Randomized Approximation of Linear Least Squares Regression at Sub-linear Cost

Victor Y. Pan[1,2],[a] and Qi Luan[2],[b],

[1] Department of Computer Science
Lehman College of the City University of New York
Bronx, NY 10468 USA
[2] Ph.D. Programs in Computer Science and Mathematics
The Graduate Center of the City University of New York
New York, NY 10036 USA
[a] victor.pan@lehman.cuny.edu
http://comet.lehman.cuny.edu/vpan/
[b] qi_luan@yahoo.com

Abstract

We prove that with a high probability (*whp*) nearly optimal solution of the highly important problem of Linear Least Squares Regression (LLSR) can be computed at *sub-linear cost* for a random input. Our extensive tests are in good accordance with this result.

Key Words: Least Squares Regression, Sub-linear cost, Gaussian random matrices

2000 Math. Subject Classification: 65Y20, 65F05, 68Q25, 68W20, 68W25

1 Introduction

The LLSR problem. LLSR is a hot research subject, fundamental for Matrix Computations and Big Data Mining and Analysis. The matrices that define Big Data are frequently so immense that realistically one can access and process only a tiny fraction of their entries and thus must perform computations at *sub-linear cost* – by using much fewer arithmetic operations and memory cells than the input matrix has entries.

Our progress. Although all LLSR algorithms running at sub-linear cost fail on the worst case inputs, we prove that sub-linear cost extension of Sarlòs algorithm of [S06] approximates an optimal solution of the problem arbitrarily closely with a high probability (whp) in the case of a random Gaussian input matrix, filled with independent identically distributed Gaussian (normal) random variables. Hereafter we call such a matrix just *Gaussian* and call the LLSR problem for random input *dual*.

Our numerical tests are in a good accordance with this theorem, thus suggesting that the LLSR problem can be solved at sub-linear cost for a large class of inputs.

Related works. Our transition to dual matrix computations in this paper extends the earlier work in [PQY15] and [PZ16] and is extended in [PLSZa], [PLSZb], [PLa], and [LPSa] to the solution at sub-linear cost of the dual problem of Low Rank Approximation of a matrix.

Organization of the paper. In the next section we recall the LLSR problem and its randomized approximate solution by Sarlos' of [S06]. We cover its variation running at sub-linear cost in Section 3. In Section 4, the contribution of the second author, we cover numerical tests.

2 Linear Least Squares Regression

Problem 2.1. [Least Squares Solution of an Overdetermined Linear System of Equations or Linear Least Squares Regression (LLSR).] *Given two integers m and d such that $1 \le d < m$, a matrix $A \in \mathbb{R}^{m\times d}$, and a vector $\mathbf{b} \in \mathbb{R}^m$, compute and output a vector $\mathbf{x} \in \mathbb{R}^d$ that minimizes the spectral norm $||A\mathbf{x} - \mathbf{b}||$ or equivalently outputs the subvector $\mathbf{x} = (y_j)_{j=0}^{d-1}$ of the vector*

$$\mathbf{y} = (y_j)_{j=0}^{d} = \text{argmin}_{\mathbf{v}}\ ||M\mathbf{v}|| \text{ such that } M = (A \mid \mathbf{b}) \text{ and } \mathbf{v} = \begin{pmatrix} \mathbf{x} \\ -1 \end{pmatrix}. \tag{2.1}$$

The minimum norm solution to this problem is given by the vector $\mathbf{x} = A^+\mathbf{b}$ for A^+ denoting the Moore–Penrose pseudo inverse of A; $A^+\mathbf{b} = (A^TA)^{-1}A^T\mathbf{b}$ if a matrix A has full rank d.

Algorithm 2.1. [Randomized Approximate LLSR from [S06].

