Surrogate-based Autotuning for Randomized Numerical Linear Algebra

Younghyun Cho, James W. Demmel, Michał Dereziński, Haoyun Li, Hengrui Luo, Michael W. Mahoney, Riley J. Murray

Abstract

The field of Randomized Numerical Linear Algebra (RandNLA) has made significant developments and shown high quality empirical performance in some scenarios (e.g., overdetermined least-squares solvers). However, the practical performance of a RandNLA method usually hinges on the careful selection of multiple algorithm-specific tuning parameters. In addition, such a parameter selection would affect both the runtime of the algorithm and the accuracy of the result, which makes the parameter selection even harder. This motivates us to develop an automated process that helps find the (near-)optimal parameters for practical performance, with a focus on the applications relevant to RandNLA practitioners.

This extended abstract, which is based on our ongoing work [1], presents a surrogate-based autotuning approach for tuning RandNLA algorithms. We present a tuning pipeline that is built based on Bayesian optimization (BO) with Gaussian Process (GP) regression, which is an empirical approach where we aim to find the optimal parameter selection for a given tuning budget. At a high level, our pipeline follows the typical BO procedure, where we evaluate several parameter configurations, (iteratively) build a surrogate performance model based on the obtained evaluation results, and then find the next sample to evaluate until we reach the given tuning budget, along with an objective function to minimize the runtime of the algorithm while providing a satisfactory accuracy. Furthermore, we also apply a transfer learning approach to further reduce the tuning cost, especially when there are previously collected evaluation data from other similar but different tasks (e.g., the same algorithm but solving with different input data matrices). This makes the tuning approach more cost efficient and practical for RandNLA practitioners. The tuning pipeline uses GPTune [11] as the BO framework. GPTune is an open-source autotuner that was originally designed for tuning large-scale high-performance computing codes but is also general and can support tuning other domains of codes.

In particular, we show the efficacy of our tuning pipeline, in the context of sketch-and-precondition (SAP) based randomized least squares methods in solving large-scale overdetermined problems, minimizing $\|\boldsymbol{A}\boldsymbol{x}-\boldsymbol{b}\|_2^2$, where $\boldsymbol{A}$ is with the size of m by n with $m \gg n$, as SAP-based randomized least squares solvers that have been one of the successful applications in RandNLA. The SAP least squares approach can be summarized into following five steps: (1) Construct a sketching matrix $\boldsymbol{S}$ (with size of d by m; multiple schemes exist such as Sparse Johnson–Lindenstrauss Transform (SJLT) [5] and LessUniform [6, 7] to form a sketching matrix) to approximate the input data matrix $\boldsymbol{A}$, (2) Compute $\hat{\boldsymbol{A}} = \boldsymbol{S}\boldsymbol{A}$, (3) Generate a preconditioner matrix $\boldsymbol{M}$ from $\hat{\boldsymbol{A}}$ (e.g., using QR or SVD), (4) Use an iterative method for the preconditioned least squares for minimizing $\|\boldsymbol{A}\boldsymbol{M}\boldsymbol{z} = \boldsymbol{b}\|_2^2$ (e.g., using preconditioned LSQR or preconditioned gradient descent (PGD)), and finally (5) Compute the result vector, $\boldsymbol{M}\boldsymbol{z}$.

We observe that the SAP-based least squares solver has multiple types of parameters to be tuned. The possible tuning parameters include some categorical variables to choose what the sparse sketching operator and the iterative solver for the preconditioned least squares to be used, as well as continuous/integer parameters to configure the size of the sketching matrix (d of $\boldsymbol{S}$) as well as the sparsity of the sketching matrix (i.e., number of nonzero elements per row or column of $\boldsymbol{S}$). In our experiments, we search this categorical space, using several implementations that are motivated by

the well-known works such as Blendenpik [2], LSRN [3], and NewtonSketch [4]. Then, we search a certain range of continuous/integer parameters to configure the sketching matrix, in terms of the size of the sketching matrix and the sparsity of the sketching matrix. In addition, the iterative solvers such as LSQR [8] and PGD finish their iterations based on the termination criteria with a desired level of accuracy (which we call "safety factor"). We regard that as a tuning parameter, and our tuning pipeline computes a relative residual error by comparing the results of the SAP least squares solver and the result obtained from a traditional direct solver. The relative error is used as the key indicator to quantify the quality of the SAP least squares solver for a given parameter configuration as well as the running time of the algorithm. For the SAP least squares solvers, we used a Python version prototype RandNLA package, PARLA [9], that provides the implementations for the SAP least squares solvers with the interface to control the abovementioned parameters.

