Reproducibility of Fast and Accurate Least-Mean-Squares Solvers

Anonymous Author(s)
Affiliation
Address
email

Abstract

Least-mean squares(LMS) solvers, one of the most elementary supervised learning algorithms, have been heavily used in solving various practical issues due to its high interpretability and nice mathematical closed-form solution. However, in many real world cases, computing the covariance matrix to produce such solution becomes impossible due to some reasons.
In this project, we investigated reproducibility of results from a paper submitted and accepted by 2019 Neural Information Processing Systems (NeurIPS), named *Fast and Accurate Least-Mean-Squares Solvers* [Maalouf et al., 2019], which proposes a novel method to derive a much smaller but maintainable covariance matrix without accuracy loss, based on a new time-efficient implementation of Caratheodory's Theorem.
In our study, we first reproduce the tests' results in the paper and examine the effect of the method. And our experiments on extension of the method reveals some property and limitations in dealing with new cases.

1 INTRODUCTION

1.1 Preliminary

Information explosion, firstly indicated and used by the Online Oxford English Dictionary in 1964 [1964], represents the rapid increase in the amount of published information and data. The problem of managing such an enormous amount of data turns to be a worldwide interest, also give birth to and inspire couples of interdisciplinary subjects. Thus, it is of huge significance to expand existing mathematical tools to apply into a big data issues, which will assist to solve a lot of practical problems in machine learning.

Least-Mean-Squares (LMS) solvers are the family of multiple optimization and regression models. containing Linear Regression, Principal Component Analysis, Lasso and Ridge Regression, Elastic Net and many more. LMS solvers have been extensively utilized in solving the classification and regression of practical big data problem, such as graph theory [Zhang and Rohe, 2018], spectral clustering [Peng et al., 2015] and many more.

Generally, there are two main approaches to a LMS solver. Like in a typical Linear Regression model:

$$y = w_0x_0 + w_1x_1 + ... + w_nx_n = \sum_{m=0}^{n} w^T x$$

$$\text{LMSError}(w) = \frac{1}{n}||y - W^T X||^2$$

i. Normal equations (closed-form solution) $\hat{w} = (X^T X)^{-1} X^T y$

Submitted to 33rd Conference on Neural Information Processing Systems (NeurIPS 2019). Do not distribute.

ii. Optimization algorithm (Gradient Descent, Stochastic Gradient Descent, Newton's method, etc.)

Compared with optimization algorithm like gradient descent, closed-form equation can provide an "exact" solution. However, a conundrum has existed in this solution due to limit of computation power in LMS solvers— matrix multiplication and inversion. Applying matrix multiplication, which takes $O(m^2n)$ ($X \in \mathbb{R}^{m\times n}$), might result in a series of problems when the dataset is enormous. (details in related work)

Here, we reproduced a fast caratheodory method with coreset and sketch approximation, which is capable of reducing the size of dataset to a small subset. The covariance matrix from matrix multiplication of new subset is equal to the original dataset. In this way, the closed-form solution is able to deal with the big data issues, provide solid and reliable solutions. And we tested this method with four LMS solvers — Linear/Ridge/Lasso Regression and Elastic Net.

1.2 Related Work

In the closed-form solution, the essential operation comes to the matrix multiplication.

Issues in Covariance Matrix Computation There are two mathematical methods to calculate this covariance matrix: (1) Directly compute the outer product of each row in matrix X: $X^TX = \sum_{i=0}^{n} a_i a_i^T$ or (2) factorization of matrix X by singular value decomposition or QR decomposition: $X^TX = VD^2V^T$. However, in method (1), each time we operate an addition in outer product, numerical errors will be introduced and accumulated over time. The errors will increase to a significant amount especially when 32-bit floating number data type is used in the algorithm, and finally influence the whole accuracy of convariance matrix. A possible solution to this issue is Cholesky decomposition [Pourahmadi, 2007], but it applies strict restriction on the data matrix [Dostal et al., 2011], which makes it not useful in real cases. On the other hand, though method (2) make us worry less about the issue of accuracy loss, we could not ignore both decomposition method prolong the computation time by introducing additional matrix for factorization so that the hidden constant behind $O(nd^2)$ would be pretty large.

