SAMPLING ON THE FLY

General Framework

- Problem with massive data. Can be read from external memory, but too large to be stored in RAM.

- A natural approach

  Draw a sample (much smaller than the input) storable in RAM.

- Algorithm processes sample and yields good estimates of answer to whole problem.

Simplest kind of sampling is Uniform Sampling : every piece of data is equally likely to be picked.

Advantages :

- "Coins can be tossed" "blindly" - prior to a pass through the data. So, sample can be extracted in one (quick) pass through the data from external memory.

- Much recent work on things we can do with a uniform sample of fairly small size.

Drawback

Much we cannot do : Toy example - find the average of n given real numbers, where there is a lot of cancellation

(i.e., |average| $\ll$ average of |reals |.)

A central principle : No free lunch.

So, with a small sample size, only “global statistical properties” can be measured. Cannot hope to get “fine structure details”.

What can we do with uniform sampling ?

Example 1 There are n data objects. (n large). For each pair (i, j) of objects, we are given whether i, j are similar. Want to test the hypothesis "the objects can be divided roughly into k clusters", where each cluster contains similar objects.

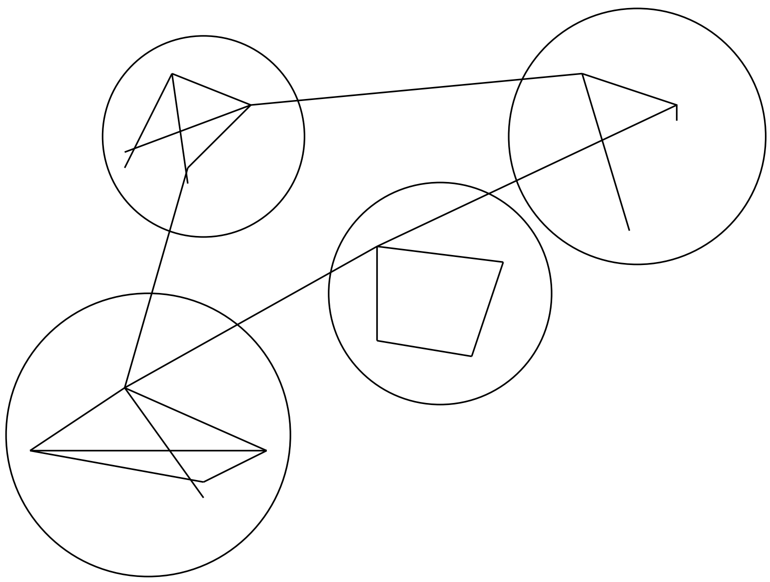
Also want a representative object from each cluster.

Often $k \ll n$, indeed, we assume $k \in O(1)$.

One formulation : what is the minimum number of new similarities we need to throw in so that there are k clusters with EVERY pair of objects inside each cluster being similar ?

Call this number ANS.

– Really a graph clique or coloring problem.



Recent Result

We can find ANS to within $\pm\epsilon n^2$ given a random subset of $O(1/\epsilon^4)$ objects and all their pairwise similarities.

No good if ANS$\ll n^2$. (No free lunch).

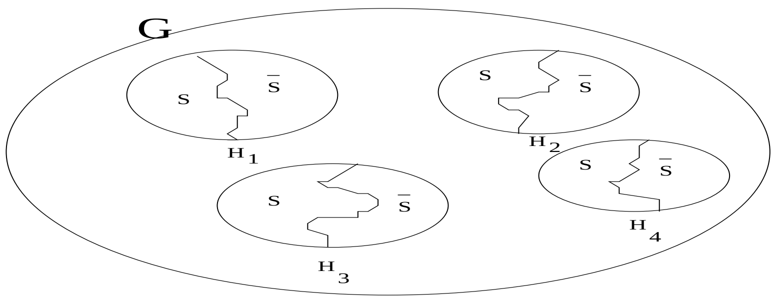
Easy Part : If there is a good clustering of the n objects, this induces a good clustering of sampled objects. – Traditional Statistical sampling arguments

Hard Part : Good clusterings of random subsets yield good clustering of the whole.

