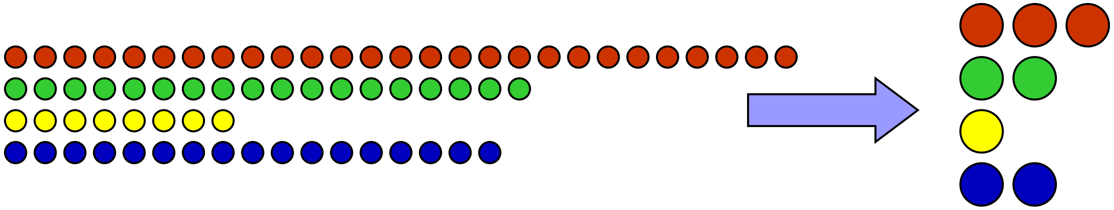
Randomized Algorithms in Databases



Graham Cormode

University of Warwick

G.Cormode@Warwick.ac.uk

Why Randomize?

- Thanks to cluster computing, we can scale up more computations
  - **Process all the transactions!**
  - **Count all the clicks!**
  - **Analyze all the analytics!**

- But this is not always necessary or desirable - sometimes, just need a quick estimate
  - Faster, less energy than doing the exact computation
  - Use the estimate to decide whether/how to do the exact thing
  - Allow real-time response on commodity PC (versus batch on cluster)
- Often achievable via algorithms that are randomized

Caveats and Cautions

- Randomized algorithms are powerful and effective, but:
  - Don't give the exact answer
    (so often not accessed via query languages)
  - Tend to be special purpose
    (so used for specific important problems)
  - Require some new terminology
    (so take some getting used to)

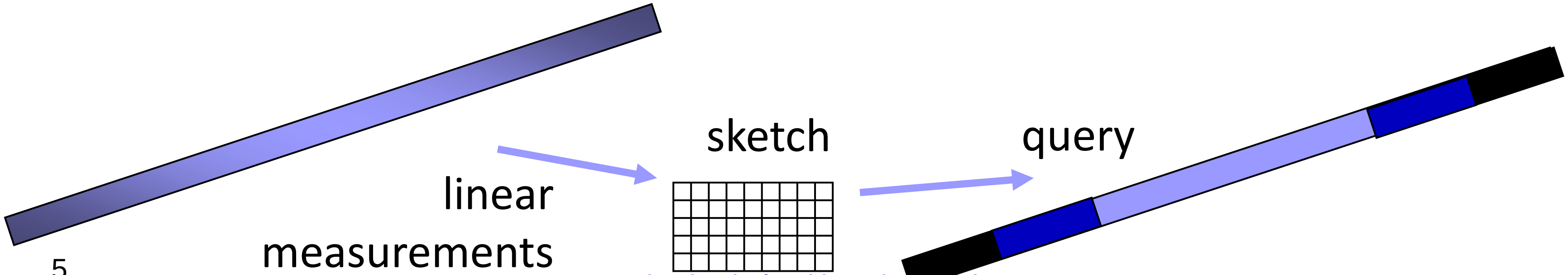
- Some resistance to randomization – can be argued against:
  - Want the exact answer? Most large data is highly noisy
  - Hard to debug? Randomized algorithms are simpler, repeatable
  - Want determinism? Hash tables are everywhere, caching, solar rays

Outline for the talk

- Some examples of ides and terminology (high level, no proofs)
  - Sampling based: simple samples, count distinct
  - Sketch based: Bloom filter, Count-Min, AMS
  - (Summaries for more complex objects): graphs and matrices
- There are also lower bounds: limitations of what we can do
  - No free lunch
- Current trends and future challenges for compact summaries
- Many abbreviations and omissions (histograms, wavelets, ...)