INPUT: *An $m \times (d+1)$ matrix M.*

OUTPUT: *A vector $\mathbf{x} \in \mathbb{R}^d$ approximating a solution of Problem 2.1.*

INITIALIZATION: *Fix an integer s such that $d \le s \ll m$.*

COMPUTATIONS: *1. Generate a matrix $F \in \mathbb{R}^{s\times m}$.*

2. Compute and output a solution $\mathbf{x}$ of Problem 2.1 for the $s \times (d+1)$ matrix FM.

The following theorem shows that the algorithm outputs approximate solution to Problem 2.1 for M whp if $\sqrt{s}\ F$ is the linear space $\mathcal{G}^{s\times m}$ of $s \times m$ Gaussian matrices.[1]

Theorem 2.1. (Error Bound for Algorithm 2.1. See [W14, Theorem 2.3].) *Let us be given two integers s and d such that $0 < d \le s$, two matrices $M \in \mathbb{R}^{m\times(d+1)}$ and $F \in \mathcal{G}^{s\times m}$, and two tolerance values γ and ϵ such that*

$$0 < \gamma < 1,\ 0 < \epsilon < 1,\ \text{and } s = ((d + \log(1/\gamma)\ \epsilon^{-2})\ \eta \tag{2.2}$$

for a constant η. Then

$$\text{Probability}\Big\{1 - \epsilon \le \frac{1}{\sqrt{s}}\ \frac{||FM\mathbf{y}||}{||M\mathbf{y}||} \le 1 + \epsilon \text{ for all vectors } \mathbf{y} \ne \mathbf{0}\Big\} \ge 1 - \gamma. \tag{2.3}$$

For $m \gg s$ the transition from M to the matrix FM substantially decreases the size of Problem 2.1; the computation of the matrix FM, however, involves order of $dsm \ge d^2m$ flops, and this dominates the overall arithmetic computational cost of the solution.

The current record upper estimate for this cost is $O(d^2m)$ (see [CW17], [W14, Section 2.1]), while the record lower bound of [CW09] has order $(s/\epsilon)(m+d)\log(md)$ provided that the relative output error norm is within a factor of $1+\epsilon$ from its minimal value.

[1] Such approximate solution serve as pre-processors for practical implementation of numerical linear algebra algorithms for Problem 2.1 of least squares computation [M11, Section 4.5], [RT08], [AMT10].

3 Dual LLSR at Sub-linear Cost

If an LLSR algorithm runs at sub-linear cost, then it does not access an entry $m_{i,j}$ for some pair i and j and so cannot minimize the norm $|M\mathbf{y}|$. Indeed we can decrease it by modifying the input entry $m_{i,j}$, and this would not change the output of the algorithm. Therefore no algorithm can solve the LLSR problem at sub-linear cost for the worst case input M, but next we solve whp its *dual variant* where we assume that the input matrix M is scaled Gaussian and allow any orthogonal multiplier F, including sparse ones with which the algorithm runs at sub-linear cost.

Theorem 3.1. [Error Bounds for Dual LLSR.] *Suppose that we are given three integers s, m, and d such that $0 < d < s < m$, and two tolerance values γ and ϵ satisfying (2.2). Define an orthogonal matrix $Q_{s,m} \in \mathbb{R}^{s\times m}$ and a matrix $G_{m,d+1} \in \mathcal{G}^{m\times(d+1)}$ and write*

$$F := a\ Q_{s,m} \text{ and } M := b\ G_{m,d+1} \tag{3.1}$$

for two scalars a and b such that $ab\sqrt{s} = 1$. Then

$$1-\epsilon\ \le \text{Probability}\Big\{\frac{||FM\mathbf{z}||}{||M\mathbf{z}||} \le 1+\epsilon \text{ for all vectors } \mathbf{z} \ne \mathbf{0}\Big\} \ge 1-\gamma.$$

Proof. Observe that the theorem is equivalent to Theorem 2.1 because the $s\times(d+1)$ matrix $\frac{1}{ab}FM$ is Gaussian by virtue of orthogonality invariance of Gaussian matrices. □

The theorem shows that for any orthogonal matrix F of (3.1) the transition $M \to FM$ changes the error of LLSR within a factor of about $1+\epsilon$ except for a narrow class of matrices M.

We can increase chances for avoiding this class by trying to solve the LSSR problem repeatedly for the same multiplier F and a sequence of input matrices M_i or equivalently, for the same matrix M and various multiplier F_i such that $FM_i = F_iM$ for all $i = 1, 2, \dots, u$. The latter way should be preferred because it runs at sub-linear cost in the case of sparse multipliers $F_1, \dots, F_u$ and a sufficiently small integer u.