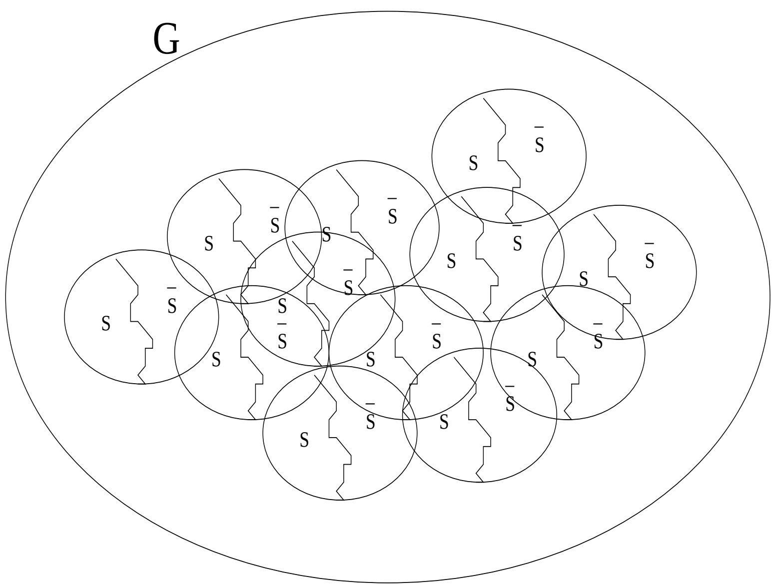
Sample is not overly optimistic.

This example is a very special case of a general result.

Easy Part A good clustering on the whole induces a good clustering on random sub-parts. An illustration for $k = 2$ - clustering into $S, \bar{S}$.



Hard Part How do we stich together clusterings in random sub-parts into a clustering of the whole ? In different random parts, the same object may be put into S or $\bar{S}$.



CONSTRAINT SATISFACTION PROBLEMS

n Boolean variables - $x_1, x_2, \ldots x_n$.

r constant.

m constraints given, each involving r literals.

(Many global variables. "Local" constraints, each involving only a fixed number).

Find a truth setting of $x_1, x_2, \ldots x_n$ which satisfies as many of the constraints as possible. Call this answer ANS.

The clustering problem above is a special cases of CSP with $r = 2$. So are many graph and Boolean problems.

Example Satisfy as many clauses as possible among :

$$(x_1 + \bar{x}_2 + \bar{x}_3)(x_4 + \bar{x}_1 + x_2)(x_7 + \bar{x}_3 + x_2) \ldots .$$

Recent Result 2

Given a uniform random subset R of $x_1, x_2, \ldots x_n$ of size c/ϵ^4 and all constraints involving only variables in R, we can find ANS to within $\pm\epsilon n^r$.

Answer to "induced" "sub-problem" on a random subset of variables is a good estimate of answer to whole problem.

Indeed, also a truth setting of all variables $x_1, x_2, \ldots x_n$ attaining this value can be constructed in $O(n)$ time from sample.

Literature

Arora, Karger, Karpinski: In poly time, we can approximate CSP to **additive error** ϵn^r.

[Can also be phrased as "we can solve dense instances with relative error ϵ.]

Goldreich, Goldwasser, Ron
r=2
First "constant" time algorithms for CSP problems for $r = 2$ with additive error ϵn^2. Good polynomial bounds (in $1/\epsilon$) on the size of the random sub-problem needed ("sample complexity") using structure of individual CSP problems...... Property Testing ......

Frieze, Kannan Poly time algorithms for all CSP problems (with fixed r) achieving additive error ϵn^r using Linear Algebra methods.

Recent Result 2 is due (independently to)

Alon, de la Vega, Kannan, Karpinski using Linear Algebra methods and

Andersson and Engebretsen using combinatorial methods.

Interesting Open problem

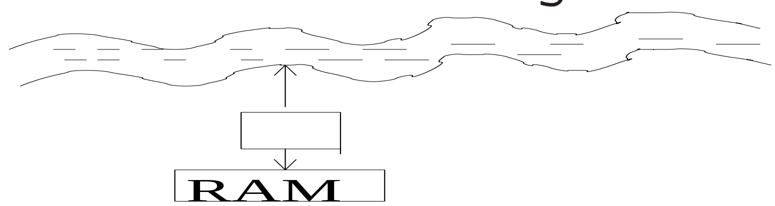
Can we say anything using the induced problem on a random sample of cn variables ?

