Predicting the convergence of BiCG method from grayscale matrix images

Ryo Ota\(^1\) and Hidehiko Hasegawa\(^2\)

\(^1\) Graduate School of Library, Information and Media Studies, University of Tsukuba, 1-2 Kasuga, Tsukuba, Ibaraki 305-8550, Japan
\(^2\) Faculty of Library, Information and Media Science, University of Tsukuba, 1-2 Kasuga, Tsukuba, Ibaraki 305-8550, Japan

E-mail \(^1\) s1411506@klis.tsukuba.ac.jp, \(^2\) hasegawa@slis.tsukuba.ac.jp

Received March 6, 2020, Accepted June 6, 2020

Abstract

The convergence of the BiConjugate Gradient (BiCG) method depends on its input matrices. We tried to predict the convergence of BiCG method by applying a Convolutional Neural Network to matrices that had been converted to grayscale images. Using 875 real non-symmetric matrices in the SuiteSparse Matrix Collection, we applied the 5-fold cross-validation method and were able to predict convergence with an average accuracy that exceeded 80% for all cases in the test collection.

Keywords  pattern recognition, classification of matrices, iterative methods

Research Activity Group  Algorithms for Matrix / Eigenvalue Problems and their Applications

1. Introduction

In the field of scientific and technical computing, numerical approximation problems can often be reduced to solving a system of linear equations. The BiConjugate Gradient (BiCG) method is widely used for solving a large system of linear equations with a real non-symmetric sparse matrix. However, in any given case, it is not clear which particular combination of iterative solver and matrix is optimal [1]. Sometimes the BiCG method will converge within \(n\) (the matrix dimension) iterations, but sometimes it does not converge at all. If the likelihood that the BiCG method would converge could be roughly predicted in advance, then a better alternative solver could be applied if BiCG is unlikely to achieve convergence. Currently, techniques involving convolutional neural networks (CNNs) [2] are producing remarkable progress within the field of image recognition and computer vision. A CNN extracts the features in images and, by learning the extracted pattern, makes it possible to classify or predict values for previously unseen images. Building on this approach, if sparse matrices are converted to images holding enough information about their nonzero value patterns [3, 4], we can apply machine learning to predict the convergence of the BiCG method. Of course this strategy requires a good way of converting the magnitude of the nonzero values in matrices to grayscale images.

To test this approach, we apply the BiCG method to each matrix available in the SuiteSparse Matrix Collection [5] and record convergence or non-convergence. We create grayscale matrix images using the modified SuiteSparse method and give them labels corresponding to the convergence results. After that, we try to learn relationship between matrices’ grayscale images and their convergence of BiCG method. The system is evaluated by the 5-fold cross-validation method. As a result, the average accuracy for the all cases of using 875 matrices exceeds 80%.

2. Framework and dataset

2.1 Framework of deep learning

Deep learning is a kind of machine learning that deepens the structure of neural networks. Recently, CNNs have become a popular deep learning model in the field of image recognition. As a framework for the research presented here, TensorFlow [6] is typically used to construct the CNN.

Fig. 1 shows an overview of the CNN in the work described here. The structure of the CNN is composed of 2 pairs of a convolutional layer and a maxpooling layer, 2 fully connected layers, and 1 output layer. The output layer gives the prior probability for the convergence class and the non-convergence class. The system predicts the result based on the maximum value of the prior probability. By comparing the probability of the two possible outcomes, we can estimate how accurately we can predict the success of the solver for each matrix.

In the learning process, the batch size is 16 and the number of training epochs is 30. The loss function is set as the cross entropy and the optimization method is the Adam method [7]. The learning rate is set as \(1.0 \times 10^{-3}\) and the probability of dropout is set as 25%. The process of dropout occurs in front of the fully connected layers to prevent the model from overfitting. The input image size is changed among four options, \(28 \times 28, 56 \times 56, 112 \times 112,\) and \(224 \times 224\) pixels. The gradation scale is set as 256 grayscale values from 0 to 255.
The BiCG method is used to solve the system of linear equations:

\[ Ax = b \]

where \( x \) is a vector and the coefficient matrix \( A \) and the right-hand side vector \( b \) are constants. The solution vector is given by \( x = [1, -1, 1, -1, \ldots, (-1)^{(n-1)}]^T \), and the right-hand side vector \( b \) is computed as \( b := Ax \). The initial solution vector is set to \( x_0 = [0, 0, \ldots, 0]^T \). The convergence criterion is defined as the relative residual norm being less than or equal to \( 10^{-6} \). The maximum number of iterations is set to \( n \), where \( n \) is the matrix dimension of \( A \). The results of a numerical experiment with the BiCG method using the above conditions are shown in Table 1. 235 matrices (27%) converged and 640 matrices (73%) which include singular matrices, did not converge.