Caratheodory's Theorem first published on 1907, states that any point $a \in \mathbb{R}^d$ lies in the convex hull of set P, can be written in convex combination of at most $d+1$ points in P [Carathéodory, 1907]. This theorem indicates that if we have a dataset containing n data and d features (X is a $n \times d$ matrix and always $n >> d$), it is possible to find a smaller $(d^2+1) \times d$ matrix S, whose covariance matrix is the same as X. Since each row in X^TX (n points in total) can be considered as a point in $\mathbb{R}^{d^2}$, so d^2+1 points are required to represent these n points. In this way, both the numerical error and time cost will significantly reduce. Unfortunately, it takes $O(nd^3)$ or $O(n^2d^2)$ to calculate the caratheodory's set in LMS solvers [Binder et al., 2014], which is still not affordable when the dataset is so huge.

Coresets and Sketches for Approximation A coreset is a reduced data set that can be used as proxy to the full data set. Same algorithm runs on coreset is supposed to provide approximate result as on full data set. A sketch is a compressed mapping of the full data set onto a data structure which is easy to update or change data, also allow approximate result when certain queries are on the sketch and full data set [Phillips, 2016]. So a matrix S is a coreset when its rows are a weighted subset of a larger matrix X, or a sketch when each row of S can be written in a combination of rows in X.

An obvious advantage of coresets over sketches is that coresets is capable of preserving the sparsity of the original large matrix, which will significantly reduce the computation time [Feldman et al., 2016]. Unfortunately, many new methods nowadays proposed to provide approximations to the covariance matrix give rise to some multiplicative errors and eigenvalues of the reduced matrix could departs evidently from the original matrix [Drineas et al., 2006]. The new method, based on Caratheodory's Theorem, discussed here addresses this problem and provide an accurate approximation via both coresets over sketches that also maintains the ability to handling data streams.

1.3 Task Introduction

The paper provided an algorithm which:

i. Using a novel approach, which combine the coresets and sketches approximations together, to discover the Caratheodory's set only taking $O(\log n)$ calls to LMS solvers.

ii. The new matrix $S \in \mathbb{R}^{(d^2+1)\times d}$ based on the former step, is a weighted subsets of the input data matrix $A \in \mathbb{R}^{n\times d}$. Plus, the covariance matrix of S and A are the same: $S^T S = A^T A$.

iii. Validating this S matrix with Linear/Ridge/Lasso Regression and Elastic Net.

Our task is to:

i. Examining the validity of this novel approach by reproducing the results of the experiments mentioned in paper and comparing the values we obtain.

ii. Applying this method to new datasets and exploring the power of the method in dealing with more complex problems.

iii. Extending and changing the parameters of the methods and situations it applies to, discussing and testing the limitations and requirements for this method to be useful.

iv. In order to change the parameter α (regularization item) and obtain its effects on the model, we need to re-implement partial model. (α is a pre-determined constant in default implementation).

v. Re-implementing the part to create coresets and compare the performance with default implementation from authors.

2 DATASET AND SETUP

2.1 Dataset for Test

All 3 sets of data for testing in the experiment are public available. (And they are the same in original paper for reproducing test)

i. 3D Road Network (North Jutland, Denmark) [2015]. It contains 434874 instances with 4 types of information. We choose two — longitude and latitude to predict the height.

ii. Individual household electric power consumption Data Set [2017]. It contains 2075259 instances with 9 types of information. We choose two — global active power and global reactive power to predict the voltage.

iii. House Sales in King County (USA) [2014-2015]. It contains 21614 instance with 21 types of information. We choose eight — bedrooms, living area, lot size, floors, waterfront, above area, basement area and built year to predict the house price.

The remaining datasets used are created by random number generation with the function provided in package numpy. Since the focus here is examining the effect of time saving in LMS solver, it wouldn't be an issue to use synthetic data.

