Implementing Randomized Matrix Algorithms in Parallel and Distributed Environments

Jiyan Yang

ICME, Stanford University

Nov 1, 2015

INFORMS 2015, Philadephia

Joint work with Xiangrui Meng (Databricks), Michael W. Mahoney (Berkeley), Jey Kottalam (Berkeley), Michael F. Ringenburg (Cray), Evan Racah (LBNL) and Prabhat (LBNL)

Roadmap

Brief Overview of Randomized Algorithms for Regressions

Spark Implementations and Empirical Results

Quick Overview of *CX* Decomposition

Applications in Bioimaging and Empirical Results

Problem formulation

- Consider the over-determined least squares problem.
- Given $A \in \mathbb{R}^{n \times d}$ and $b \in \mathbb{R}^n$ with $n \gg d$, we wish to solve

$$\min_{x \in \mathbb{R}^d} \|Ax - b\|.$$

- The memory of a single machine cannot hold the entire matrix or it takes too much time to apply a direct method.
- Currently, cases where d is a few thousand can be well handled.

An important tool: sketch

- Given a matrix $A \in \mathbb{R}^{n\times d}$, a sketch can be viewed as a compressed representation of A, denoted by ΦA.
- The matrix $\Phi \in \mathbb{R}^{r\times n}$ preserves the norm of vectors in the range of A up to small constants. That is,

$$(1-\epsilon)\|Ax\| \leq \|\Phi Ax\| \leq (1+\epsilon)\|Ax\|, \quad \forall x \in \mathbb{R}^d.$$

- $r \ll n$.

Types of sketch

- Sub-Gaussian sketch
  e.g., Gaussian transform: $\Phi A = GA$
  time: $\mathcal{O}(nd^2)$, $r = \mathcal{O}(d/\epsilon^2)$
- Sketch based on randomized orthonormal systems [Tropp, 2011]
  e.g., Subsampled randomized Hadamard transform (SRHT): $\Phi A = SDHA$
  time: $\mathcal{O}(nd \log n)$, $r = \mathcal{O}(d \log(nd) \log(d/\epsilon^2)/\epsilon^2)$
- Sketch based on sparse transform [Clarkson and Woodruff, 2013]
  e.g., count-sketch like transform (CW): $\Phi A = RDA$
  time: $\mathcal{O}(\mathtt{nnz}(A))$, $r = (d^2 + d)/\epsilon^2$
- Sampling with exact leverage scores [Drineas et al., 2006]
  Leverage scores can be viewed as a measurement of the importance of the rows in the LS fit.
  time: $\mathcal{O}(nd^2)$, $r = \mathcal{O}(d \log d/\epsilon^2)$
- Sampling with approximate leverage scores [Drineas et al., 2012]
  e.g., using CW transform to estimate the leverage scores
  time: $\mathtt{t}_{\mathtt{proj}} + \mathcal{O}(\mathtt{nnz}(A)) \log n$, $r = \mathcal{O}(d \log d/\epsilon^2)$

Summary of sketches

- There are tradeoffs between running time and sketch size r.
- In general, the sketch size r only depends on d and ϵ, independent of n!

Solvers for ℓ_2 regression

After obtaining a sketch, one can use it in one of the following two ways:

- Low-precision solvers: compute a sketch
  $+$ solve the subproblem
- High-precision solvers: compute a sketch
  $+$ preconditioning
  $+$ invoke an iterative solver

Low-precision solvers

Algorithm

1. Compute a sketch for $\bar{A} = \begin{pmatrix} A & b \end{pmatrix}$ with accuracy $\epsilon/4$, denoted by $\Phi\bar{A}$.
2. Solve for $\hat{x} = \arg\min_x \|\Phi A x - \Phi b\|$.

Low-precision solvers (cont.)

Analysis

- From

$$
\begin{aligned}
\|A\hat{x} - b\| = \|\bar{A}\left(\hat{x} \quad -1\right)\| &\le (1+\epsilon/4)\|\Phi\bar{A}\left(\hat{x} \quad -1\right)\| \\
\le \quad (1+\epsilon/4)\|\Phi\bar{A}\left(x^* \quad -1\right)\| &\le \frac{1+\epsilon/4}{1-\epsilon/4}\|\bar{A}\left(x^* \quad -1\right)\| \\
\le \quad \frac{1+\epsilon/4}{1-\epsilon/4}\|Ax^* - b\| &\le (1+\epsilon)\|Ax^* - b\|.
\end{aligned}
$$

  we can show that $\hat{x}$ is indeed a $(1+\epsilon)$-approximate solution to the original problem.