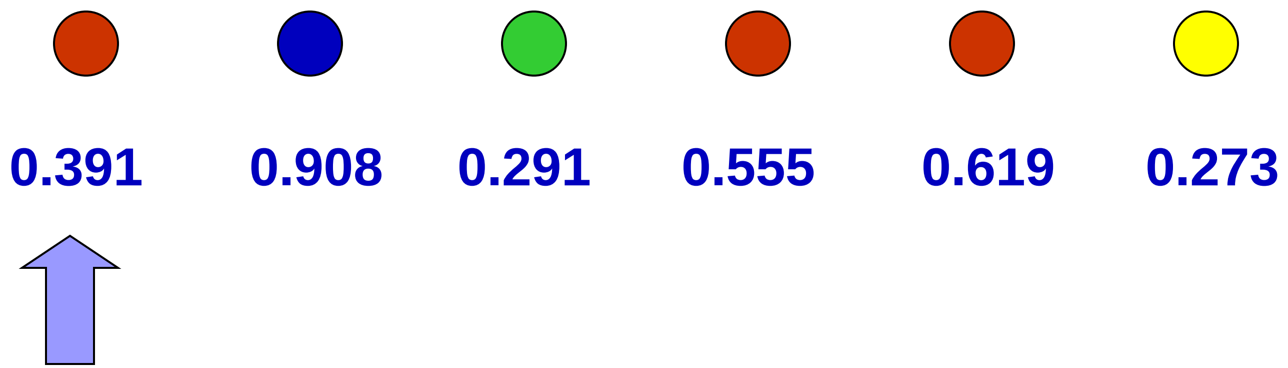
Menu for the day

- Different randomized techniques are task-specific
  - Random samples: a small representative subset of the data
    - Many uses in current and emerging databases
  - Bloom filter: summarize a set in a bit-efficient manner
    - Used for browser databases of malware-hosting sites
  - Count-Min/Count sketch: summarize a set of frequencies
    - Used in stream DBs to identify heavy source/destination combos
  - Hyperloglog: estimate cardinality of sets
    - Used by ad networks to track user demographics in logs



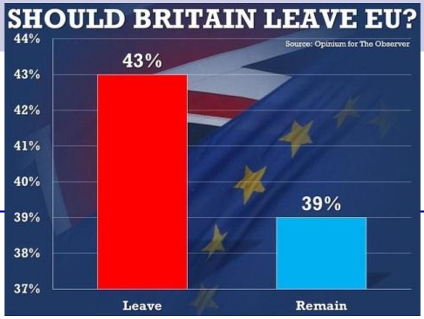
1. Min-wise Sampling

- Fundamental problem: sample m items uniformly from data
  - Allows evaluation of query on sample for approximate answer
  - Challenge: don't know how large total input is, so how to set rate?
- For each item, pick a random fraction between 0 and 1
- Update: store item(s) with the smallest random tag [Nath et al.'04]



- Each item has same chance of least tag, so it is uniform
- Can run on multiple inputs separately, then merge

