Sampling on the Fly

How does one draw a sample from streaming data for a specific end ?
Simple Example: Suppose we are given values of two functions at points $1, 2, \ldots, n$, where n is large. Say we are given two streams -

$$f(1), f(2), \ldots, f(n)$$

$$g(1), g(2), \ldots, g(n).$$

We need to find (say to RELATIVE ERROR 1 %) the mean squared distance between them:

$$\sum_{i=1}^{n} (f(i) - g(i))^2.$$

We have very little space to store things - say space is $<< n$.

Some simple attempts

To find $\sum_i (f(i) - g(i))^2$ to relative error:

- Draw a sample of values of f. Note down g at the same points. Find mean squared distance between the samples.
  - What if $f(i) = g(i)$ for all but a small fraction of the i ?
- Sample $f(i)$ with probability proportional to their (absolute) value. (Importance Sampling). What if large values of $f(i)$ are cancelled out by corresponding large values of $g(i)$?

Random Projection Theorem - A Geometric Fact

Suppose v is a vector in n dimensions (n very high) and we want to find the length of v. [Think of v as $f - g$.]
THEOREM If we pick a random set of coordinate axes and v_1, is the component of v along the first coordinate axis, then with high probability

$$|v_1| \approx\approx \frac{|v|}{\sqrt{n}}.$$

Surprise of high dimensions In 2-d, in random coordinate axes, θ (the angle of v to first axis) is equally likely to be between 0 and 90 degrees, so $|v_1|$ is well spread out between 0 and $|v|$!

Using random projections for $|f - g|$

- Choose a random vector u. [This is my first coordinate axis.]
- While $f(1), f(2), \ldots, f(n)$ stream by, find the RUNNING SUM $\sum_{i=1}^{n} f(i)u_i$. This is then the dot product of f and u, namely the component of vector f along the first axis.
- Do the same for g
- Find $(f - g) \cdot u$. Its absolute value is a good estimate of $|f - g|/\sqrt{n}$. So good that there is only small RELATIVE ERROR.
- Actually need several u to reduce variance of estimate.

Flaw and Fix

We had a random vector u stored and on reading $f(i)$, added $f(i)u_i$ to our running sum which at the end gave us $f \cdot u$.
But to do this, need to store u which is a $n-$vector and needs n space to store!!
Pseudo-random u will do instead.

- Briefly: In a pseudo-random number generator, a purely random seed of length say 100 can be used to generate a pseudo-random string of much bigger length (n).
- Store only the seed and compute each u_i as needed from the seed; add $u_i f(i)$ to running sum

High Dimensional Data

Many Examples of high dimensional data arising naturally (even when the problem has no inherent geometry):

- Patient Gene Expression data (Each patient is a point in $n-$ dimensional space, where n is the number of genes).
- Document-Term Matrix (each document is a point in n space, where $n =$ number of terms)
- In $f - g$ example above: 2 days of traffic data in a network. $f_i =$ number of messages sent by source i on day 1 and g_i on day 2. Wanted to compare the two.
- As in these examples, the data is represented in a matrix. This is not just for bookkeeping. Linear Algebra is really of use.

Dimension Reduction

Understanding and computing with high dimensional data is difficult. So want to reduce the number of dimensions. Two main techniques for dimension reduction:

- Random Projection which we just saw. Basic Property: Preserves lengths of vectors (with a known multiplicative constant $1/\sqrt{n}$). But intuitively should have a problem since it does not use the data to figure out the projection.
- Principal Component Analysis (PCA) More traditional and widely used for making the problem easier by reducing dimension as well as for "denoising".

SAMPLING IN MASSIVE MATRICES

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.

Remark 1 In general, routine statistics argument : can estimate one fixed quantity by taking a sample.

Remark 2 : 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.

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.

