65M	1.6B	7.2E9
clueweb	978M	42B	6.8E10
eu	1.1B	84B	1.9E11
flock	-	$O(100B)$	5.1E12

- Hard to validate!
- Stratified sample of vectors by degree (sparsity)
  - 1000 vectors for degree in $[10^i, 10^{i+1}]$
  - Full similarity compute for all of them

Major caveat!







- Prune all high degree vertices on left
  - Removing spammers, or those that follow too many
  - Removes < 5% of edges in real instances
- Removing dimensions that participate in too many vectors
- Reduces skew in communication

Total shuffle: $\tau = 0.2$

Dataset	WHIMP (TB)	DISCO est. (TB)
`friendster`	4.9	26.2
`clueweb`	90.1	247.4
`eu`	225.0	691.2
`flock`	287.0	18553.7

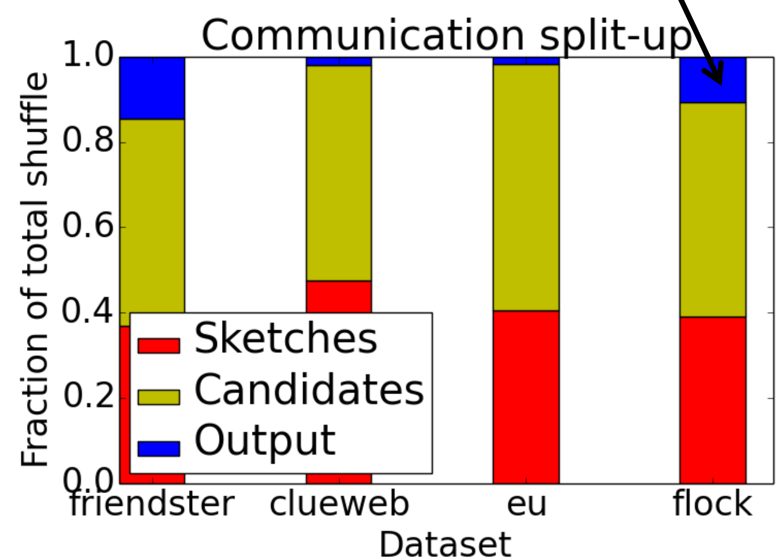
Infeasible to feasible

Communication

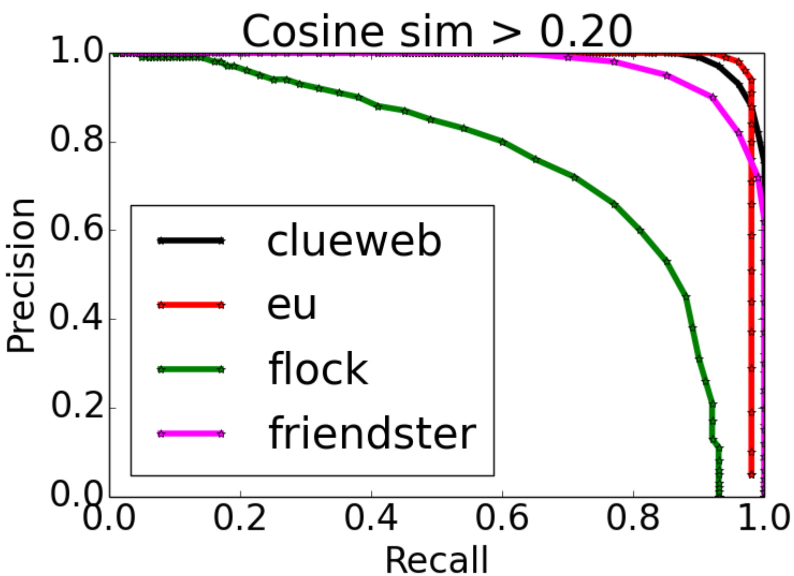
Dataset	WHIMP (TB)	DISCO est. (TB)
friendster	4.9	26.2
clueweb	90.1	247.4
eu	225.0	691.2
flock	287.0	18553.7