- Error bound for error vector $\|\hat{x} - x^*\|$ can also be derived.

- The total running time is

$$t(\texttt{sketch}) + t(\texttt{solving the subproblem}).$$

  The latter is $\mathcal{O}(rd^2)$. The tradeoffs among sketches are manifested here.

High-precision solvers

Algorithm

1. Compute a sketch ΦA.
2. Compute the economy QR factorization $\Phi A = QR$.
3. Invoke an iterative solver such as LSQR with R^{-1} as a right-preconditioner.

Analysis

- Theoretical results ensure the quality of the preconditioner, e.g., using Gaussian transform with sketch size $2d$, $\kappa(AR^{-1}) \leq 6$ with high probability.
- Normally, the convergence rate of the iterative solver depends on $\text{cond}(A)^2$.
- Given target accuracy ϵ, we expect the algorithm to converge to a solution within a constant number of iterations.

Distributed setting

- We assume that dataset is partitioned along the high dimension and stored in a distributed fashion.
- Unlike traditional computing, in the distributed setting we want to minimize the communication cost as much as possible.

The costs of computing in distributed settings

- floating point operations
- bandwidth costs: $\propto$ total bits transferred
- latency costs: $\propto$ rounds of communication

$$\textbf{FLOPS}^{-1} \ll \textbf{bandwidth}^{-1} \ll \textbf{latency}.$$

Basically, we want to make as few passes over the dataset as possible.

Spark

“Apache Spark is a fast and general engine for large-scale data processing.”

— `http://spark.apache.org`

Solvers for ℓ_2 regression

- Low-precision solvers: compute a sketch (MapReduce, 1-pass)
  + solve the subproblem (local SVD)
- High-precision solvers: compute a sketch (MapReduce, 1-pass)
  + preconditioning (local QR)
  + invoke an iterative solver (Involves only matrix-vector products, which can be well handled by Spark. # passes is proportional to # iterations)

Notes

- Methods for computing sketches are embarrassingly parallel and can be implemented under the MapReduce framework.
- Since the sketch is small, operations like SVD or QR can be performed exactly locally.
- Preconditioning is crucial because in distributed computing where communication cost is expensive, we want to reduce the number of iterations in the iterative solver.

Experimental setup

Sketches

- `PROJ CW` — Random projection with the input-sparsity time CW method (sparse)
- `PROJ GAUSSIAN` — Random projection with Gaussian transform (dense)
- `PROJ RADEMACHER` — Random projection with Rademacher transform (dense)
- `PROJ SRDHT` — Random projection with Subsampled randomized discrete Hartley transform (dense, no longer fast)
- `SAMP APPR` — Random sampling based on approximate leverage scores (fast approximate leverage scores)
- `SAMP UNIF` — Random sampling with uniform distribution (for completeness)

Experimental setup (cont.)

Datasets

- We used synthetic datasets with *uniform* or *nonuniform* leverage scores, and *low* or *high* condition number.
- Recall that leverage scores can be viewed as a measurement of the importance of the rows in the LS fit.
- These properties of the matrix have a strong influence on the solution quality.

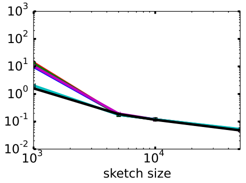
Resources

The experiments were performed on an Amazon EC2 cluster with 16 nodes (1 master and 15 slaves), each of which has 4 CPU cores at clock rate 2.5GHz with 25GB RAM.

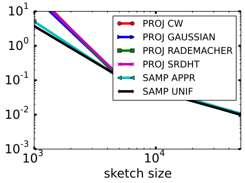
Results

More details can be found in [Implementing Randomized Matrix Algorithms in Parallel and Distributed Environments. Y, Meng and Mahoney, 2015].