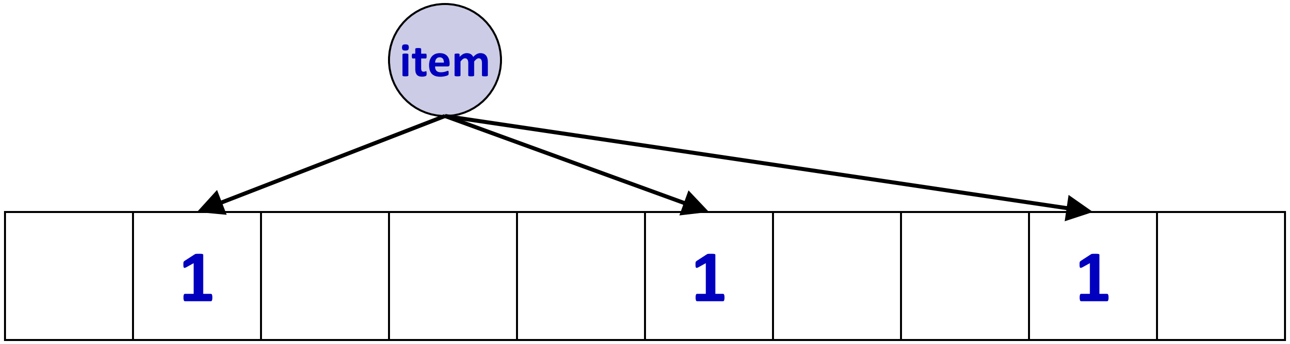
Uses of Samples in DB



- Random samples are used ubiquitously (50+ years of CS history)
  - Every opinion poll is a random sample (with confidence bounds)
- Query Planning:
  - Use samples to estimate the cost of a particular query plan
    - E.g. estimate selectivity of a predicate, estimate join size
  - Some parts need different estimation approach (e.g. cardinality)
- Data Integration:
  - Use samples to compare attributes: are they similar in nature?
- Data Reduction:
  - Code/debug/test query/analytics on a small sample
- Current status: In databases (behind the scenes) for decades

2. Bloom Filters

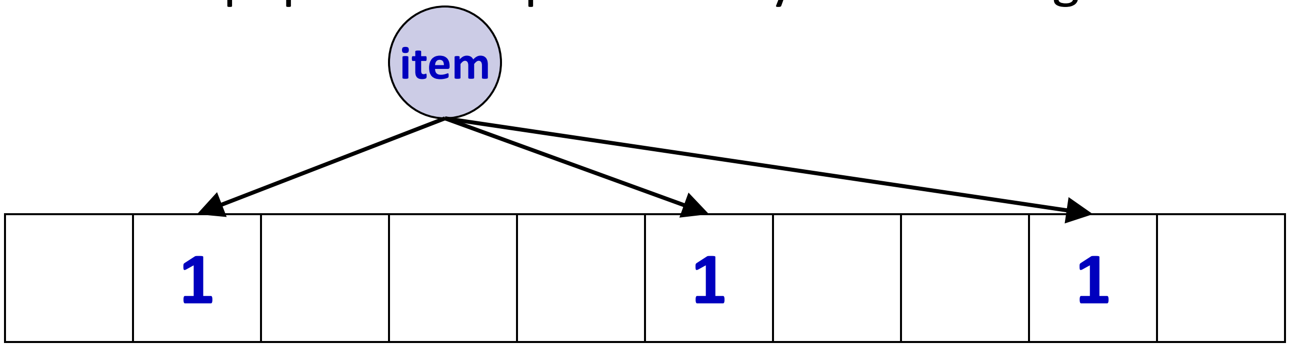
- Bloom filters [Bloom 1970] compactly encode set membership
  - E.g. store a list of many long URLs compactly
  - k hash functions map items to m-bit vector k times
  - Update: Set all k entries to **1** to indicate item is present
  - Query: Can lookup items, store set of size n in O(n) bits
    - Analysis: choose k and size m to obtain small false positive prob



- Duplicate insertions do not change Bloom filters
- Can be merge by OR-ing vectors (of same size)

Bloom Filters Applications

- Bloom Filters widely used in “big data” applications
  - Many problems require storing a large set of items
- Can generalize to allow deletions
  - Swap bits for counters: increment on insert, decrement on delete
  - If representing sets, small counters suffice: 4 bits per counter
  - If representing multisets, obtain (counting) sketches
- Bloom Filters are an active research area
  - Several papers on topic in every networking conference…