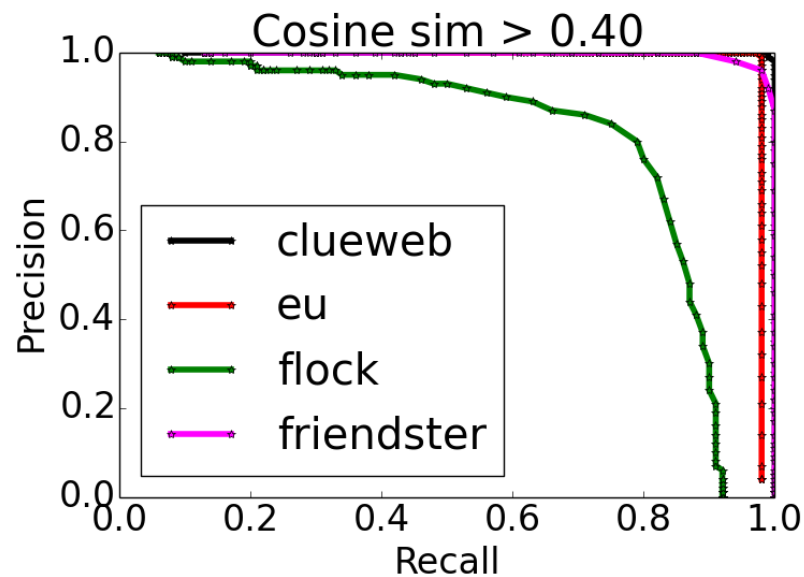
Infeasible to feasible

About 9X overhead over optimum

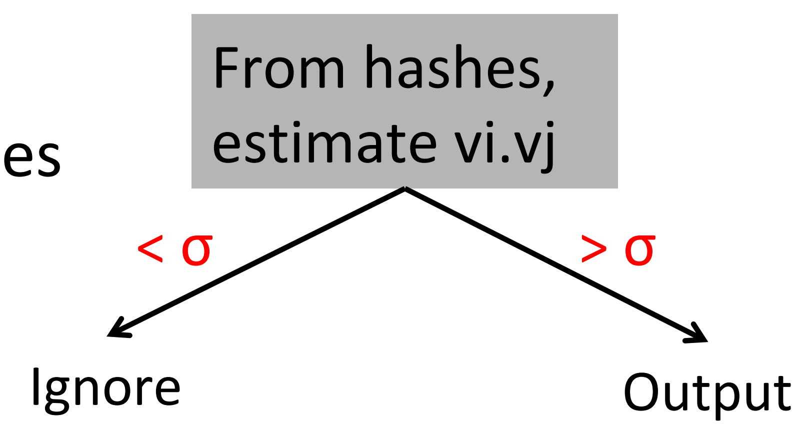


Precision-recall curves: τ = 0.2, 0.4





- Vary σ for precision-recall curves

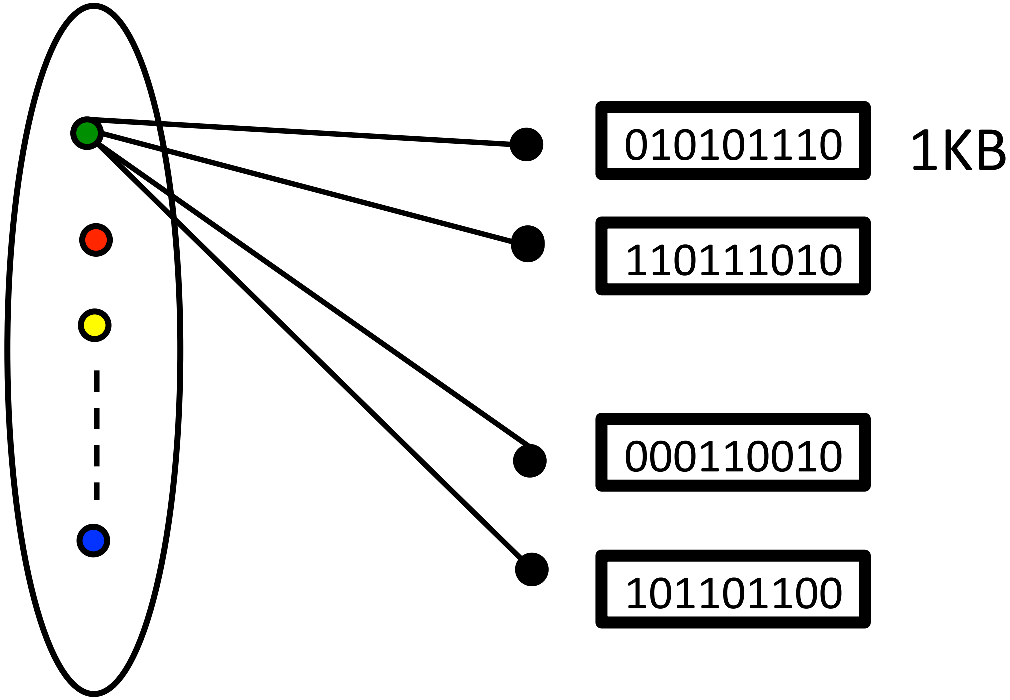


Miles to go before I sleep…

The skew problem?