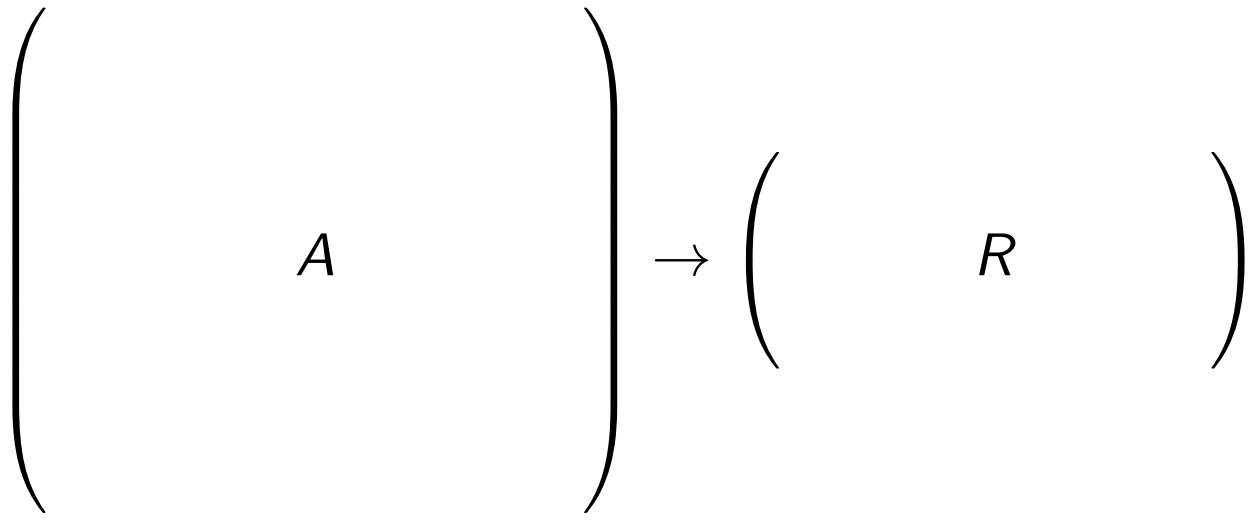
BEYOND A SINGLE PASS

In many problems : input data can be stored on disk and read a few times (costly).
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

Simple Starting Question : How does one pick a good random sample of rows of a matrix A quickly ?



Good : Want R to represent A. Many possible measures of what is good.

Basis Rows of R span row space of A (and independent)...

Modify to : Row span of R contains a vector "close" to each/most rows of A. Interpolative approximation.

Here simpler notion of good : Preserve pair-wise dot products of columns.

$$A^T A \approx R^T R.$$

Notation A is $m \times n$.
Number of rows in sample $= s$. $(s << m, n.)$

Quickly : Could mean polynomial time.

Here, Quickly : In one or two passes thro' A.

Randomization will help.

Uniform random sample won't do : All but one row zero !!

Sample with probabilities depending on size of entries in row.

The Length-squared distribution : Pick rows with probabilities proportional to their squared lengths : Make s i.i.d. trials. In each trial, pick a row $A_{(i)}$ (the i th row of A) with

$$\text{Probability of picking row } i = \frac{|A_{(i)}|^2}{||A||_F^2}.$$

If $A_{(i)}$ is picked, include a scaled version : $A_{(i)}/\sqrt{sP_i}$ as the next row of R.

If all row lengths are equal, uniform sampling will do and no scaling is necessary.

[In fact, same if all row lengths are within $O(1)$ of each other.]

