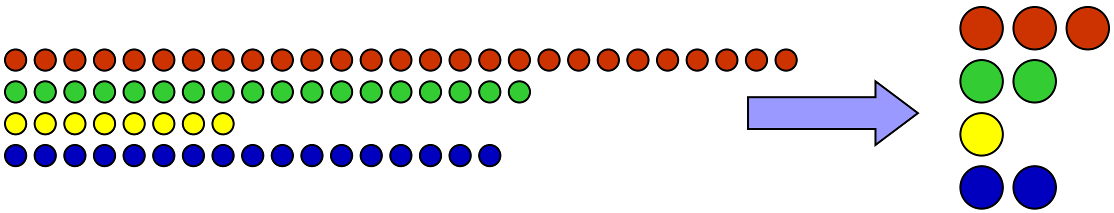
Sub-quadratic search for significant correlations



Graham Cormode

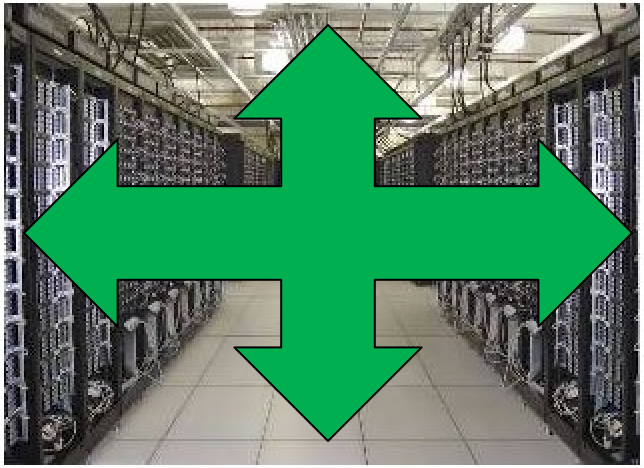
Jacques Dark

University of Warwick

G.Cormode@Warwick.ac.uk

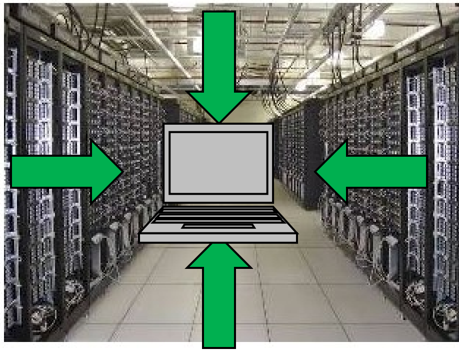
Computational scalability and “big” data

- Most work on massive data tries to **scale up the computation**
- Many great technical ideas:
  - Use many cheap commodity devices
  - Accept and tolerate failure
  - Move data to code, not vice-versa
  - MapReduce: BSP for programmers
  - Break problem into many small pieces
  - Add layers of abstraction to build massive DBMSs and warehouses
  - Decide which constraints to drop: noSQL, BASE systems



- Scaling up comes with its disadvantages:
  - Expensive (hardware, equipment, **energy**), still not always fast
- This talk is not about this approach!

Downsizing data



- A second approach to computational scalability: **scale down the data!**
  - A compact representation of a large data set
  - Capable of being analyzed on a single machine
  - What we finally want is small: human readable analysis / decisions
  - Necessarily gives up some accuracy: approximate answers
  - Often randomized (small constant probability of error)
  - Much relevant work: samples, histograms, wavelet transforms
- Complementary to the first approach: not a case of either-or
- Some drawbacks:
  - Not a general purpose approach: need to fit the problem
  - Some computations don't allow any useful summary

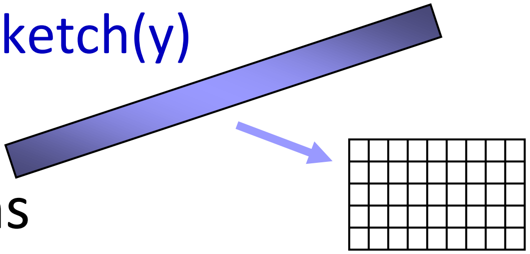
Outline for the talk

- An introduction to sketches (high level, no proofs)
- An application: Finding correlations among many observations

- There are many other (randomized) compact summaries:
  - Sketches: Bloom filter, Count-Min, AMS, Hyperloglog
  - Sample-based: simple samples, count distinct
  - Locality Sensitive hashing: fast nearest neighbor search
  - Summaries for more complex objects: graphs and matrices
- Not in this talk – ask me afterwards for more details!