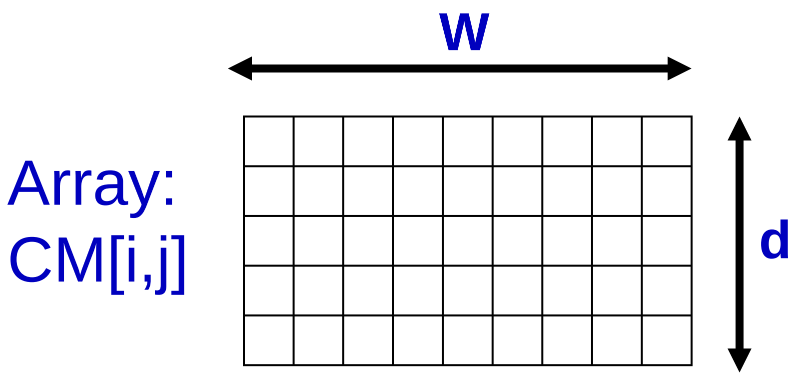


Bloom Filters in the wild

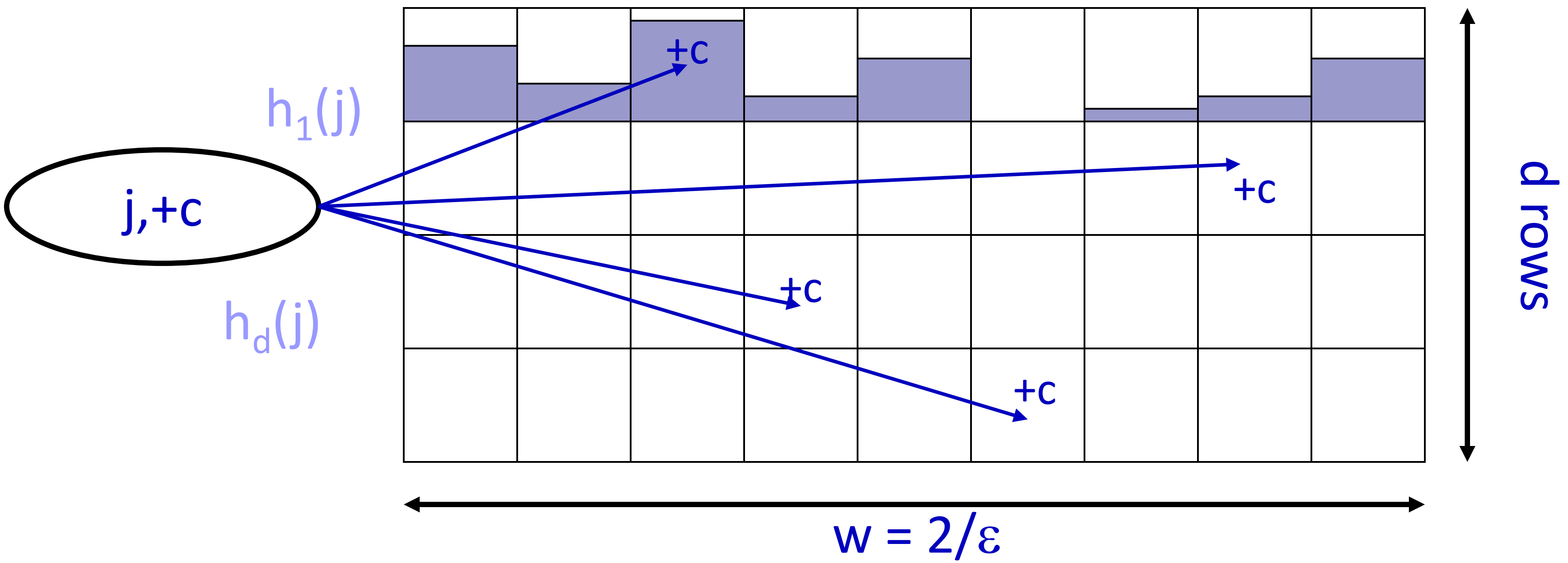
- Google: URL blacklisting in Chrome
  - Google compiles list of malicious URLs centrally
  - Exact storage of this list is high overhead for local storage:
    - ~1M URLS * ~100 bytes = 100MB
  - Bloom maths: use ~25 bits per item for 10^{-5} false positive rate
    - 3MB to store Bloom filter
  - Consequence of (false) positive: API call to Google to check URL
    - Effect of Bloom filter is to reduce costly API calls
- Current status: use of a variant encoding of hash values
  - A variant (randomized) algorithm

3. Count-Min Sketch

- Count Min sketch [C, Muthukrishnan 04] encodes item counts
  - Allows estimation of frequencies (e.g. for selectivity estimation)
  - Some similarities in appearance 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]