### 2.4 Details of the datasets

We constructed two datasets, one having a complete set of convergence matrices and the same number of non-convergence matrices (470 matrices in total, denoted by C), and the other using all 875 matrices (denoted by G). In dataset C, 235 matrices are selected randomly from the non-convergence matrices. In order to apply 5-fold cross-validation, we split whole matrices in both datasets into 5 groups, each containing the same number of matrices. In this experiment, the \( C_i \), which means the \( i \)-th group of dataset \( C \), is a subset of \( G_i \), and the members of each group are fixed.

### 3. Matrix Image

Color matrix images can be generated using the software SuiteSparse [8], which is provided by the SuiteSparse Matrix Collection. This method is not for classifying the matrices but for visualizing the positions of the nonzero elements and the magnitude relationship of its elements. To produce grayscale images used for pattern recognition in machine learning, we developed the modified SuiteSparse method. Grayscale values of 0 and 255 correspond to black and white, respectively. The modified SuiteSparse method for generating the required images is described as follows: Let \( d \) represent the image size, \( s = \lfloor n/d \rfloor \) represent the block size, and \( m = \lfloor n/s \rfloor \) represent the size of working matrix.

1. Construct \( ms \times ms \) block matrix \( A^{(1)} \), which has non-negative values, by taking the absolute value of \( A \). \( A^{(1)} \) includes \( m \times m \) blocks, and the size of each blocks (called \( B \)) is \( s \times s \).
2. Construct \( m \times m \) matrix \( A^{(2)} \) and set of maximum value of each blocks of \( A^{(1)} \), which means \( a_{i,j}^{(2)} \) is set the maximum element of \( B_{i,j} \) \( (i - j \) block of \( A^{(1)} \)).
3. Compute the median \( Me \) and the standard deviation \( \sigma \) of the logarithm base 10 of the nonzero elements of \( A^{(2)} \). Construct \( m \times m \) matrix \( A^{(3)} \) from the nonzero elements of \( A^{(2)} \) by using following formula:

\[
\begin{align*}
a_{i,j}^{(3)} := & \max \left( 1, 128 + \left\lfloor 127 \log_{10}(a_{i,j}^{(2)}) - Me \right\rfloor \right) \\
\end{align*}
\]

If \( a_{i,j}^{(3)} \) is greater than 255 then set 255 to \( a_{i,j}^{(3)} \).
4. If the size of working matrix \( m \) is less than \( d \) then enlarge \( A^{(3)} \) to \( d \times d \) matrix \( A^{(4)} \) by using bicubic interpolation, otherwise copy \( A^{(3)} \) to \( A^{(4)} \).

This converts \( n \times n \) matrix \( A \) to \( d \times d \) grayscale matrix \( A^{(4)} \). The grayscale value of 0 originates from the zero element, and the grayscale value of 128 originates from the median of logarithm base 10 of nonzero elements of \( A^{(2)} \).

A distribution of the grayscale values for 875 matrices is shown in Fig. 3, which presents a histogram that corresponds to the case of 56×56-pixel images. The total number of nonzero grayscale values is 750,589 (24.7%). The number of grayscale values of 0 is 1,993,411 (72.6%), the number of grayscale values of 1 is 67,442 (2.4%), and the number of grayscale values of 255 is 158,146 (5.8%).
trices and each group of G consists of 175 matrices. A
datasets C and G. Each group of C consists of 94 ma-
4.2 Classification result

where TN+FN+FP+TP is equal to the number of ma-
the correct prediction. The accuracy of the cross-
lem, the "negative data" refers to the data that did
means False prediction for Positive data, and TP
means True prediction for Negative data, TN
means True prediction for Negative data, FN
means False prediction for Negative data, FP
in each group.