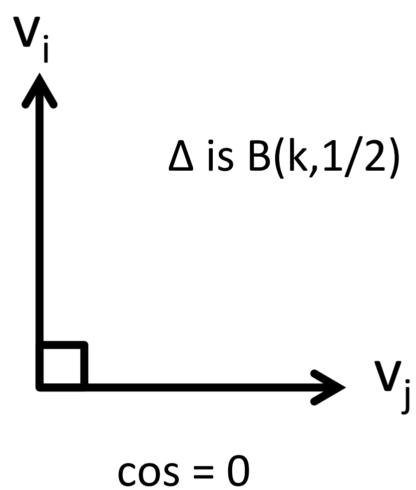
d = 10^6

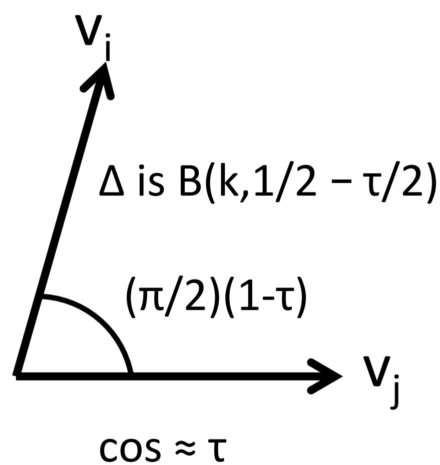
Comm
= 10^6 KB
= 1 GB

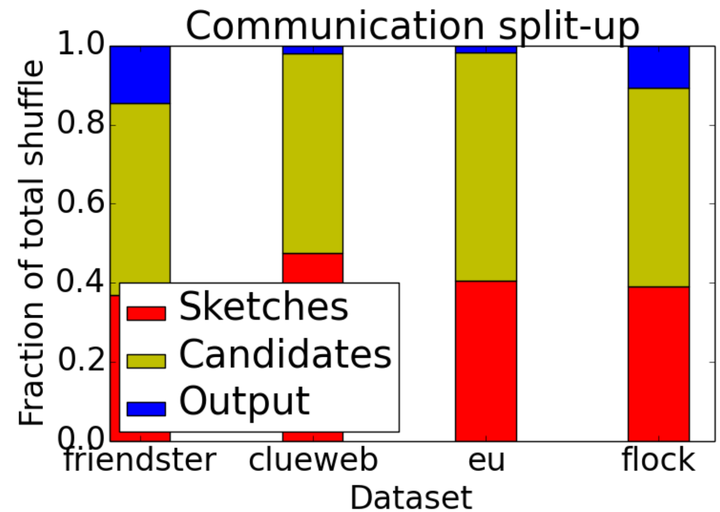


- Communication to node: O(d X hash size)
  - $O(d\ \tau^{-1} \log n)$, can be too much
- Alternate scheme to bound max communication?