2.2 Data Preprocess

Errors, like NaN, are removed from the datasets. Data type casting is used to satisfy the input requirements. For synthetic data, different sample sizes n and parameters' values $\alpha \in \mathbb{A}$ and feature size d is chosen in order to create more artificial data and investigate the power of this new method.

3 EXPERIMENT AND RESULTS

3.1 Environmental Setup

In this experiment, we first download code file named "Booster" provided by the authors of this paper and find the datasets used in authors' experiments on the Internet. The experiments with default implementation is run on the 2018 MacBook Pro with processor - 2.3 GHz Intel Core i5 - and memory - 16 GB 2133 MHz LPDDR3. And re-implementation of the same algorithm based on Caratheodory's Theorem is run on the Lenovo Thinkpad Yoga with Intel(R) Core(R) i7-4600U @ 2.10GHz processor and 8GB DDR3 1600MHz memory. We all use CPython distribution.

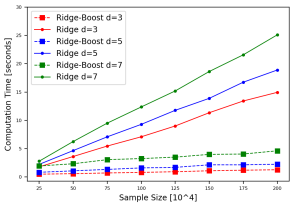
This limited the parameters' order of magnitudes we could test and the environment to examine effect of the method on streaming data is hard to set up. Whereas, it is enough for many available choices of the combination of parameters we could select to test, reproducing parts of its results and showing the power this novel approach bears in solving many issues.

3.2 Our Experiment

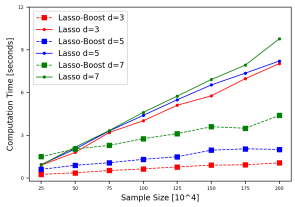
Our experiments are also based on four widely used LMS solvers Linear / Ridge / Elastic / Lasso-Regression.

3.2.1 Reproduction and Method Validity

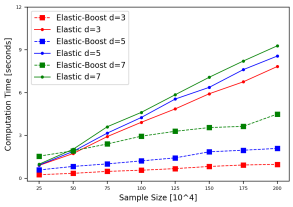
We follow the steps from the paper, use existing datasets and construct new synthetic datasets with the same parameter. The result obtained from the LMS solvers with or without the boosting of this novel method is recorded. Also, departure in accuracy is collected in testifying the effect of the method on numerical stability. Then, we compare the result with the ones shown in the paper. Fig.1



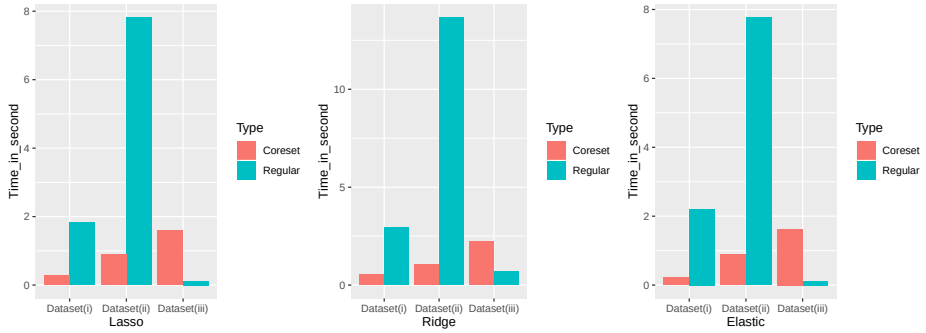
(a) Ridge Regression



(b) Lasso Regression



(c) Elastic Net



(d) The runtime for three datasets with and without boosting when alpha is set to 100

Figure 1: Reproducibility tests on default method implementation

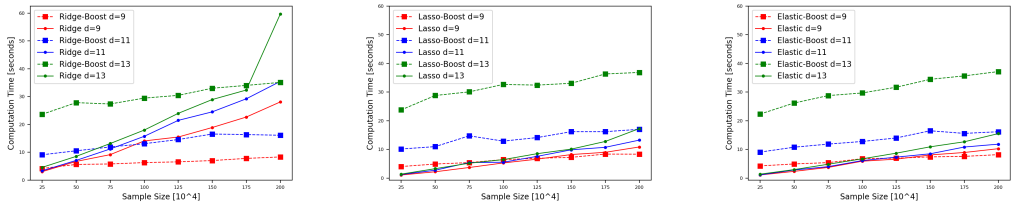
3.2.2 Hyperparameter and Extension

After carefully examining the content of the paper, it is not hard to imagine that there could be many cases different from the ones shown and used in the experiments displayed in the paper. In real world, situations might be more complex and so it is reasonable and intuitive to study the impact of different sample size n, feature size d and the number of alpha $|A|$ with the aim of justifying the method's flexibility and improvement.