4 Numerical Tests for LLSR

In this section we present the results of our tests of Algorithm 2.1 for LLSR on both synthetic and real-world data. We worked with random orthogonal multipliers, let $\bar{x} := \arg\min_{\mathbf{x}} ||FA\mathbf{x} - F\mathbf{b}||$, and computed the relative residual norms

$$\frac{||A\bar{\mathbf{x}} - \mathbf{b}||}{\min_{\mathbf{x}} ||A\mathbf{x} - \mathbf{b}||}.$$

In our tests these ratios quite closely approximated one from above.

We used the following random multipliers $F \in \mathbb{R}^{k\times m}$:

(i) submatrices of $m \times m$ random permutation matrices,

(ii) products of random matrices of Givens Rotation, and

(iii) ASPH matrices from [PLSZa] and [PLSZb], which are output after 4 recursive steps out of k steps of generation of subsampled matrices of Hadamard thansform.

(iv) For comparison we also included the test results with Gaussian multipliers.

We generated a multiplier of class (ii) as the matrix $F = \prod_{t=1}^{h} G(i_t, j_t, \frac{\pi}{4})$, where $G(i, j, \phi)$ denotes the matrix of Givens rotation with the 2×2 Givens block in the ith row and jth column, ϕ is the angle of rotation (cf. [GL13, Section 5.1.8]), i_t and j_t are two indices chosen uniformly

from 1 to m and independently over t, and $h = o(m)$. Such a product F is very sparse and can be multiplied by matrix $M = (A \mid \mathbf{b})$ at sub-linear cost.

We performed our tests on a machine with Intel Core i7 processor running Windows 7 64bit; we invoked the *lstsq* function from Numpy 1.14.3 for solving the LLSR problems.

4.1 Synthetic Input Matrices

For synthetic inputs, we generated input matrices $A \in \mathbb{R}^{m \times n}$ by following (with a few modifications) the recipes of extensive tests in [AMT10], which compared the running time of the regular LLSR problems and the reduced ones with WHT, DCT, and DHT pre-processing.

We used input matrices A of size 4096×200 and 16834×500, and of the following types: Gaussian matrices, ill-conditioned random matrices, and semi-coherent matrices. We generated the input vectors $\mathbf{b} = \frac{1}{||A\mathbf{w}||}A\mathbf{w} + \frac{0.001}{||\mathbf{v}||}\mathbf{v}$, where $\mathbf{w}$ and $\mathbf{v}$ were random Gaussian vectors of size d and m, respectively, and so $\mathbf{b}$ is in the range of A up to a smaller term $\frac{0.001}{||\mathbf{v}||}\mathbf{v}$.

Table 4.1 displays the test results for Gaussian input matrices.

Table 4.2 displays the results for ill-conditioned random inputs generated through SVD $A = UV^*$ where the orthogonal matrices U and V of singular vectors were given by the Q factors in QR-factorization of independent Gaussian matrices and where $\Sigma = \mathrm{diag}(\sigma_j)_j$ with $\sigma_j = 10^{5-j}$ for $j = 1, 2 \ldots, 14$ and $\sigma_j = 10^{-10}$ for $j > 14$.

Table 4.3 displays the test results for the input matrices

$$A_{m \times d} = \begin{bmatrix} G_{(m-d/2) \times d/2} & \\ & D_{d/2} \end{bmatrix}$$

where $G_{(m-d/2) \times d/2}$ is a random Gaussian matrix and $D_{d/2}$ is a diagonal matrix with diagonal entries chosen independently uniformly from ± 1. We call these matrices $A_{m \times d}$ *semi-coherent*.

Remark 4.1. *The* coherence *of a matrix $A_{m \times n}$ with SVD $U\Sigma V^*$ is defined as the maximum squared row norm of its left singular matrix U, with 1 being its maximum and n/m being its minimum. If the test input has coherence 1, then in order to have an accurate result the multiplier must "sample" the corresponding rows with maximum row norm in the left singular matrix. The semi-coherent inputs have coherence* 1 *and are the harder cases.*

Our input matrices A are highly over-determined, having many more rows than columns. We have chosen $k = rd$, $r = 2, 4, 6$ for the multipliers F. By decreasing the ratio $r = k/d$ we would accelerate the solution, but we had to keep it large enough in order to yield accurate solution.