What are "Sketch" Data Structures?

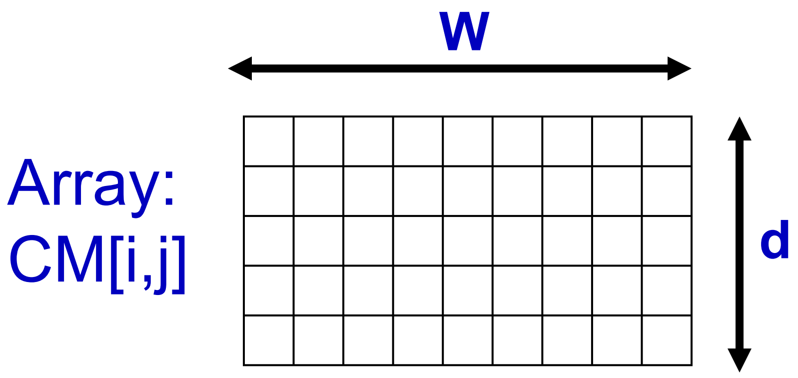
- Sketch is a class of summary that is a linear transform of input
  - Sketch(x) = Sx for some matrix S
  - Hence, Sketch(αx + βy) = α Sketch(x) + β Sketch(y)
  - Trivial to update and merge



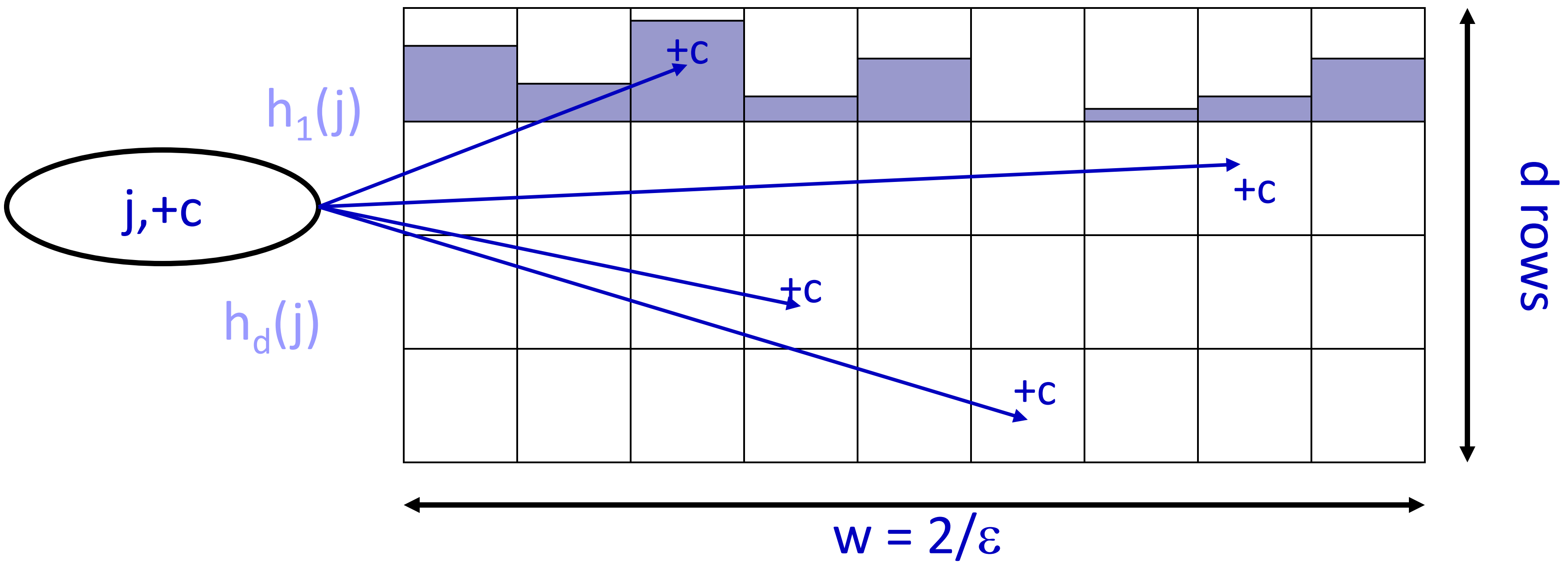
- Often describe S in terms of hash functions
  - S must have compact description to be worthwhile
  - If hash functions are simple, sketch is fast
- Analysis relies on properties of the hash functions
  - Seek "limited independence" to limit space usage
  - Proofs usually study the expectation and variance of the estimates