Two Properties of the sampling

$$\text{Unbiased } E(R^T R) = A^T A.$$

This distribution minimizes the total variance

$$E||A^T A - R^T R||_F^2.$$

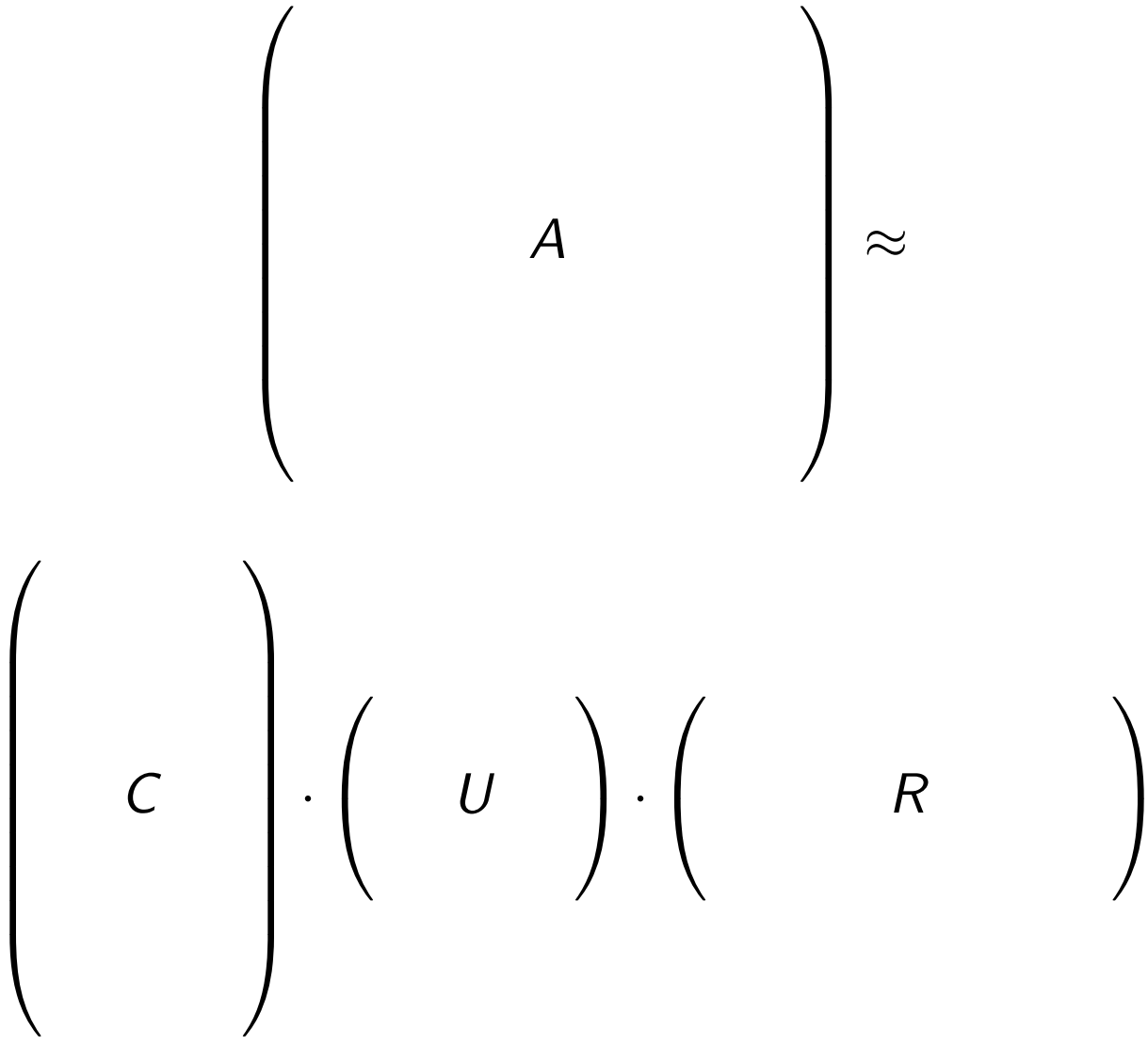
[Measuring $E||A^T A - R^T R||_F^2$ greatly simplifies the expression.]
For most results, approx length-squared distribution, where probability of picking row i is at least $\frac{c|A_{(i)}|^2}{||A||_F^2}$ suffices.
Many other properties of the distribution - fast SVD.....

$R^T R \approx A^T A$ implies the singular values of $R \approx$ the singular values of A.

More difficult questions Can one also say the same about the singular vectors of A, R ? Is there a sense in which

$$R \approx A?$$

Yes, given a sample of rows of A and a set of columns of A, both picked according to length-squared distribution, we can get an approximation to A. :



No free lunch

We cannot hope to pick from any general $m \times n$ matrix, a set of $s << m, n$ rows to form an R with $R^T R$ close to $A^T A$. Call a matrix A a PCA matrix if for $k \in O(1)$:

$$\lambda_1(A^T A) + \lambda_2(A^T A) + \ldots \lambda_k(A^T A) \geq c||A||_F^2.$$

Then, above says : $E||R^T R - A^T A||_F^2 \leq \epsilon||A^T A||_F^2$ for $s \in O(1)$.
Myriad applications of Principal Component Analysis(assume matrix is a PCA matrix or more strongly that they are numerically low-rank).
TCS contribution : Low-rank approximations to matrices and their extensions to tensors can also help solve combinatorial optimization problems.

What can be wrong with the length-squared distribution ?
Least-Squares Fit

$$\text{Min}|Ax - b|.$$

An Example : A has the first $m - 1$ columns all equal and orthogonal to b and the last column equal to b.
Repeated sampling only yields the first vector. Error $\not< O($ best error $)$!!

Tensors

Here, tensor is just a multi-dimensional array. [Matrices are 2-tensors.] No nice analog of Linear Algebra for tensors. But in many problems, we do get tensor data. One result:

- A rank-1 tensor is an outer product: entry (i, j, k) is $u_i v_j w_k$, where, u, v, w are vectors.
- Theorem Easy For any tensor A, there are $1/\epsilon^2$ rank-1 tensors whose sum B approximates A with $||A - B|| \leq \epsilon\sqrt{\text{sum of squared entries of } A}$.
- Hard We can find in polynomial time $4/\epsilon^2$ rank-1 tensors meeting the above error bound.