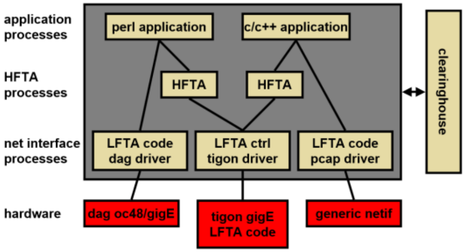


Count-Min Sketch Structure



- 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

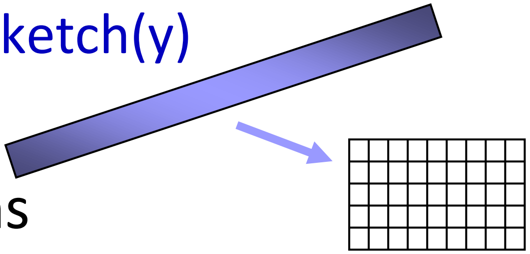
Application: Packet stream analysis



- 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, other query log analysis tools

Generalization: Sketch Structures

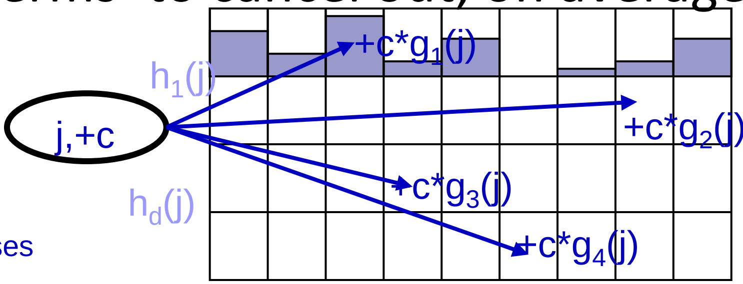
- Sketch is a class of summary that is a linear transform of input
  - Sketch(x) = Sx for some matrix S
  - Hence, Sketch(αx + βy) = α Sketch(x) + β Sketch(y)
  - Trivial to update and merge



- 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

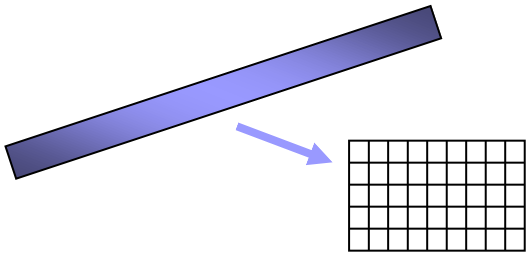
Sketching for Euclidean norm

- AMS sketch presented in [Alon Matias Szegedy 96]
  - Allows estimation of F_2 (second frequency moment)
  - Leads to estimation of (self) join sizes, inner products
  - Used at the heart of many streaming and non-streaming applications: achieves dimensionality reduction ('Johnson-Lindenstrauss lemma')
- Here, describe (fast) AMS sketch by generalizing CM sketch
  - Use extra hash functions $g_1 \ldots g_d$ $\{1 \ldots U\} \rightarrow \{+1,-1\}$
  - Now, given update $(j,+c)$, set $CM[k,h_k(j)]$ += $c*g_k(j)$
- Estimate squared Euclidean norm $(F_2) = \text{median}_k \sum_i CM[k,i]^2$
  - Intuition: g_k hash values cause 'cross-terms' to cancel out, on average
  - The analysis formalizes this intuition
  - median reduces chance of large error



Application to Large Scale Machine Learning

- In machine learning, often have very large feature space
  - Many objects, each with huge, sparse feature vectors
  - Slow and costly to work in the full feature space
- "Hash kernels": work with a sketch of the features
  - Effective in practice! [Weinberger, Dasgupta, Langford, Smola, Attenberg '09]
- Similar analysis explains *why:*
  - Essentially, not too much noise on the important features

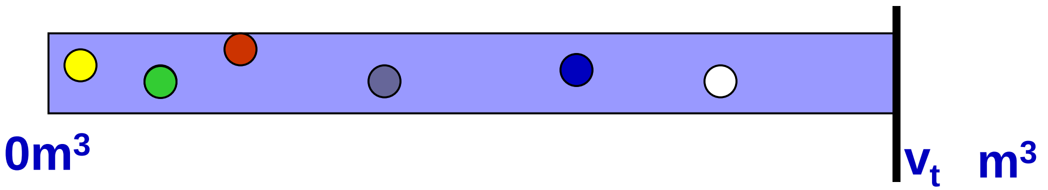


4. Cardinality Estimation / Count Distinct

- The cardinality of a set is the number of distinct items in the data
  - A fundamental quantity with many applications
  - COUNT DISTINCT estimation in DBMS
- Application: track online advertising views
  - Want to know how many distinct viewers have been reached
- Early approximate summary due to Flajolet and Martin [1983]
- Will describe a generalized version of the FM summary due to Bar-Yossef et. al
  - Known as the “k-Minimum values (KMV)” algorithm