The Fig.2 shows the method's influence when facing the cases with feature size d goes large with three LMS solver. Next, n is changed and we expect to find how large the n should be for this novel approach boosted regression model to out-compete the unvarnished one Fig.3.

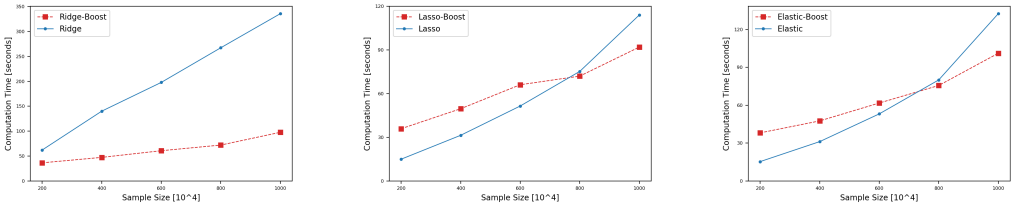
3.2.3 Optimized Re-Implementation

After re-implementing algorithms detailed in the default paper, we have found some differences on performance between the re-implementation and the author's default implementation. Notably, we've found slight increase of performance, due to cleaner code. In Fig.4, we can notice a very slight



(a) Ridge Regression (b) Lasso Regression (c) Elastic Net

Figure 2: Time comparison of changing d on different regression models on default method implementation



(a) Ridge Regression (b) Lasso Regression (c) Elastic Net

Figure 3: Time comparison of changing sample size n on different regression models on default method implementation

decrease in computation time, but due to hardware and small data samples, we also observe a increase in instability of the performance.

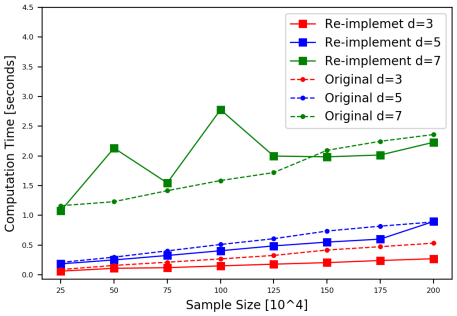


Figure 4: The effect of d on re-implement and original model

Additionally, our implementation allows us to tweak the value of k as describe in the algorithm formulation, whereas in the default implementation, k is fixed as $2d^2 + 2$. This let us explore the accuracy/speed trade-off by varying k as hinted in the paper. In Fig.5, we observe that computation time increase with the value of k. Note that values lower than $2d^2 + 2$ causes sometimes unexpected behaviour, such as non convergence of the algorithm. Thus we confirm that a value of $2d^2 + 2$ as theorized by the authors is indeed optimal.

3.3 Results

We successfully acquire the similar result displayed in the paper Fig.1 with the default implementation, which is a good indication for the time saving effect brought by this method. Also there is no accuracy loss, as a result of the merge-and-reduce property of this method. See Fig.6. The result is further testified by our re-implementation of the methods, which achieves even higher speed compared to the default implementation provided on the datasets without cross validation.

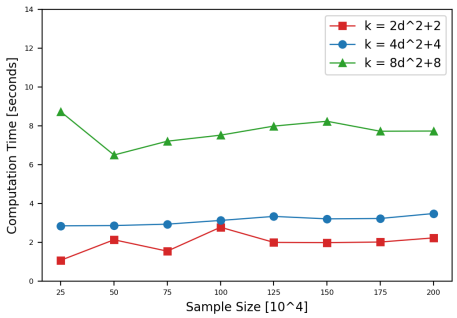


Figure 5: The effect of k on re-implement model. $d = 5$.