We performed 100 tests for every triple of the input class, multiplier class, and test sizes, and computed the mean of the relative residual norm.

The test results displayed in Tables 4.1–4.3 show that our multipliers were consistently effective for random matrices. The performance was not affected by the conditioning of input matrices.

Size of A	Multiplier	k = 2n	k = 4n	k = 6n
4096×200	Gaussian	1.4132	1.1553	1.0956
	ASPH	1.3984	1.1308	1.0725
	Product of GRs	1.3972	1.1342	1.0713
	SubPerm	1.3973	1.1332	1.0706
16384×500	Gaussian	1.4070	1.1556	1.0958
	ASPH	1.4056	1.1412	1.0817
	Product of GRs	1.4043	1.1420	1.0814
	SubPerm	1.4041	1.1394	1.08281

Table 4.1: Relative error norm in tests with Gaussian inputs

Size of A	Multiplier	k = 2n	k = 4n	k = 6n
4096×200	Gaussian	1.0186	1.0089	1.0055
	ASPH	1.0172	1.0065	1.0040
	Product of GRs	1.0167	1.0067	1.0039
	SubPermu	1.0155	1.0070	1.0039
16384×500	Gaussian	1.0072	1.0032	1.002
	ASPH	1.0056	1.0029	1.0016
	Product of GRs	1.0066	1.0027	1.0017
	SubPermutation	1.0060	1.0029	1.0018

Table 4.2: Relative error norm in tests with ill-conditioned random inputs

Size of A	Multiplier	k = 2n	k = 4n	k = 6n
4096×200	Gaussian	1.4148	1.1519	1.0976
	ASPH	6.0446	2.5770	1.2943
	Product of GRs	10.4066	8.6045	6.9614
	SubPerm	13.1626	12.2729	11.6731
16384×500	Gaussian	1.4179	1.1545	1.0946
	ASPH	5.1576	3.1368	1.9406
	Product of GRs	7.0490	6.0421	5.2992
	SubPerm	8.1964	7.9180	7.6295

Table 4.3: Relative error norm in tests with semi-coherent inputs

Multiplier	$k = 2n$	$k = 4n$	$k = 6n$
Gaussian	1.437	1.155	1.090
ASPH	1.430	1.157	1.090
Product of GRs	1.555	1.204	1.124
SubPerm	2.190	1.324	1.170

Table 4.4: Relative residual norms in tests with Red Wine Quality Data

Multiplier	$k = 2n$	$k = 4n$	$k = 6n$	$k = 8n$	$k = 10n$
Gaussian	1.4196	1.1569	1.0944	1.0735	1.0495
ASPH	1.4760	1.1822	1.1055	1.0691	1.0541
Product of GRs	1.5883	1.1890	1.1186	1.0892	1.0686
SubPerm	1.6738	1.3418	1.1698	1.1237	1.1039

Table 4.5: Relative residual norms in tests with California Housing Prices Data

4.2 Red Wine Quality Data and California Housing Prices Data

In this subsection we present the test results for real-world inputs, namely the Red Wine Quality Data and California Housing Prices Data. For each triple of the dataset, multiplier type and multiplier size, we repeated the test 100 times and computed the mean relative residual norm. The results for these two datasets are displayed in Tables 4.4 and 4.5.

The Red Wine Quality Data includes 11 physiochemical feature data (input variables), such as *fixed acidity*, *residual sugar level*, and *pH level*, and one sensory data *wine quality* (output variable) for 1599 different variants of the Portuguese "Vinho Verde" wine. See further information in [CCAMR09]. This dataset is often applied in regression tests that use physiochemical data of a specific wine in order to predict its quality, and among various types of regression LLSR is regarded as a popular choice.

From this dataset we constructed 2048×12 input matrix A with each row representing one variant of red wine, and with columns consisting of a bias column and eleven physiochemical feature columns. The input vector $\mathbf{b}$ is a vector consisting of the wine quality level (between 0 and 10) for each variant. For simplicity, besides the 1599 rows of the original data, we padded the rest of rows with zeros and performed a full row permutation of A.

The California Housing Prices data appeared in [PB97] and were collected from the 1990 California Census, including 9 attributes for each of the 20,640 Block Groups observed. This dataset is used for regression tests in order to predict the *median housing value* of a certain area given collected information of this area, such as *population*, *median income*, and *housing median age*.