Sketches

- Count Min sketch [C, Muthukrishnan 04] encodes item counts
  - Allows estimation of frequencies (e.g. for selectivity estimation)
  - Some similarities to Bloom filters
- Model input data as a vector x of dimension U
  - Create a small summary as an array of $w \times d$ in size
  - Use d hash function to map vector entries to [1..w]



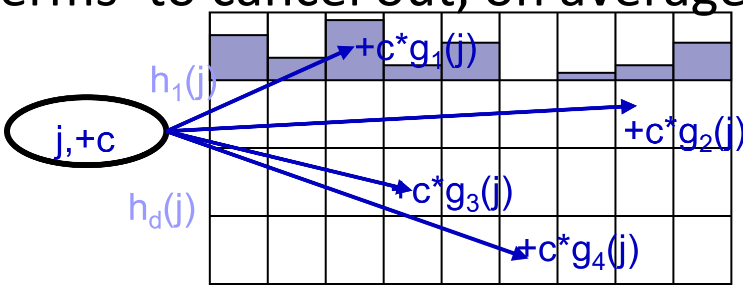
Count-Min Sketch Structure



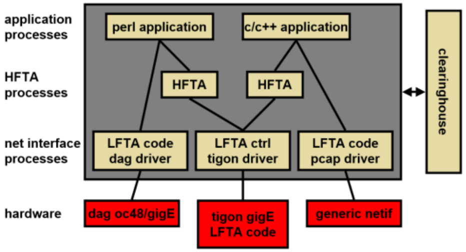
- Update: each entry in vector x is mapped to one bucket per row.
- Merge two sketches by entry-wise summation
- Query: estimate x[j] by taking $\min_k CM[k,h_k(j)]$
  - Guarantees error less than $\varepsilon||x||_1$ in size $O(1/\varepsilon)$
  - Probability of more error reduced by adding more rows

Sketching for Euclidean norm

- AMS sketch presented in [Alon Matias Szegedy 96]
  - Allows estimation of Euclidean norm of a sketched vector
  - Leads to estimation of (self) join sizes, inner products
  - Data-independent dimensionality reduction ('Sparse Johnson-Lindenstrauss lemma')
- Here, describe (fast) AMS sketch by generalizing CM sketch
  - Use extra hash functions $g_1 \dots g_d$ $\{1 \dots U\} \rightarrow \{+1,-1\}$
  - Now, given update $(j,+c)$, set $CM[k,h_k(j)]$ += $c*g_k(j)$
- Estimate squared Euclidean norm $= \text{median}_k \sum_i CM[k,i]^2$
  - Intuition: g_k hash values cause 'cross-terms' to cancel out, on average
  - The analysis formalizes this intuition
  - median reduces chance of large error