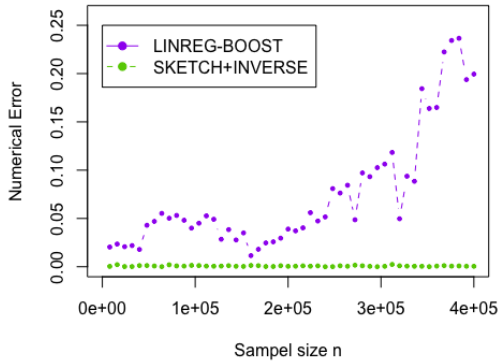


Figure 6: Numerical error on default method implementation

After finishing the experiments on new datasets new parameters, we discover that the boosting even exerts some negative influence as it is indicated in Fig.2 and Fig.3 suggests that in order for the boosting effect to be positive again, the demand of sample size increases dramatically.

4 DISCUSSION AND CONCLUSION

In this project, after we reproduced part of results shown in the paper, re-implemented the method codes, examined the robustness of this novel approach and investigated its limitations, we have many interesting discoveries.

The building up of coreset matrix with around d^2 rows within $O(nd^2)$ times does make acceleration possible for covariance matrix computation and there is no accuracy loss ignoring $\pm 10^{-16}$ Fig.6 Applying coreset reveals its effectiveness even when the sample size is not large (Fig.1), but the effectiveness remains restricted when dealing with more complicated cases.

We originally intended to also experiment on the effect of regularization. α is the parameter that tunes the regularization on the ridge/lasso regression and elastic net models. The process to explore a well-performance α value is expected to optimize the LMS solvers models. However, we found that the original testing framework used by the authors selects alpha automatically based on results of cross-validation. We've attempted to create a new testing environment without success. We've found unexpected errors, caused by unclear details of the default implementation. We couldn't produce any interesting and consistent results, since the overall computation time increased and the models' accuracy decreased.

In real world, when adopting the linear regression or other LMS solvers, we could be confronted with cases where sample size is limited, where many features that might need to be preprocessed, or

both. And it is not rare for such cases to happen when we apply machine learning to problems in various fields such as advertising or bioinformatics [Saeys et al., 2007]. Cases where some features requires to be combined as a polynomial or simply where there is a need of slightly larger feature size to make accurate predictions would result in a rapid growth in the number of features and reduces this method's effectiveness, as it is indicated by Fig.2. Although the effect of a relatively large feature size can be offset by increasing the sample size Fig.3, the cost in collecting more samples or the circumstances under which we choose LMS solvers such as in problems like text classification [Sordo and Zeng, 2005] render this novel method impractical.

Notwithstanding, there exits many restrictions for this approach, it remains powerful when we have high demand for numerical accuracy on calculation results. Usually, when the feature size goes up, more computation time will be required. If the problem requires very low numerical error and computation time is not an concern, this approach would probably be our ideal choice [Xue et al., 2014]. In addition, there exists many dimension reduction techniques which can help, in addition of enough domain knowledge, reduce the feature size and fully utilise this approach. [Bharti and Singh, 2015].

5 Further Development

There are many potential improvements on this novel LMS booster method. For example, more efficient and complete implementation of the code is possible. Higher dimension data might introduce some other unexpected issues and must be considered carefully. And due to some limitations, the effect of this method with streamed or distributed data, or GPU acceleration requires more testing.

6 STATEMENT OF CONTRIBUTIONS

All members have made significant contributions to the project. The distribution of work for each member is described as follows:

Jiewen Liu : Model modification, experimentation, report writing.

Jianchen Zhao : Algorithm re-implementation, experimentation, report writing.

Zhenzhe Zhang : Publication discovering, data organization, report writing.

References

Alaa Maalouf, Ibrahim Jubran, and Dan Feldman. Fast and Accurate Least-Mean-Squares Solvers. *arXiv e-prints*, art. arXiv:1906.04705, Jun 2019.

information explodation. 1964. URL `https://en.wikipedia.org/wiki/Information_explosion`.