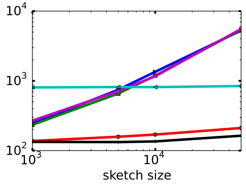
Low-precision solvers: effect of sketch size



(a) $\|x - x^*\|_2 / \|x^*\|_2$

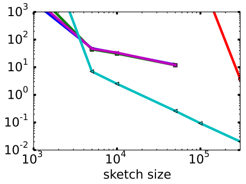


(b) $|f - f^*|/f^*$

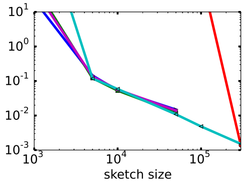


(c) Running time(sec)

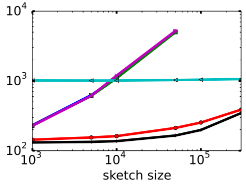
$1e7 \times 1000$ **matrix with** *uniform* **leverage scores**



(d) $\|x - x^*\|_2 / \|x^*\|_2$



(e) $|f - f^*|/f^*$

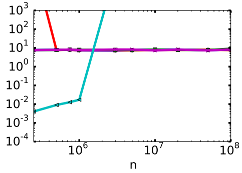


(f) Running time(sec)

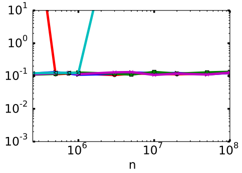
$1e7 \times 1000$ **matrix with** *nonuniform* **leverage scores**

Figure: Evaluation of all 6 algorithms on matrices with uniform and nonuniform leverage scores.

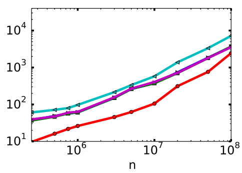
Low-precision solvers: effect of n



(a) $\|x - x^*\|_2 / \|x^*\|_2$

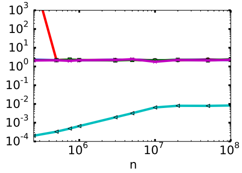


(b) $|f - f^*|/f^*$

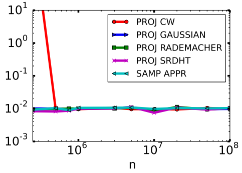


(c) Running time(sec)

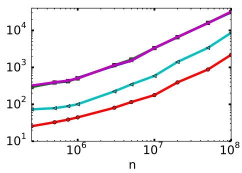
Small sketch size



(d) $\|x - x^*\|_2 / \|x^*\|_2$



(e) $|f - f^*|/f^*$



(f) Running time(sec)

Large sketch size

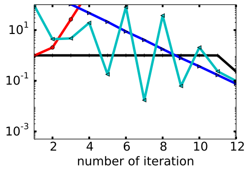
Figure: Performance of all algorithms on matrices with *nonuniform* leverage scores, *high* condition number, varying n from 2.5e5 to 1e8 with fixed $d = 1000$.

High-precision solvers: preconditioning quality

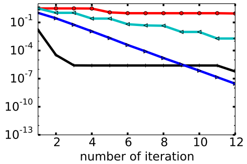
r	PROJ CW	PROJ GAUSSIAN	SAMP APPR
5E2	1.08E8	2.17E3	1.21E2
1E3	1.1E6	5.74	75.03
5E3	5.5E5	1.91	25.87
1E4	5.1E5	1.57	17.07
5E4	1.8E5	1.22	6.91
1E5	1.14	1.15	4.76

Table: Quality of preconditioning, i.e., $\kappa(AR^{-1})$, on a matrix of size $1e6$ by 500 and condition number $1e6$ using several kinds of sketch.

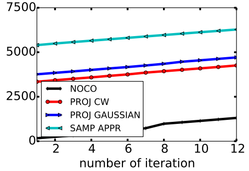
High-precision solvers: on badly conditioned matrix



(a) $\|x - x^*\|_2 / \|x^*\|_2$

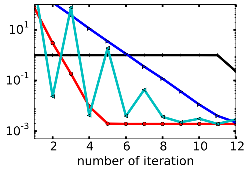


(b) $|f - f^*| / f^*$

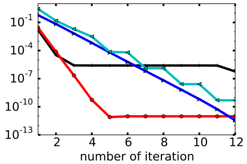


(c) Running time(sec)

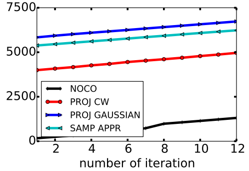
sketch size $s = 5e3$



(d) $\|x - x^*\|_2 / \|x^*\|_2$



(e) $|f - f^*| / f^*$



(f) Running time(sec)

sketch size $s = 5e4$

Figure: LSQR with randomized preconditioner on a matrix of size $1e8$ by 1000 and condition number $1e6$.