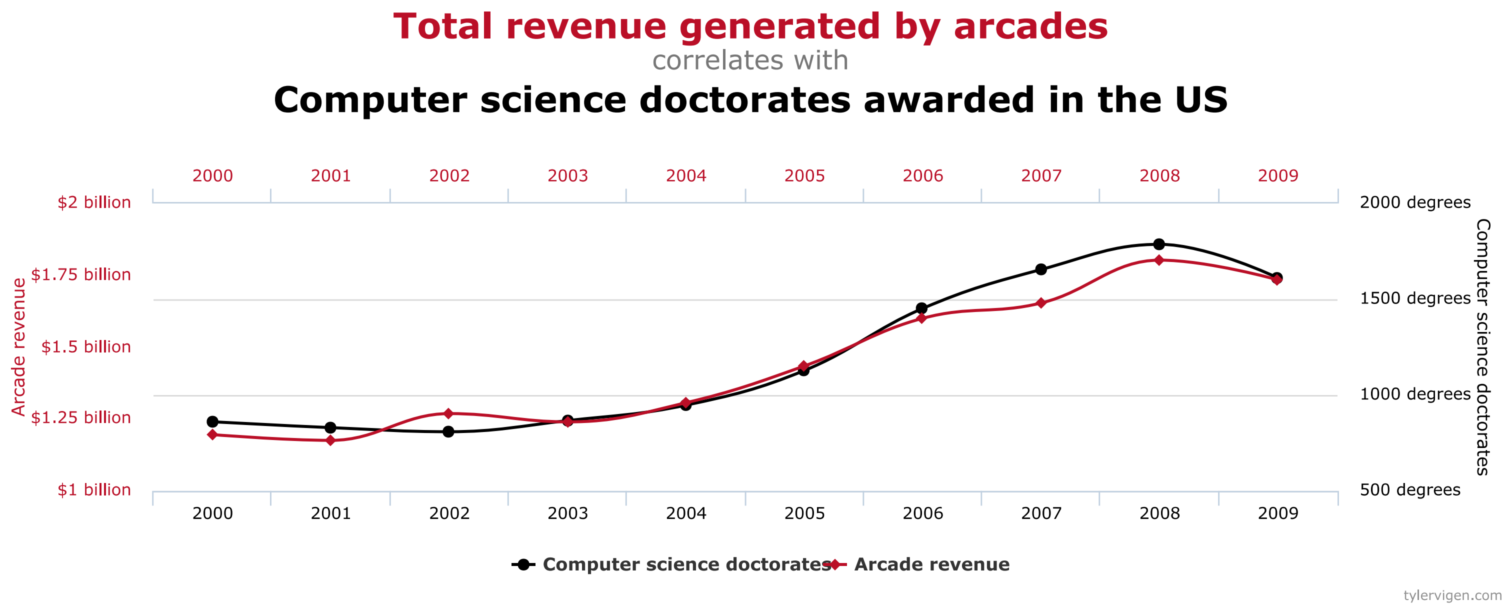


Sketches in practice: Packet stream analysis



- AT&T Gigascope / GS tool: stream data analysis
  - Developed since early 2000s
  - Based on commodity hardware + Endace packet capture cards
- High-level (SQL like) language to express continuous queries
  - Allows "User Defined Aggregate Functions" (UDAFs) plugins
  - Sketches in gigascope since 2003 at network line speeds (Gbps)
  - Flexible use of sketches to summarize behaviour in groups
  - Rolled into standard query set for network monitoring
  - Software-based approach to attack, anomaly detection
- Current status: latest generation of GS in production use at AT&T
  Also in Twitter analytics, Yahoo, other query log analysis tools