Yilin Zhang and Karl Rohe. Understanding Regularized Spectral Clustering via Graph Conductance. *arXiv e-prints*, art. arXiv:1806.01468, Jun 2018.

Xi Peng, Zhang Yi, and Huajin Tang. Robust subspace clustering via thresholding ridge regression, 2015. URL `https://www.aaai.org/ocs/index.php/AAAI/AAAI15/paper/view/9416`.

Mohsen Pourahmadi. Cholesky Decompositions and Estimation of A Covariance Matrix: Orthogonality of Variance–Correlation Parameters. *Biometrika*, 94(4):1006–1013, 12 2007. ISSN 0006-3444. doi: 10.1093/biomet/asm073. URL `https://doi.org/10.1093/biomet/asm073`.

Zdenek Dostal, Tomás Kozubek, Alexandros Markopoulos, and Martin Mensík. Cholesky decomposition of a positive semidefinite matrix with known kernel. *Applied Mathematics and Computation*, 217:6067–6077, 03 2011. doi: 10.1016/j.amc.2010.12.069.

C. Carathéodory. Über den variabilitätsbereich der koeffizienten von potenzreihen, die gegebene werte nicht annehmen. *Mathematische Annalen*, 64(1):95–115, Mar 1907. ISSN 1432-1807. doi: 10.1007/BF01449883.

Ilia Binder, Cristobal Rojas, and Michael Yampolsky. Computable carathéodory theory. *Advances in Mathematics*, 265:280 – 312, 2014. ISSN 0001-8708. doi: https://doi.org/10.1016/j.aim.2014.07.039. URL http://www.sciencedirect.com/science/article/pii/S0001870814002813.

Jeff M. Phillips. Coresets and Sketches. *arXiv e-prints*, art. arXiv:1601.00617, Jan 2016.

Dan Feldman, Mikhail Volkov, and Daniela Rus. Dimensionality reduction of massive sparse datasets using coresets. pages 2766–2774, 2016. URL http://papers.nips.cc/paper/6596-dimensionality-reduction-of-massive-sparse-datasets-using-coresets.pdf.

Petros Drineas, Michael W. Mahoney, and S. Muthukrishnan. Sampling algorithms for l2 regression and applications. pages 1127–1136, 2006. URL http://dl.acm.org/citation.cfm?id=1109557.1109682.

Ryan A. Rossi and Nesreen K. Ahmed. The network data repository with interactive graph analytics and visualization. In *AAAI*, 2015. URL http://networkrepository.com.

Dheeru Dua and Casey Graff. UCI machine learning repository, 2017. URL http://archive.ics.uci.edu/ml.

House sales in king county, usa, 2014-2015. URL https://www.kaggle.com/harlfoxem/housesalesprediction.

Yvan Saeys, Iñaki Inza, and Pedro Larrañaga. A review of feature selection techniques in bioinformatics. *Bioinformatics*, 23(19):2507–2517, 08 2007. ISSN 1367-4803. doi: 10.1093/bioinformatics/btm344. URL https://doi.org/10.1093/bioinformatics/btm344.

Margarita Sordo and Qing Zeng. On sample size and classification accuracy: A performance comparison. In José Luís Oliveira, Víctor Maojo, Fernando Martín-Sánchez, and António Sousa Pereira, editors, *Biological and Medical Data Analysis*, pages 193–201, Berlin, Heidelberg, 2005. Springer Berlin Heidelberg. ISBN 978-3-540-31658-9.

J. Xue, J. Li, D. Yu, M. Seltzer, and Y. Gong. Singular value decomposition based low-footprint speaker adaptation and personalization for deep neural network. In *2014 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 6359–6363, May 2014. doi: 10.1109/ICASSP.2014.6854828.

Kusum Kumari Bharti and Pramod Kumar Singh. Hybrid dimension reduction by integrating feature selection with feature extraction method for text clustering. *Expert Systems with Applications*, 42 (6):3105 – 3114, 2015. ISSN 0957-4174. doi: https://doi.org/10.1016/j.eswa.2014.11.038. URL http://www.sciencedirect.com/science/article/pii/S0957417414007301.