In practice - may have a Constraint Satisfaction Problem with, say, 200 variables which is too large to solve (but not because of massive data). But we may have heuristics which effectively solve 100 variable problems. This is theoretically modeled as picking cn out of n variables.

Beyond Uniform Sampling

Frequency moments of streaming data

Given a "**STREAM**" S of data consisting of pairs (i, a) of integers.



Examples NetFlow information - Pair (i, a) logs a message with addresses (source-destination) i containing a packets. Possibly logs of many messages written in by multiple agents in arbitrary order.

Data Bases : (i, a) is an element of a relation.

Estimate "second moment" :

$\sum_i$ (total number of packets for address i)^2

More generally, p th moment - replace 2 by p above.

$p = 0$ distinct number of values of attribute i (in a database).

$p = 2$ and higher - degree of skew of the data..

Dewitt, Naughton, ... - useful in query optimization methods in database applications.

a may be negative too !!

Streaming Model

We are allowed to make one "leisurely" pass through the data (slowly flowing stream) -

For each bit (or block) read, can do poly($\log n$) amount of processing (where n is the total number of bits).

Also have RAM of poly($\log n$).

Thus most data must be seen once and thrown away.

We must sample on the fly - after poly($\log n$) processing, must decide if current block read should be preserved as part of sample (or "folded into running sample" in some way).

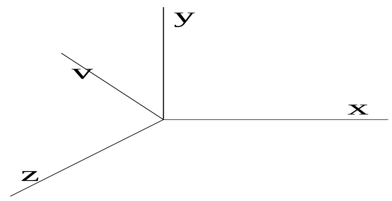
Recent Results ($p =$)0,1,2 moments can all be approximated to **relative error** ϵ in the streaming model.

Random Projections

Want $\sum_i$(total number of packets for address $i)^2$.

Let v_i = total number of packets for address i. Really want $\sum_i v_i^2 = |v|^2$.

Old Theorem In a random coordinate system, the first component of v will be almost exactly $|v|/\sqrt{n}$ in absolute value.



This first component $= v \cdot u$ for a random unit length vector u.

If we randomly generated u and kept in ($O(n)$ bits needed !!), then we can compute $v \cdot u$ as data streams - on seeing (i, a), just add $u_i a$ to running total $u \cdot v$.

Avoiding $O(n)$ space - use a pseudo-random u; just store seed for pseudo-random generator and regenerate u_i each time it is needed !!

Literature

Dewitt, Naughton, Schneider and Seshadri and Haas, Naughton, Seshadri, Stokes : Computation and application of frequency moments in databases.

Henzinger, Raghavan and Rajagopalan: The Streaming model.

Alon, Mathias, Szegedy : Relative error ϵ streaming algorithms for $p = 0$ (number of distinct values of an attribute) and $p = 2$ using only $O(\log n)$ RAM (all with only non-negative a).

Feigenbaum, Sampath Kannan, Strauss, Viswanathan $p = 1$ when a can be negative. (Useful for comparison of streams.) - Beware Cancellations.

Indyk (pseudo) Random Projections

Many other problems have been tackled in the streaming model.

For example, clustering problems :

Given a census, say, of incomes (n real numbers), cluster them into k ($k \ll n$) clusters. Some work on $O(1)$ factor approximations for $k-$median and $k-$center measures - Guha, Mishra, Motwani and O'Callahan.

Open : With one or more passes, can we do more realistic clustering ??

BEYOND A SINGLE PASS

Model

Data stored in external memory. We can read entire data by making a quick pass through the data.

Quick Pass - $O(1)$ time per block read. [Not poly($\log n$) as in Streaming model.]

Measure three resources

(i) number of (quick) PASSES.

(ii) Extra TIME (besides the passes)

(iii) Extra RAM

Early work on pass Vs space trade-offs - Munro and Patterson.

Why this modification to Streaming ?