The mean of the distribution is 144.6 and the standard
deviation σ is 79.20.

4. Evaluation

4.1 K-fold cross-validation

K-fold cross-validation uses multiple training and test
datasets that cover all of the available data, and then av-
erages the results. The process of k-fold cross-validation
is as follows:
1. Divide the dataset into k groups randomly.
2. Select one of the groups as the test set, and use the
other groups as the training set.
3. Train the pattern using the training set and evaluate
the accuracy by the test set.
4. Repeat Steps 2 and 3 k times.
5. Calculate the average accuracy over the k iterations.

In this research, an experiment is done setting k = 5.

Table 2 is a confusion matrix often used in evaluation
of classification problems. In this confusion matrix,
TN means True prediction for Negative data, FN
means False prediction for Negative data, FP
means False prediction for Positive data, and TP
means True prediction for Positive data. In our prob-
lem, the "negative data" refers to the data that did
not converge in computation, and the "positive data"
refers to the data that did converge. Data, whether
positive or negative, is called "true" when it makes
the correct prediction. The accuracy of the cross-
validation is defined as (TN+TP)/(TN+FN+FP+TP)
where TN+FP+FN+TP is equal to the number of ma-
trices in each group.

4.2 Classification result

Table 3 shows a comparison of the accuracy for the
datasets C and G. Each group of C consists of 94 ma-
tices and each group of G consists of 175 matrices. A
group C_i is a subset of G_i. In Table 3, In all cases, the
average accuracy of G is higher than that of C by ap-
proximately 5.2%–8.5%. As this result shows, dataset
G, which consists of many more matrices, gives superior
predictions to dataset C. From these results, we con-
clude that even if the number of matrices of the dataset
in the non-convergence class is two or three times more
than that of the convergence class, the system can still
classify the images well. In Table 3, the 56-pixel case
gives the best accuracy.

The process of k-fold cross-validation as follows:
1. Divide the dataset into k groups randomly.
2. Select one of the groups as the test set, and use the
other groups as the training set.
3. Train the pattern using the training set and evaluate
the accuracy by the test set.
4. Repeat Steps 2 and 3 k times.
5. Calculate the average accuracy over the k iterations.

In this research, an experiment is done setting k = 5.

Table 2 is a confusion matrix often used in evaluation
of classification problems. In this confusion matrix,
TN means True prediction for Negative data, FN
means False prediction for Negative data, FP
means False prediction for Positive data, and TP
means True prediction for Positive data. In our prob-
lem, the "negative data" refers to the data that did
not converge in computation, and the "positive data"
refers to the data that did converge. Data, whether
positive or negative, is called "true" when it makes
the correct prediction. The accuracy of the cross-
validation is defined as (TN+TP)/(TN+FN+FP+TP)
where TN+FP+FN+TP is equal to the number of ma-
tices in each group.

4.2 Classification result

Table 3 shows a comparison of the accuracy for the
datasets C and G. Each group of C consists of 94 ma-
tices and each group of G consists of 175 matrices. A
group C_i is a subset of G_i. In Table 3, In all cases, the
average accuracy of G is higher than that of C by ap-
proximately 5.2%–8.5%. As this result shows, dataset
G, which consists of many more matrices, gives superior
predictions to dataset C. From these results, we con-
clude that even if the number of matrices of the dataset
in the non-convergence class is two or three times more
than that of the convergence class, the system can still
classify the images well. In Table 3, the 56-pixel case
gives the best accuracy.

Table 4 and 5 give more detailed information. The correct answer rate of TP (TP/(TP+FP+FN)) for conver-
gence matrices is from 78% to 89% in dataset C and
from 57% to 72% in dataset G. The correct answer rate
of TN (TN/(TN+FN)) for non-convergence matrices is
from 53% to 82% in dataset C and from 86% to 95%
in dataset G. The F measure of dataset C is 0.798 and
that of dataset G is 0.693. If a dataset includes more
non-convergence matrices and the ratio of convergence
matrices is decreased, the correct answer rate of TP is
decreased, but not by too much. From this result, we see
that making predictions about a smaller group is more
difficult than making them about a larger group. The
correct answer rate is a little different in each group.
The evaluation was performed on a system with Intel(R) Xeon(R) E5-1620 v3 CPU @ 3.50 GHz with 4 cores, an NVIDIA Tesla K40m GPU, and 64 GB of main memory. TensorFlow is installed on a Ubuntu16.04 LTS using both the CPU and GPU. Table 6 shows the computation time for training and testing one image in group G. This time does not include computation time for converting the matrix to an image, but the computation time for this is not a big issue.