Minhash alternative?

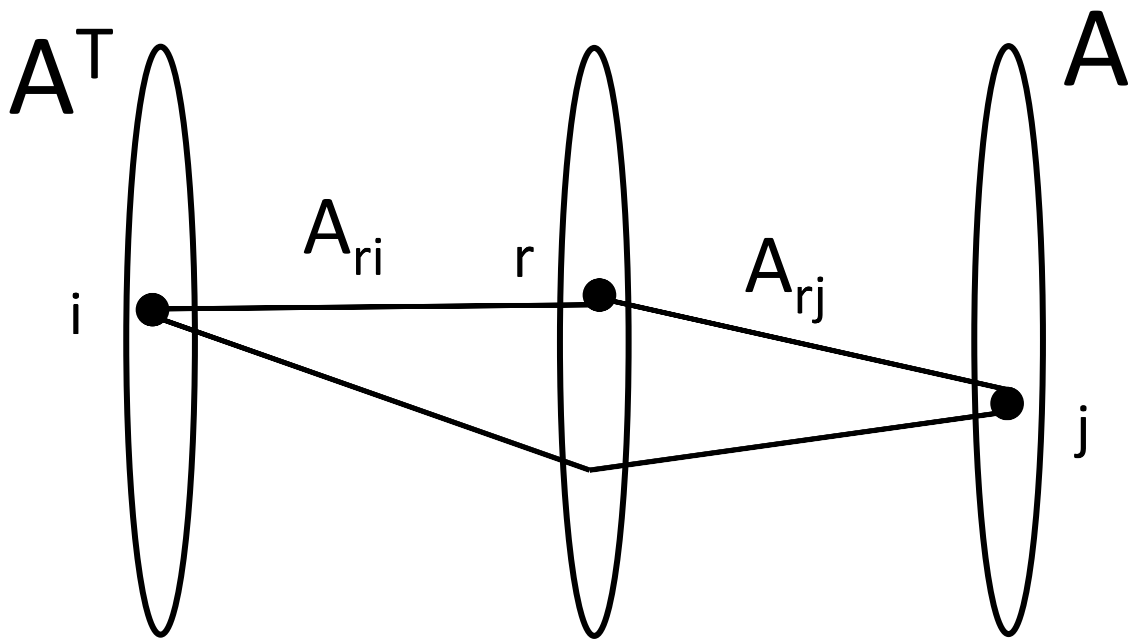






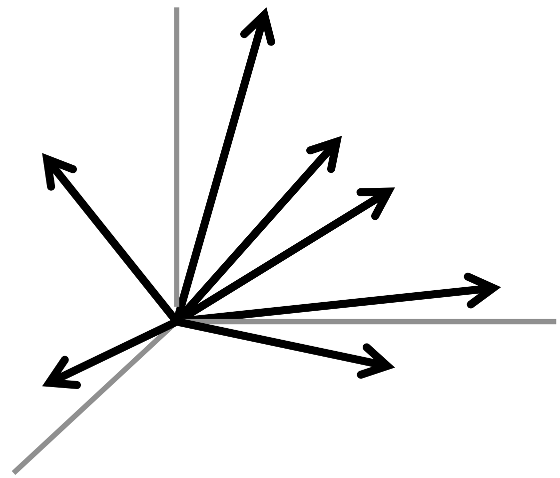
- 1KB (≈ 8000 bits) sketch barely distinguishes 0 from 0.1
- Better sketches? Even saving ½ in length would be useful

Uses of non-negativity?



- Power of Cohen-Lewis trick
- [Andoni-Razenshteyn 15, 16] Data dependent hashing
  - Using low dimensional structure

Finding large entries in matrix product?