Disk storage has grown rapidly and I/O time has shrunk. So, large data sets can be stored in external memory and read once or twice or a few times.

But, we cannot hope to apply even low polynomial time algorithms on the whole data.

Frequency moment problems above can be viewed as dealing with one $n-$ vector (or two if we are comparing two streams).

Matrices

Document-Term matrix A of a (large) collection of documents has :

$A_{ij} =$ number of occurrences of term j in document i or a function of that number.

Many Information Retrieval applications based on the document-term matrix.

Generally, a collection of m objects described by n features.

$A_{ij} =$ "intensity" of feature j in object i

Similarity of two objects - dot product.

Problem Process a large A, store "summary" so that later, for a query object (an $n-$vector) x, we can find (approximately) the similarity of x to each original object.

No free lunch - cannot hope to get it right if only 1 or 2 or very few of the original objects are mildly similar to the query.

Given an $m \times n$ matrix A, can we find an approximation A' to A (also an $m \times n$ matrix) so that

(i) A' is storable in much less space than A.

(ii) A' can be found from A in few passes of A from external memory.

(iii) $A' \approx A$.

Yes; indeed enough to store $O(1)$ randomly picked columns and $O(1)$ randomly picked rows of A, but picked according to a certain probability distribution. Also, the columns and rows can be picked in two passes through A.

– Drineas, Kannan

A any $m \times n$ matrix. C an $m \times s$ matrix of s random columns of A picked in i.i.d. trials; in each trial,

$$\text{Prob of picking column } j = \frac{\sum_i A_{ij}^2}{\sum_{k,l} A_{kl}^2}.$$

[Probabilities are proportional to length squared of the columns.]

Similarly R - $s \times n$ from s random rows of A. From C, R, we can compute a $s \times s$ matrix U so that

$$E\left(\max_{x:|x|=1} |(A - CUR)x|^2\right) \leq \frac{4}{\sqrt{s}} \sum_{ij} A_{ij}^2.$$

$$\text{Rewritten } E\left(||A - CUR||_2^2\right) \leq \frac{4}{\sqrt{s}} ||A||_F^2.$$

$$\text{Also, } E\left(||(A - CUR)||_F^2\right) \leq \frac{4}{s^{1/4}} ||A||_F^2 + \text{ Best we can do with a rank } \sqrt{s} \text{ matrix.}$$

$$\begin{pmatrix} \\ \\ & & A & & \\ \\ \\ \end{pmatrix} \approx$$

$$\begin{pmatrix} \\ \\ & C & \\ \\ \\ \end{pmatrix} \cdot \begin{pmatrix} & U & \end{pmatrix} \cdot \begin{pmatrix} & & R & & \end{pmatrix}$$

Matrix Reconstruction

m users and n products. A_{ij} measures the preference of user i for product j.

Suppose we have observed some entries of the matrix. Can we infer the other entries ? [So, having observed some market behaviour, we want to recommend to users what they would like.]

[Recommendations Systems / Collaborative filtering]

Azar, Fiat, Karlin, McSherry and Saia

Achlioptas and McSherry

Drineas, Kerenidis and Raghavan

Achlioptas and McSherry's algorithm :

p probability. Independently for each entry A_{ij} of matrix, replace it with A_{ij}/p with probability (w.p) p and 0 with probability $1-p$. So, number of non-zero entries reduced by a factor of p.

$$\hat{A}_{ij} = \begin{cases} 0 & \text{w.p.} 1-p \\ A_{ij}/p & \text{w.p.} p. \end{cases}$$

$$\begin{pmatrix} 5 & 3 & 3 & -2 & -7 & 8 & 9 \\ 1 & 2 & 2 & -17 & 1 & -8 & 9 \\ 21 & 41 & 22 & -2 & 0 & 0 & 0 \end{pmatrix} \rightarrow$$

$$\begin{pmatrix} 10 & 6 & 0 & 0 & -14 & 16 & 0 \\ 2 & 4 & 0 & 0 & 0 & -16 & 18 \\ 0 & 0 & 44 & 0 & 0 & 0 & 0 \end{pmatrix}$$

If $|A_{ij}| \leq 1$, WHP, $||A - \hat{A}||_2$ is small.