Feature analysis for selection of implementations in an accurate matrix-matrix multiplication library

Shota Aoki
Nagoya University
Japan
aoki@hpc.itc.nagoya-u.ac.jp

Takahiro Katagiri
Nagoya University
Japan
katagiri@cc.nagoya-u.ac.jp

Satoshi Ohshima
Nagoya University
Japan
ohshima@cc.nagoya-u.ac.jp

Toru Nagai
Nagoya University
Japan
nagai@cc.nagoya-u.ac.jp

1. INTRODUCTION

In this study, we aim to develop a software auto-tuning (AT) mechanism with AI to reduce the man-hours required for performance tuning in the field of numerical computations. We show an example of explainability of a result by machine learning (ML) when ML is applied to a case study of performance parameter tuning on a library for verified numerical computations.

2. OZAKI METHOD

The following (1) and (2) are performed on the input matrices in Ozaki method [2], which is a numerical algorithm for accurate matrix-matrix multiplication (MMM): (1) Split the matrices *A* and *B* into *p* and *q* matrices, respectively. (2) Compute split matrices' products A_jB_k $(j=1, \dots, p, k=1, \dots, q)$, and add them using the accurate sum algorithm.

The matrix product in the process (2) is transformed into an error-free operation with this method [2]. In addition, on the splitting process of process (1), a sparse matrix is generated according to range of elements of the input matrices:

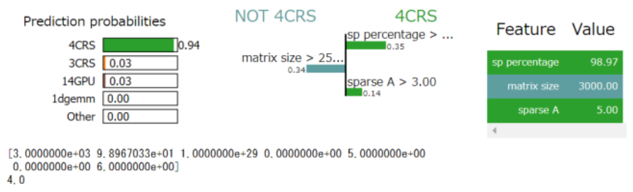
In this study, for the MMM in the process (2), we use a dgemm on CPU and GPU. In addition, 11 implementations of sparse matrix operations on CPU or GPU are utilized[3]. We constructed a machine learning (ML) model to predict which implementation is the fastest. We explained the prediction results of the classifier using LIME [1], which can examine how much each feature contributes to the classification.

3. PRELIMINARY RESULT

We used the supercomputer "Flow" Type II subsystem installed at the Information Technology Center, Nagoya University to acquire the training data for ML, and Version 0.2.0.1 of LIME was used.

The input data (matrices A and B) are: (1) a matrix whose elements are generated in range of 0 to 1, and values of pow (10, rand()% Φ) are inserted for some sparsity elements (the upper limit is Φ=15); (2) a unit matrix in which values in the range 0 to 1 are inserted for some sparsity elements (sparsity 90 to 98); (3) a unit matrix in which values pow(10, rand()%Φ) are inserted for some sparsity elements (upper limit is Φ=15);

The ML model (classifier) is xgboost, ver.1.4.2, and we predict the fastest implementation of those described in Chapter 2. The matrix sizes are 1000, 1500, 2000, 2500, 3000, and 4000 (only (2)). Numbers of training data and test data are 199 and 23, respectively.



```
[3.0000000e+03 9.8967033e+01 1.0000000e+29 0.0000000e+00 5.0000000e+00
 0.0000000e+00 6.0000000e+00]
4.0
```

Fig. 1. A result of LIME.

There are following 7 explanatory variables for the classifier: 1) matrix size; 2) sparsity; 3) the maximum element of *A*; 4) the minimum element of *A*; 5) number of sparse split matrices of *A*; 6) number of dense split matrices of *A*; 7) number of split matrices of *B*.

Accuracy of the model is from 83% to 96% depending on how the data is taken, and the results of LIME when the accuracy is about 96% are shown in Fig. 1. In Fig. 1, SpMV with CRS (multiple right-hand-sides, internal parallelism) on CPU is the fastest on the prediction. High sparsity was analyzed as a positive factor. In fact, this implementation is often faster when the sparsity is high. Hence this is reasonable explanation. Large matrix size was also analyzed as a negative factor. In fact, when the matrix size is larger than 2500, SpMM with CRS on GPU is often the fastest. Hence this indicates reasonable explanation.

ACKNOWLEDGMENTS

This work was supported by JSPS KAKENHI (Grant No. JP19H05662), and was supported by "Joint Usage/Research Center for Interdisciplinary Large-scale Information Infrastructures" in Japan (Project ID: jh210002-NAHI).

REFERENCES

[1] M. T. Ribeiro, S. Singh, and C. Guestrin: Why should I trust you?: Explaining the predictions of any classifier, Proc. of 22nd ACM SIGKDD, pp.1135-1144, 2016.

[2] K. Ozaki, T. Ogita, S. Oishi, S. M. Rump: Error-Free Transformation of Matrix Multiplication by Using Fast Routines of Matrix Multiplication and its Applications Numerical Algorithms, Vol. 59, No. 1, pp. 95-118, 2012.

[3] F. Ishiguro, T. Katagiri, S. Ohshima, T. Nagai: Performance Evaluation of Accurate Matrix–Matrix Multiplication on GPU Using Sparse Matrix Multiplications, Proc. of CANDARW2020, IEEE Xplore, 2020.