CX decomposition

Given an $n \times d$ matrix A, the CX decomposition decomposes A into two matrices C and X, where C is an $n \times c$ matrix that consists of c actual columns of A, and X is a $c \times d$ matrix such that

$$A \approx CX.$$

A quantitative measurement of the closeness between CX and A is obtained by using the matrix Frobenius norm of the difference: if the residual error

$$\|A - CX\|_F$$

is smaller, then CX provides a better quality approximation to A.

Approximate low-rank leverage scores

Let $A = U\Sigma V^T$ be its SVD. Given a low-rank parameter k, let U_k be the matrix consisting of the top-k singular vectors. The leverage scores associated with rank k are the square row norms of U_k,

$$\ell_i = \|U_{i,1:k}\|^2 \qquad i = 1, \ldots, n.$$

- Randomized matrix algorithms can be used to approximate ℓ_i's.
- First, use techniques similar to randomized SVD to approximate U_k, denoted by $\tilde{U}_k$. [Finding Structure with Randomness: Probabilistic Algorithms for Constructing Approximate matrix Decompositions. Halko, Martinsson and Tropp, 2010].
- Second, compute or estimate the row norms of $\tilde{U}_k$ as our estimation of leverage scores associated with k.

CX decomposition with approximate leverage scores

Algorithm

1. Given A and a rank parameter k, approximate the leverage scores associated with rank k.
2. The matrix C can be constructed by sampling c columns from A based on the approximate leverage scores associated with k.
3. Construct X based on C.

- Theoretical results indicate if we sample $c = \mathcal{O}(k \log k/\epsilon^2)$ columns, then $\|A - CX\|_F \leq (1+\epsilon)\|A - A_k\|_F$.
- Here c can be higher than k. We can restrict X to be a rank-k matrix so that CX is a rank-k approximation to A.
- Unlike PCA, we are getting actual columns/rows of A that can be used to reconstruct A well.

Application to OpenMSI image analysis

- Here we show the results on a real dataset — mass spectrometry imaging dataset of a complex biological sample, which is the largest (1TB) mass spectrometry imaging dataset in the field.
- Each column corresponds to an ion and each row to a pixel. Elements are intensities.
- We would like to apply CX decomposition on A. The selected columns can be viewed as important ions.
- Because A is very large, we invoke randomized algorithms to obtain approximate leverage scores.
- We implement the algorithm using Spark on three contemporary platforms. For all platforms, we sized the Spark job to use 960 executor cores.
- More results can be found in [A Multi-platform Evaluation of Low-rank Matrix Factorizations in Spark. Gittens, Kottalam, Y, et al.].

Platforms

Platform	Total Cores	Core Frequency	DRAM	SSDs
Amazon EC2 `r3.8xlarge`	960 (32 per-node)	2.5 GHz	244 GiB	2 x 320 GB
Cray XC40	960 (32 per-node)	2.3 GHz	252 GiB	None
Experimental Cray cluster	960 (24 per-node)	2.5 GHz	126 GiB	1 x 800 GB

Table: Specifications of the three hardware platforms used in these performance experiments.

Running time

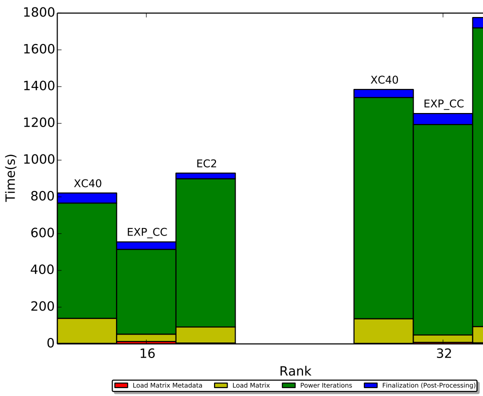


Figure: Run times for the various stages of computation of CX on the three platforms using $k = 16$ and $k = 32$ on the 1 TB dataset, using the default partitioning on each platform.