Looking for Correlations



tylervigen.com/spurious-correlations

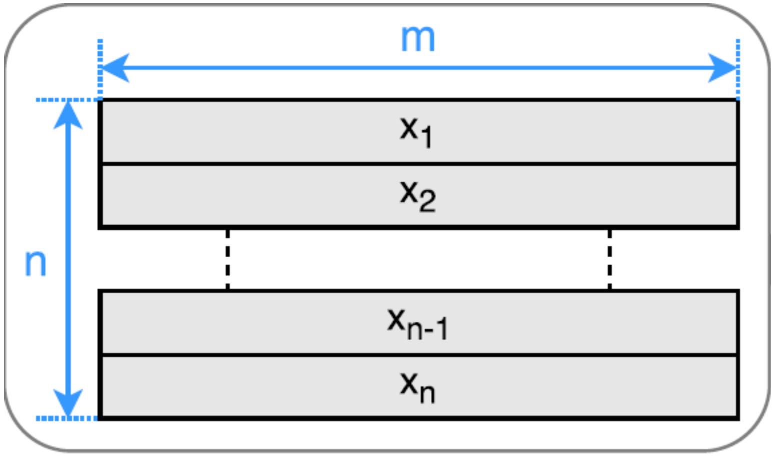
- Given many (time) series, find the highly correlated pairs
  - And hope that there aren't too many spurious correlations…
- Input model: we have m observations of n time series
  - One new observation of all series at each time step

Computing the Correlation

- Stats refresher: time series modeled as random variables X, Y
  - The covariance $Cov(X,Y) = E[XY] – E[X] E[Y] = E[(X–E[X])(Y – E[Y])]$
  - The correlation is covariance normalized by standard deviations $Cor(X,Y) = Cov(X,Y)/\sigma(X)\sigma(Y)$
- If we had all the time (and space) in the world:
  - Compute a vector $x = 1/\sigma(X)\ [\ X_1 – \mu_x, X_2 – \mu_x \ldots X_m – \mu_x]$
  - For all x, y pairs, compute $Cor(X,Y) = x \cdot y$ (vector inner product)
  - Time taken: $O(nm)$ preprocessing $+\ O(n^2m)$ for pair computations
  - Can write as a matrix product MM^T, where M is normalized data
- $O(nm)$ not so bad: linear in the size of the input data
- $O(n^2m)$ is bad: grows quadratically as number of series increases
  - Can't do better if many pairs are correlated
  - But in general, most pairs are uncorrelated – so there is hope

Sketching version 1

- Can apply sketching to the data
  - Replace each series with a sketch of the series
  - Can use linear properties of sketches to update and zero mean even as new observations are made