- Find all large entries in product AB (or A^TA)
- What is the complexity of this problem?
  - Fine-grained complexity anyone?

Takeaways

- Similarity search/nearest neighbor is extremely relevant when sim values are closer to 0 than 1
  - And it is hard
- WHIMP deals with this regime using wedge sampling and hashing
- Big data required (minor?) rethink
- Systems solutions don’t always work


I DON'T ALWAYS THINK BIG DATA IS INTERESTING
BUT WHEN I DO, IT INVOLVES THEORY
memegenerator.net

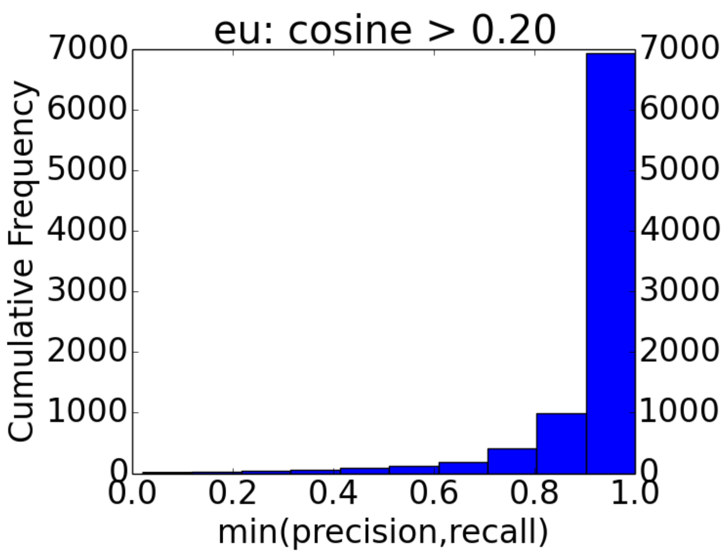
Evaluations

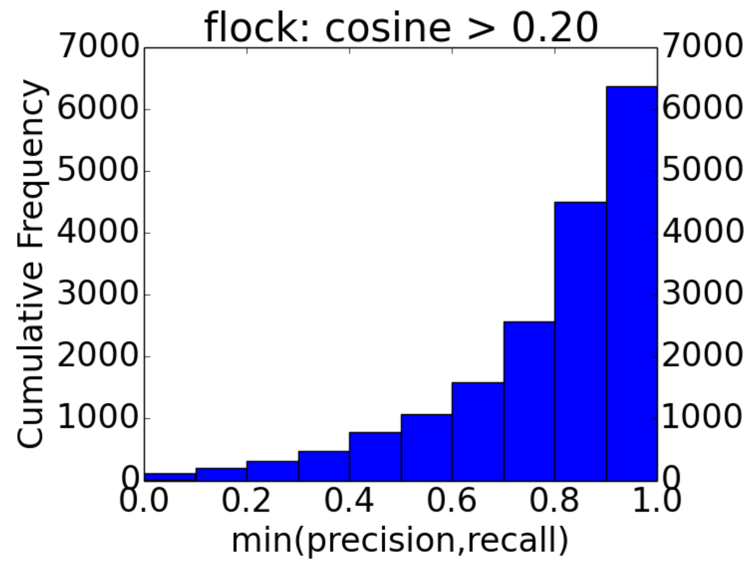
Dataset	Dimensions $n = d$	Size (nnz)	$\|A^T A\|_1$
friendster	65M	1.6B	7.2E9
clueweb	978M	42B	6.8E10
eu	1.1B	84B	1.9E11
flock	-	$O(100B)$	5.1E12

- Hard to validate!
- Stratified sample of vectors by degree (sparsity)
  - 1000 vectors for degree in $[10^i, 10^{i+1}]$
  - Full similarity compute for all of them

- Precision: is everything output similar?
- Recall: does algorithm output all similar pairs?