Analysis of the running time

- First, both the EC2 nodes and the experimental Cray cluster nodes have fast SSD storage local to the compute nodes that they can use to store Spark's shuffle data. The Cray® XC40™ system's nodes, on the other hand, have no local persistent storage devices. Thus we must emulate local storage with a remote Lustre filesystem.
- Second, the Cray XC40 and the experimental Cray cluster both communicate over the HPC-optimized Cray Aries interconnect, while the EC2 nodes use 10 Gigabit Ethernet.

Platform	**Total Runtime**	**Load Time**	**Time Per Iteration**	**Average Local Task**	**Average Aggregation Task**	**Average Network Wait**
Amazon EC2 `r3.8xlarge`	24.0 min	1.53 min	2.69 min	4.4 sec	27.1 sec	21.7 sec
Cray XC40	23.1 min	2.32 min	2.09 min	3.5 sec	6.8 sec	1.1 sec
Experimental Cray cluster	15.2 min	0.88 min	1.54 min	2.8 sec	9.9 sec	2.7 sec

Table: The average amount of time spent waiting for a network fetch, to illustrate the impact of the interconnect.

Strong scaling

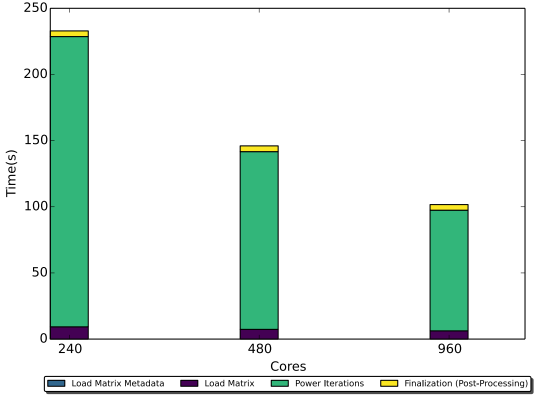


Figure: Strong scaling for the 4 phases of CX on an XC40 for 100GB dataset at $k = 32$ and default partitioning as concurrency is increased.

Distribution of approximate leverage scores

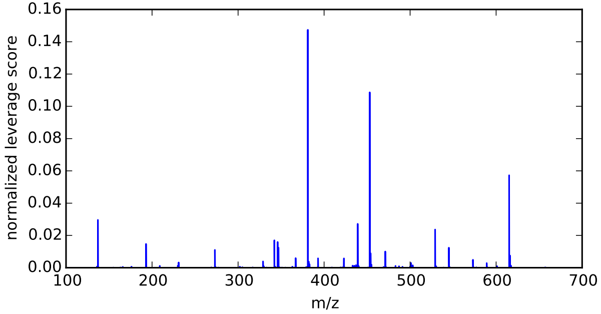


Figure: Normalized leverage scores (sampling probabilities) for m/z marginalized over τ. Three narrow regions of m/z account for 59.3% of the total probability mass.

Plots of important pixels

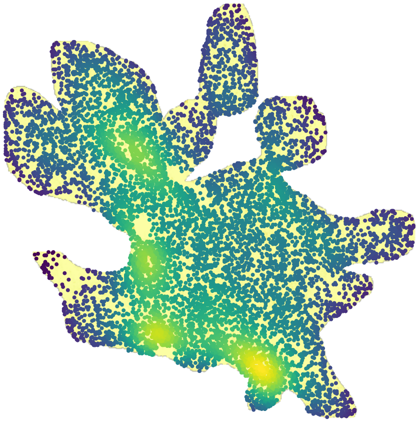


Figure: Plot of 10000 points sampled by leverage score. Color and luminance of each point indicates density of points at that location as determined by a Gaussian kernel density estimate.

Conclusion

- Sketching is a useful and important tool in randomized matrix algorithms. It can be used in two ways to compute an approximate solution for least-squares problems depending on the desired accuracy.
- These randomized algorithms are amenable to parallel and distributed environments using Spark.
- Empirical results verify theoretical properties of the algorithms and demonstrate that over-determined least squares problems can be solved to low, medium or high precision in existing distributed systems on up to terabyte-sized data.
- Randomized linear algebra can also be applied to low-rank subspace approximation. CX decomposition with approximate leverage scores is amenable to distributed computing.
- Its application in bioimaging produces promising scientific results.