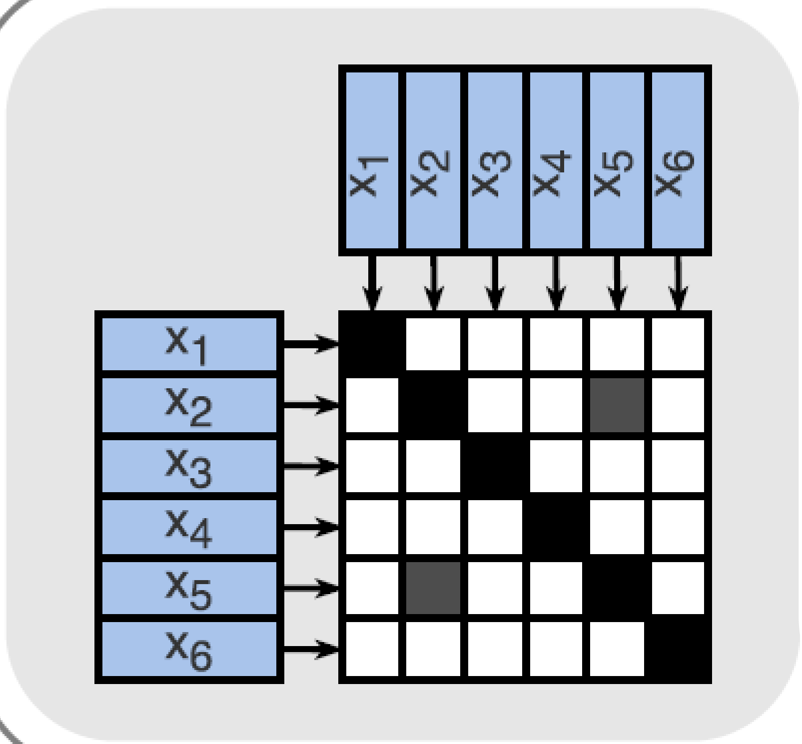


- Obtain approximate correlations (with error ε)
  - Time cost reduced to $O(mn + n^2 b)$, with $b = O(1/\varepsilon^2)$
  - Better, but still quadratic in n!

Sketching version 2

- Need a smarter data structure to find large correlations quickly
  - If most pairs are uncorrelated, no use testing them all
- Simple idea: bunch series into groups, add them up in groups
  - If no correlations in two groups, their sum should be uncorrelated
  - If there is a correlation, the sum should remain correlated
- Challenge:
  1. Turn the “should be”s into more precise statements!
  2. How to find the correlated pair(s) from correlated groups?
- Solution outline: a combination of sketching + group testing
  1. Use some standard statistical techniques to analyze probabilities
  2. Use some nifty coding theory to “decode” results

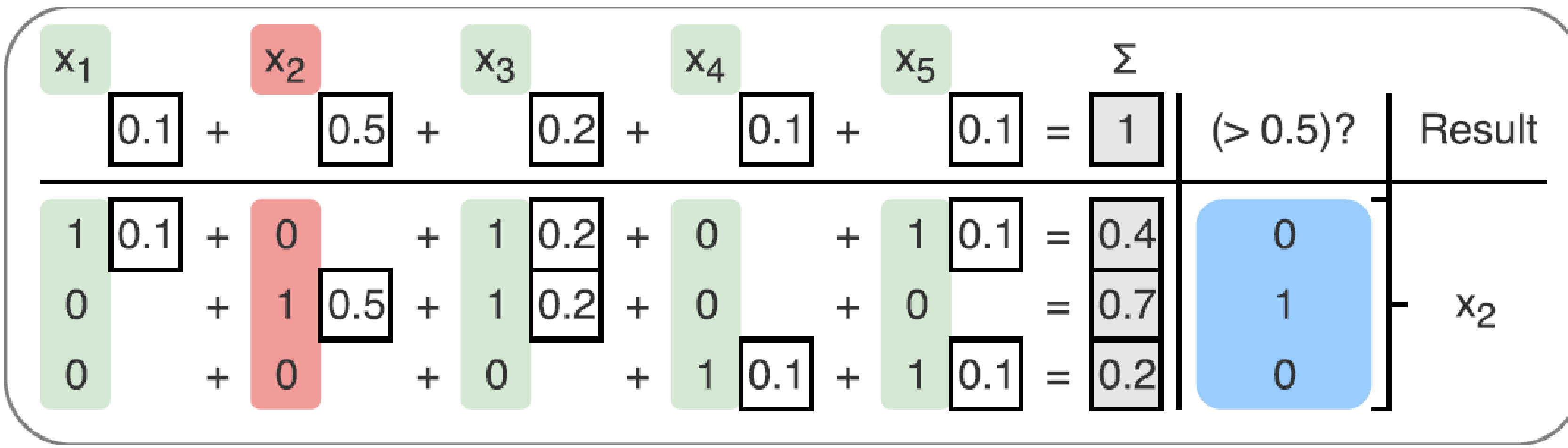
Bucketing the sketches



- Create a smaller correlation matrix
  - Randomly permute the indexing of the series
  - Sum together the series placed in the same bucket
  - Subtract the effect of diagonal elements (self-correlations)