### 5. Summary and future work

This research has presented a way of using CNN to predict convergence of the BiCG method when applied to grayscale matrix images. According to our evaluation, the average accuracy of 5-fold cross-validation is over 80% in all cases of using 875 matrices. Because of space restrictions, we did not show which types of matrices are predicted to converge easily, nor which type of matrices are difficult to predict convergence for.

We used only the modified SuiteSparse method for converting nonzero elements in sparse matrices to grayscale images. In this process, an absolute value was applied, which means that this method does not distinguish signs. A logarithm base 10 was also applied. Images produced by this method are very rough and precise information is not captured in the image, even though acceptable accuracy can be achieved. It is possible that even better accuracy could be achieved by tuning this conversion procedure.

Some ideas are proposed as near future work of this research:

- Changing the convergence criterion
- Modifying distribution of grayscale values
- Using other converting method
- Using multiple values instead of binary values
- Applying to other iterative methods
- Standardization of matrix

The convergence criterion $10^{-6}$ in the relative residual norm is somewhat loose. In practice, we should test more precise convergence criterion such as $10^{-8}$ or $10^{-12}$. Because the modified SuiteSparse method was the only method applied to convert matrix to image, it is possible that a different distribution of grayscale values produced by using a different method might achieve better accuracy. Using a sigmoid function to map numerical values in matrices to grayscale values in images would be a plausible alternative to try. In addition, this research focused on selecting whether the residual norm in the BiCG method converges or not. But some matrices that are nearly but did not completely converged are labeled as non-convergence. To find such matrices, a multi-class classification is needed. For example, in addition to the convergence and non-convergence classes, a new class, such as “convergence-but-difficult,” could be added. In this way the system would be enabled to make a more detailed prediction. Moreover, this research limited its exploration of iterative solvers to the BiCG method. If the same approach were applied to different solvers, its predictive power could be compared for several solvers. Matrix $A$ and $SA$, $S$ is diagonal matrix, will not have completely different convergence, but a grayscale image of $A$ and $SA$ is different. It is not clear that standardizing images achieves acceptable accuracy in this type of pattern recognition using machine learning, however some standardization of matrix images will be necessary. Preconditioned matrix is also a big problem.

These ideas will be further discussed in our future work.

### Acknowledgments

The authors would like to thank Terry Moore and Piotr Luszczek for their valuable comments. This research was supported by JSPS KAKENHI Grant Number JP17K00164.

### References

[1] R. Barrett et al., Templates for the Solution of Linear Systems: Building Blocks for Iterative Methods, SIAM, Philadelphia, 1994.
[2] A. Krizhevsky, I. Sutskever and G. E. Hinton, ImageNet classification with deep convolutional neural networks, in: Proc. of the 25th International Conference on Neural Information Processing Systems, 1097–1105, 2012.
[3] M. Götz and H. Anzt, Machine learning-aided numerical linear algebra: convolutional neural networks for the efficient preconditioner generation, in: 2018 IEEE/ACM 9th Workshop on Latest Advances in Scalable Algorithms for Large-Scale Systems (scalA), 49–56, 2018.
[4] K. Yamada et al., Preconditioner auto-tuning using deep learning for sparse iterative algorithms, in: 2018 Sixth International Symposium on Computing and Networking Workshops (CANDARW), 257–262, 2018.
[5] T. A. Davis, Y. Hu and S. Kolodziej, The SuiteSparse Matrix Collection, https://sparse.tamu.edu.
[6] M. Abadi et al., Tensorflow: A system for large-scale machine learning, in: 12th USENIX Symposium on Operating Systems Design and Implementation (OSDI 16), 265–283, 2016.
[7] D. P. Kingma and J. Ba, Adam: A method for stochastic optimization, arXiv:1412.6980 (cs.LG).
[8] T. A. Davis, SuiteSparse: A suite of sparse matrix software, http://faculty.cse.tamu.edu/davis/suitesparse.html.