We randomly selected 16,384 observations from the dataset in order to construct independent input matrix A_0 of size 16384×8 and dependent input vector $\mathbf{b} \in \mathbb{R}^{16384}$. Furthermore, we augmented A_0 by a single bias column, i.e. $A = \begin{bmatrix} A_0 & \mathbf{1} \end{bmatrix}$.

The result presented in Table 4.4 and 4.5 shows that the approximate solution obtained by applying our multipliers is almost as accurate as the optimal solution. Moreover, the reference multipliers (Gaussian) only provide marginal improvement in terms of relative residual norm comparing to proposed sparse multipliers.

Acknowledgements: Our work was supported by NSF Grants CCF–1563942 and CCF–1733834 and PSC CUNY Award 69813 00 48.

References

[AMT10] H. Avron, P. Maymounkov, S. Toledo, Blendenpik: Supercharging LAPACK's Least-squares Solver, *SIAM Journal on Scientific Computing*, **32**, 1217–1236, 2010.

[CCAMR09] P. Cortez, A. Cerdeira, F. Almeida, T. Matos and J. Reis, Modeling wine preferences by data mining from physicochemical properties, In Decision Support Systems, Elsevier, **47 4**, 547–553, 2009.

[CW09] K. L. Clarkson, D. P. Woodruff, Numerical linear algebra in the streaming model, in *Proc. 41st ACM Symposium on Theory of Computing (STOC 2009)*, pages 205–214, ACM Press, New York, 2009.

[CW17] K. L. Clarkson, D. P. Woodruff, Low Rank Approximation and Regression in Input Sparsity Time, *Journal of the ACM*, **63, 6**, Article 54, 2017; doi: 10.1145/3019134.

[GL13] G. H. Golub, C. F. Van Loan, *Matrix Computations*, The Johns Hopkins University Press, Baltimore, Maryland, 2013 (fourth edition).

[LPSa] Q. Luan, V. Y. Pan, J. Svadlenka, Low Rank Approximation Directed by Leverage Scores and Computed at Sub-linear Cost, preprint, 2019.

[M11] M. W. Mahoney, Randomized Algorithms for Matrices and Data, *Foundations and Trends in Machine Learning*, NOW Publishers, **3, 2**, 2011.

[PB97] R.K. Pace, R. Barry, Sparse Spatial Autoregressions, Statistics and Probability Letters, 33, 291-297, 1997.

[PLa] V. Y. Pan, Q. Luan, Refinement of Low Rank Approximation of a Matrix at Sub-linear Cost, preprint, 2019.

[PLSZa] V. Y. Pan, Q. Luan, J. Svadlenka, L. Zhao, Low Rank Approximation by Means of Subspace Sampling at Sub-linear Cost, preprint, 2019.

[PLSZb] V. Y. Pan, Q. Luan, J. Svadlenka, L. Zhao, CUR Low Rank Approximation at Sub-linear Cost, preprint, 2019.

[PQY15] V. Y. Pan, G. Qian, X. Yan, Random Multipliers Numerically Stabilize Gaussian and Block Gaussian Elimination: Proofs and an Extension to Low-rank Approximation, *Linear Algebra and Its Applications*, **481**, 202–234, 2015.

[PZ16] V. Y. Pan, L. Zhao, Low-rank Approximation of a Matrix: Novel Insights, New Progress, and Extensions, *Lecture Notes in Computer Science (LNCS)*, **9691**, 352–366, Springer International Publishing, Switzerland (2016).

[S06] T. Sarlós, Improved Approximation Algorithms for Large Matrices via Random Projections, *Proceedings of IEEE Symposium on Foundations of Computer Science (FOCS)*, 143–152, 2006.

[RT08] V. Rokhlin, M. Tygert, A Fast Randomized Algorithm for Overdetermined Linear Least-squares Regression, *Proc. Natl. Acad. Sci. USA*, **105, 36**, 13212–13217, 2008.

[W14] D.P. Woodruff, Sketching As a Tool for Numerical Linear Algebra, *Foundations and Trends in Theoretical Computer Science*, **10, 1–2**, 1–157, 2014.