We use multiple synthetic matrices and several real-world input matrices to test the efficacy of our tuning pipeline [1]. Our experimental results show promising results that GP-based BO approach is effective in tuning the parameters for RandNLA algorithms, in comparison with other primitives such as random search or grid search. Moreover, we also show that transfer learning can further improve the tuning efficiency by leveraging the data obtained from other input data matrices. For transfer learning, within the Bayesian optimization process, our tuner chooses the categorical variable, i.e., the SAP algorithm and the sketching operator, using the Upper Confidence Bound (UCB) bandit function, and then we apply a GP-based multitask learning technique [12], called Linear Coregionalization Model (LCM), in order to learn from historical samples within the same chosen category from the source matrices. That improves the tuning quality and cost, compared to non transfer learning-based tuning. Overall, the success of the empirical tuning approach suggests possible practical use cases. For example, users can use our autotuning pipeline in order select the parameters for running a RandNLA algorithm. If the user has a larger dataset size, the user can down-sample their input data and perform autotuning (with or without transfer learning), and then use the chosen parameter configuration to run the algorithm on a larger dataset.

For future work, our tuning pipeline can be extended or tested for other RandNLA problems. While our experiments have primarily focused on the problem of overdetermined least squares, the basic lessons from our work are applicable in other contexts, such as low-rank approximation, and also for tuning large-scale high-performance computing applications. In addition, our tuning pipeline can further be improved to be even more robust and effective in tuning RandNLA workloads that are hard to achieve valid parameter configurations for a given residual accuracy requirement. From a theoretical perspective, the integration of surrogate-based optimization techniques with RandNLA algorithms opens up new avenues for research at the intersection of machine learning and numerical linear algebra. We can also explore how these autotuning techniques could be incorporated directly into adaptive algorithms, allowing numerical methods to automatically adjust their behavior based on the properties of the input data. In conclusion, the development of these surrogate-based autotuning techniques represents a significant step forward in bridging the gap between theoretical advances in RandNLA and their practical performance engineering.

References

[1] Y. Cho, J. W. Demmel, M. Dereziński, H. Li, H. Luo, M. W. Mahoney, R. J. Murray, *Surrogate-based Autotuning for Randomized Sketching Algorithms in Regression Problems*, in arXiv:2308.15720 (2023).

[2] H. Avron, P. Maymounkov, and S. Toledo, *Blendenpik: Supercharging LAPACK's Least-Squares Solver*, SIAM Journal on Scientific Computing, 32 (2010).

[3] X. Meng, M. A. Saunders, and M. W. Mahoney, *LSRN: A parallel iterative solver for strongly over- or underdetermined systems*, SIAM Journal on Scientific Computing, 36 (2014).

[4] M. Pilanci and M. J. Wainwright, *Newton Sketch: A near linear-time optimization algorithm with linear-quadratic convergence*, SIAM Journal on Optimization, 27 (2017).

[5] A. Dasgupta, R. Kumar, and T. Sarlos, *A sparse Johnson-Lindenstrauss transform*, in Proceedings of the Forty-Second ACM Symposium on Theory of Computing (STOC), STOC '10, 2010, Association for Computing Machinery, p. 341–350.

[6] M. Dereziński, Z. Liao, E. Dobriban, and M. Mahoney, *Sparse sketches with small inversion bias*, in Conference on Learning Theory (COLT), PMLR, 2021, pp. 1467–1510.

[7] M. Dereziński, J. Lacotte, M. Pilanci, and M. W. Mahoney, *Newton-LESS: Sparsification without trade-offs for the sketched Newton update*, Advances in Neural Information Processing Systems, 34 (2021).

[8] C. C. Paige and M. A. Saunders, *LSQR: An algorithm for sparse linear equations and sparse least squares*, ACM Trans. Math. Softw., 8 (1982), pp. 43–71.

[9] Ballistic, *Python Algorithms for Randomized Linear Algebra (PARLA)*, 2022, `https://github.com/BallisticLA/parla/tree/main`.

[10] R. Murray, J. Demmel, M. W. Mahoney, N. B. Erichson, M. Melnichenko, O. A. Malik, L. Grigori, P. Luszczek, M. Dereziński, M. E. Lopes, T. Liang, H. Luo, and J. Dongarra, *Randomized Numerical Linear Algebra : A perspective on the field with an eye to software*, arXiv:2302.11474v2 (2023).

[11] Y. Cho, J. W. Demmel, G. Dinh, X. S. Li, Y. Liu, H. Luo, O. Marques, and W. M. Sid-Lakhdar, *GPTune user guide*. `https://gptune.lbl.gov/documentation/gptune-user-guide`, 2022.

[12] Y. Liu, W. M. Sid-Lakhdar, O. Marques, X. Zhu, C. Meng, J. W. Demmel, and X. S. Li, *GPTune: Multitask learning for autotuning exascale applications*, in Proceedings of the 26th ACM SIGPLAN Symposium on Principles and Practice of Parallel Programming, PPoPP '21, 2021, Association for Computing Machinery, pp. 234–246.