Coding up the buckets

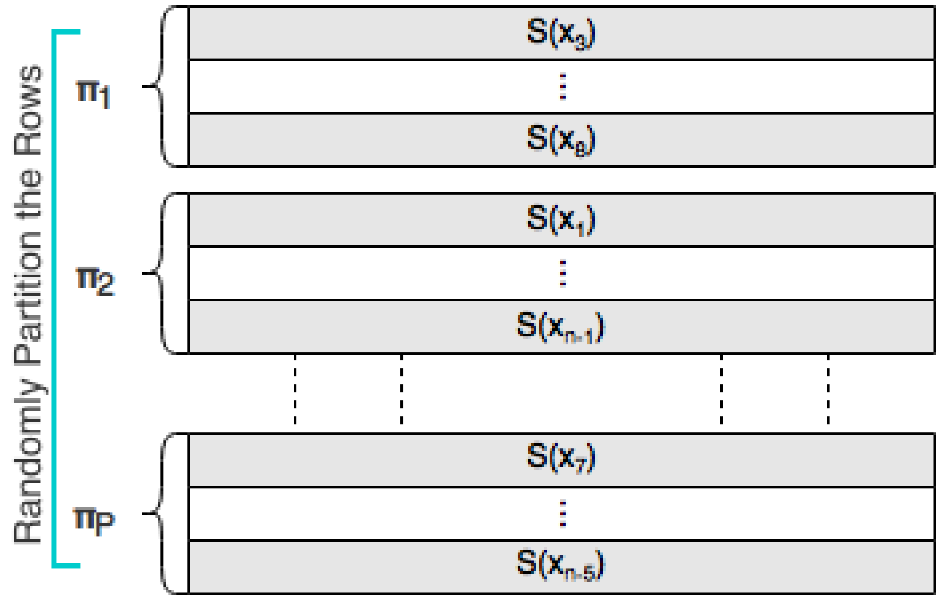
- For each pair of buckets, do additional coding to find which entries were heavy (group testing within buckets)
  - Repeat the sketching with different subsets of series
- Intuition: use a Hamming code to mask out some entries
  - See which combinations are “heavy” to identify the heavy index



- Rather vulnerable to noise from sketching, collisions

More sketching! Sketch all the things!

- Improvement 1: use sketching ideas within the buckets!
  - Randomly multiply each series in the bucket by +1 or -1
  - Decreases the chance of errors (in a provable way)



More coding! Code all the things!

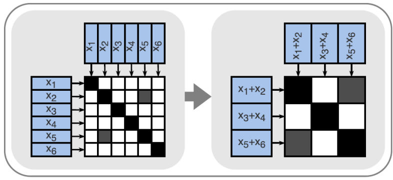
- Improvement 2: Error correcting codes to recover (noisy) pairs

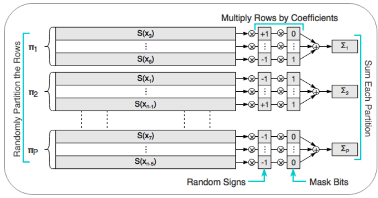
Correlation Row i	1	1	1	1	0	1	0	1
Correlation Column j	1	0	0	1	1	1	1	1
Bucket (p, q)	1.22	0.34	-0.20	1.17	0.02	-0.67	-0.28	-0.44
(> 0.5 or < -0.5?)	1	0	0	1	0	1	0	0
	✓	✓	✓	✓	✓	✓	✓	✗

- Care needed in code choice: each extra bit = more sketches
  - Only need to code the low-order bits of the permuted (i, j)
  - The high order bits are given by the bucket id
  - Can just store the random permutation of ids explicitly
  - Use Low Density Parity-Check codes: simple & work with sketches

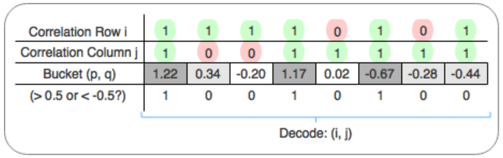
Putting it all together

- Mistakes still happen: from sketches, collisions etc.
  - Repeat the process a few times in parallel
  - Only report pairs found at least half the time
  - Makes false positives vanishingly small, recall is high