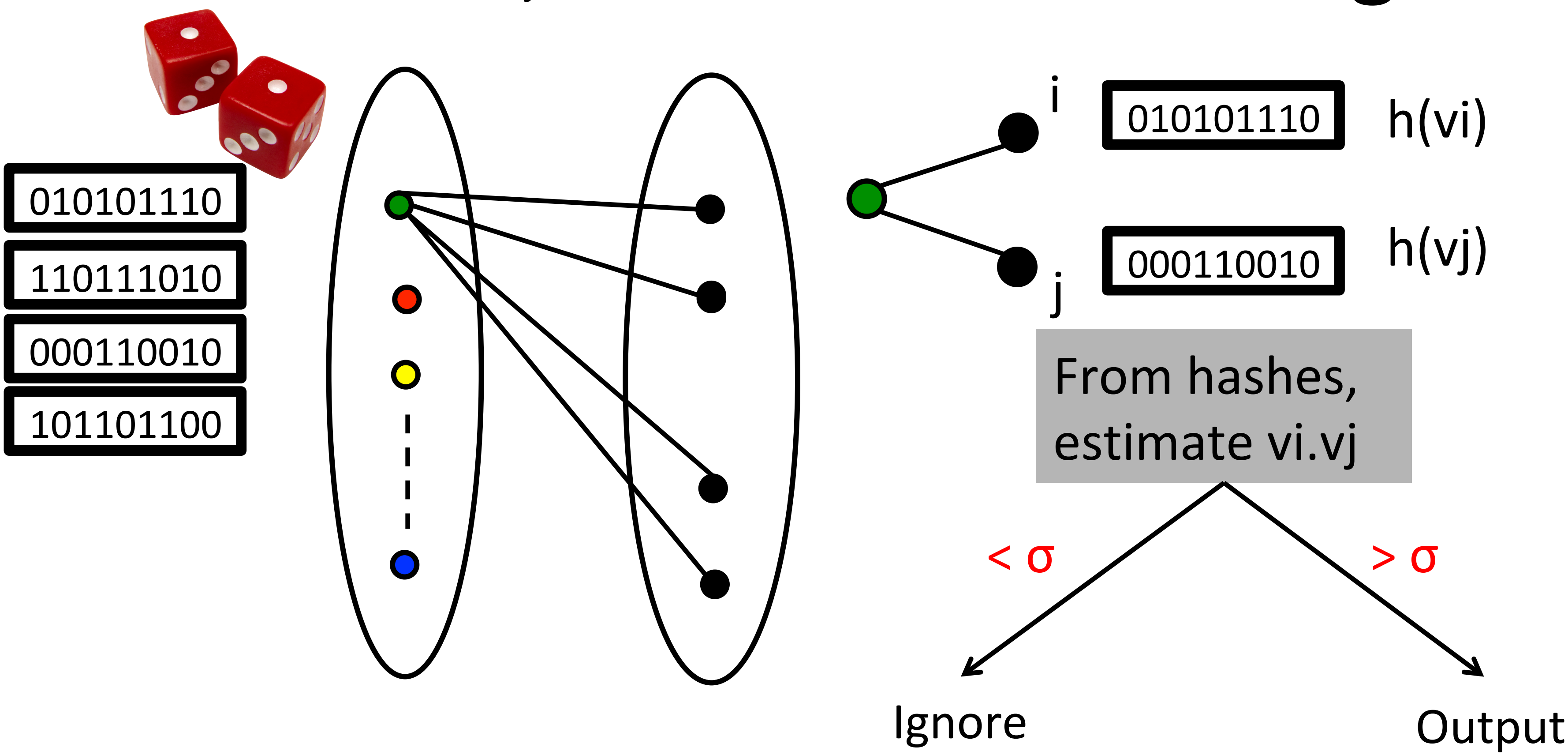
Per-user results: $\tau = 0.2$





- Accurate for most users
  - Important for recommendation applications

WHIMP, Round 3: The Wedges



- Normally, σ = τ
- Vary σ for precision-recall curves