KMV estimation algorithm

- Let m be the domain of the n distinct data elements
  - Each item in data is from $[1\ldots m]$
- Pick a random (pairwise) hash function $h: [m] \rightarrow [R]$
  - For R "large enough" (polynomial), assume no collisions under h



- Keep the t distinct items achieving the smallest values of $h(i)$
  - Note: if same i is seen many times, $h(i)$ is same
  - Let v_t = t'th smallest (distinct) value of $h(i)$ seen
- If $n < t$, give exact answer, else estimate $n' = tR/v_t$
  - $v_t/R \approx$ fraction of hash domain occupied by t smallest
  - Analysis sets $t = 1/\varepsilon^2$ to give ε relative error

Engineering Count Distinct

- Hyperloglog algorithm [Flajolet Fusy Gandouet Meunier 07]
  - Hash each item to one of $1/\varepsilon^2$ buckets (like Count-Min)
  - In each bucket, track the function $\max \lfloor \log(h(x)) \rfloor$
    - Can view as a coarsened version of KMV
    - Space efficient: need $\log \log m \approx 6$ bits per bucket
  - Take harmonic mean of estimates from each bucket
    - Analysis much more involved
- Can estimate intersections between instances
  - Make use of identity $|A \cap B| = |A| + |B| - |A \cup B|$
  - Error scales with $\varepsilon \sqrt{(|A||B|)}$, so poor for small intersections
    - Lower bound implies should **not** estimate intersections well!
  - Higher order intersections via inclusion-exclusion principle

Application: User tracking and profiling

- Domain: advertising on the web
  - Each user sees many ads
  - Each user has some set of attributes (possibly imputed):
    - Geolocation, age, sex, income, interests, education level
  - Advertisers want fast answers to slice-and-dice queries
    - How many 18-35 yo males with university education saw ad?
- Some companies (Aggregate Knowledge) used HLL for this
  - Faster and more compact than sifting through data on Hadoop
- Current status: AK using Amazon Redshift solution
  - This amount of data is now stored exactly and interrogated fast
  - HLL used with Google log analysis platform (and others)

What randomization can't do

- Some basic problems require storing the full input data
  - Determining whether or not a certain item was seen
  - Determining whether two large sets have any item in common
- So there can't be sketches for some problems we'd like to solve
  - Can't give accurate answer to cosine similarity of two vectors
  - Can't good accuracy for matrix multiplication
    - Can't tell whether the answer is zero, or close to zero
- Can make progress on these problems with weaker guarantees
  - But needs more care: will this work for the desired application?

Current Directions in DB Randomized algorithms

- Sparse representations of high dimensional objects
  - Compressed sensing, sparse fast fourier transform
- General purpose numerical linear algebra for (large) matrices
  - k-rank approximation, linear regression, PCA, SVD, eigenvalues
- Summaries to verify full calculation: a 'checksum for computation'
- Geometric (big) data: coresets, clustering, machine learning
- Graph/social net data: algorithms for matching, connectivity
- Use of summaries in large-scale, distributed computation
  - Build them in MapReduce, Continuous Distributed models
- Communication-efficient maintenance of summaries
  - As the (distributed) input is modified

Summary

- There are two approaches in response to growing data sizes
  - Scale the computation up; scale the data down
- The theory and practice of randomized algorithms has many guises
  - Sampling theory (since the start of statistics)
  - Streaming algorithms in computer science
  - Compressive sampling, dimensionality reduction… (maths, stats, CS)
- Continuing interest in applying and developing new theory
  - Ad: Postdoc & PhD studentships available at U of Warwick/ATI

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