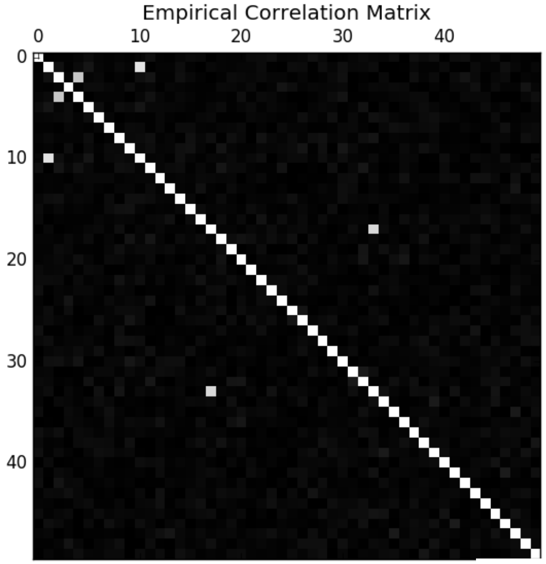


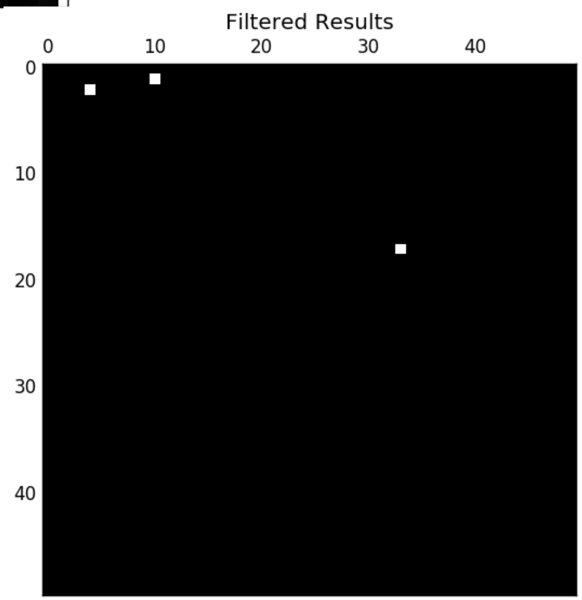
- Proof needed: Formal analysis of correctness to show:
  - Good chance that each heavy pair is isolated in a bucket
  - Noise from colliding pairs is small
  - Sketches for the bucket are (mostly) correct



- Assumptions: if small correlations are polynomially small, not too many large correlations, the space is subquadratic
  - And fast: sketch computations done via fast matrix multiply

Proof-of-concept experiments





- Tests on synthetic data
  - 50 vectors of length 1000
  - Sketches size 120
  - 10 buckets, 10 repetitions
- A few “planted” correlations
  - Test threshold 0.35
- Can recover significant correlations, miss some close to the boundary
  - Experiments ongoing!

Caveats and Cautions

- Randomized sketches can be powerful and effective, but they:
  - Don't give the exact answer
    (so not widely implemented or used)
  - Tend to be special purpose
    (so used for specific important problems)
  - Require some new ways of thinking
    (so take some getting used to)

- Some resistance to the randomness– can be argued against:
  - Want the exact answer? Most large data is highly noisy
  - Hard to debug? Randomized algorithms are simple(ish), repeatable
  - Want determinism? Hash tables are everywhere, caching, solar rays

Summary

- There are two approaches in response to growing data sizes
  - Scale the computation up; scale the data down
- Sketches are a useful general technique for data reduction
  - Developed for streaming algorithms (in computer science)
  - Related to compressed sensing, dimensionality reduction (math/stat)
- Continuing interest in applying and developing new theory
  - Always looking for new collaborators/students/postdocs

European Research Council
Established by the European Commission
EPSRC
Engineering and Physical Sciences Research Council
YAHOO! RESEARCH
Microsoft Research
THE ROYAL SOCIETY

Ad: The Alan Turing Institute in London