3135 An effective classification approach for big data with parallel generalized Hebbian algorithm

By Ahmed Hussein Ali

An effective classification approach for big data with parallel generalized Hebbian algorithm

Article Info

Keywords:

Big data
Generalized Hebbian algorithm
Machine learning
Neural network
Principal component analysis
Spark Radoop

ABSTRACT

Advancements in information technology is contributing to the excessive rate of big data generation recently. Big data refers to datasets that are huge in volume and consumes much time and space to process and transmit using the available resources. Big data also covers data with unstructured and structured formats. Many agencies are currently subscribing to research on big data analytics owing to the failure of the existing data processing techniques to handle the rate at which big data is generated. This paper presents an efficient classification and reduction technique for big data based on parallel generalized Hebbian algorithm (GHA) which is one of the commonly used principal component analysis (PCA) neural network (NN) learning algorithms. The new method proposed in this study was compared to the existing methods to demonstrate its capabilities in reducing the dimensionality of big data. The proposed method in this paper is implemented using Spark Radoop platform.

1. INTRODUCTION

The problem of big data borders on their size, volume, and rate of generation from multiple sources (including machines and human) [1]-[13]. There are many forms of big data, such as web and social media data, business transaction data, machine-to-machine data, and biometric data [14]-[39]. Big data cannot be described just as a large database but is often unstructured and is currently on the increase in all domains. High dimensional input data streams are highly important for most information processing tasks, such as communication and pattern recognition, and can help in reducing noise and redundancy to allow for the extraction of useful information from input signals. Consequently, information processing, transmission, and storage on big software and hardware has become easier due to the ability to reduce data dimensionality. One of the common feature extraction methods is principal component analysis (PCA) which is used to extracts useful information through establishing the patterns in the input space. PCA is mainly aimed at obtaining the accurate data representation that can reduce the redundant components [40]-[47].

The PCA [48], [49] transform is mainly used for tasks such as pattern recognition, data compression, and classification. It is also called Karhunen-Loeve transform (KLT) [50]-[53]. Despite the wide application of numerous PCA-based algorithms [14], [54], most are not suitable for real time applications due to the high computational complexity of such algorithms in high dimension feature vectors. So, the computational speed of PCA can be improved by using a number of algorithms, even though the nagging problem. Most of these algorithms are only implemented as software to achieve moderate performance. PCA and its variants can also be implemented on hardware, but this requires enough resources and complex circuit control systems. Hence, it is only considered for small dimension. Implementation on PCA neural network (NN) is another alternative for PCA implementation [55]. This is done using the GHA [56] but the problem is the slow convergency of GHA which make it mandatory to perform several iterations, thereby prolonging the computational time for most GHA-based algorithms. Most data reduction techniques are exceptional in saving bandwidth and time through enabling user to process large datasets using minimal available resources. Being that much data is involved in data mining process, data reduction processes have become mandatory as the aim is to retrieve important information from such large datasets. Data size reduction is also a nagging problem because most of the straightforward techniques only work on small data and not on big data. Hence, software design stage is a crucial phase during the building of data reduction algorithms for big data processing.

The recent works on parallel big data dimensionality reduction are reviewed in this section. The study by [57] presented a hybrid PGO-SVM-based model that combined SVM with PGO for improved classification accuracy even when faced with small number of feature subsets. The proposed PGOSVM was implemented with Spark Radoop with distributed data points storage using Hadoop dispersed file system (HDFS). The classification efficiency of the proposed model on large dataset was better and exhibited faster execution time than the benchmark method. Scala was used as the programming language to implement the

PGOSVM while Covtype and Higgs datasets were analyzed. Another study present a fast HP-PL model [58] as a new way of improving DR and classification accuracy. The system was implemented on Apache Spark and was capable of selecting the best features within the shortest computational time. Even though the improvement level is dependent on the data features, the system showed good performance on the number of evaluated nodes for the tested datasets. The iterated PCA (IPCA) [59] method has been proposed for fault detection in a continuously stirred tank reactor (CSTR) model. The proposed IPCA relies on the GHA for memory complexity problems. The reason for addressing the fault detection problem is to facilitate online computation of the principal components in a recursive manner. The GHA was developed to define a function that can merge all the major factors that affect the fault detection capability of the developed model. Song et al proposed the TOC-based PCA algorithm [60] that can exploit the advantage of optical computing in big data computation to solve the issues related to the PCA algorithm in electronic computers. The parallel operation of the system ensured that the efficiency is greatly improved. Another paper by Jian *et al*. [59] demonstrated that the GHA has non-approaching adaptive learning rates by investigating the convergence of the GHA using the DDT method. It is simple to solve the computational roundoff constraints and satisfy the tracking requirements in real applications because these adaptive learning rates can achieve non-zero constants convergence. As a generalization of the Hebbian learning paradigm, Eraldo and colleagues [60] proposed a new adaptation strategy for linear neural networks. In this paper, an efficient classification and reduction technique for big data based on parallel generalized Hebbian algorithm (GHA) and implemented by using Spark Radoop platform will be presented. The new method proposed will be compared to the existing methods to demonstrate its capabilities in reducing the dimensionality of big data.

This paper is organized in this manner: apache spark radoop is presented in section 2. The principle of GHA and the suggested method are presented in section 3. The materials used in this work, and the methods employed are presented in section 4, and the results and discussion are presented in section 5 while section 6 presents the conclusion and possible future works.

2. APACHE SPARK RADOOP

RapidMiner Radoop is an extension of the in-memory functionality of RapidMiner that allows for the provision of sophisticated operators that are implementable for in-Hadoop execution [61]-[66]. It was developed as an extension of the in-memory functionality of RapidMiner for the provision of sophisticated operators that are implementable for in-Hadoop execution [67]-[73]. For data transformation in Radoop [61], there are more than 60 operators available. It is also capable of advanced and predictive modeling on Hadoop clusters in a distributed manner. RapidMiner [74] is a data mining application. Radoop relies on RapidMiner Studio's visual workflow designer to make the creation, implementation, and maintenance of predictive analytics in Hadoop as simple as possible. Because of Hadoop's code-free environment [62], [74] and built-in intelligence, the intricacies of the system are kept to a minimum, allowing the operator to concentrate solely on addressing business challenges rather than on technical concerns. This ensures that predictive analytics for both TBs and PBs of data is effective and scalable because the workflow execution is handled by Radoop rather than the user; all computations are executed in the Hadoop cluster that holds the data. Radoop was developed as an extension to ensure that Hadoop and RapidMiner could work together seamlessly. It is a data science software that simplifies the process of preparing data for machine learning on Hadoop and Radoop Spark (refer to Figure 1). Throughout RapidMiner Studio, all parallel operations and data processes are implemented on the SparkRM platform within the Hadoop cluster to ensure that Apache Spark may be used for task execution, hence broadening the applicability of the tool and enabling stronger algorithms. Hive and Mahout are made up of data analytics routines that have been well optimized, and as a result, they were used in this study as well. Figure 2 depicts the overall framework for the integration of Hadoop into RapidMiner. In this study, an extension was developed that allows for close connection with Hadoop while also providing the same Hadoop features as those used in memory-based RapidMiner operations. The initial stage in creating the Radoop is to include the RadoopNest meta-operator, which contains the basic cluster parameters. This meta-operator serves as a foundation for the operation of the remaining operators.

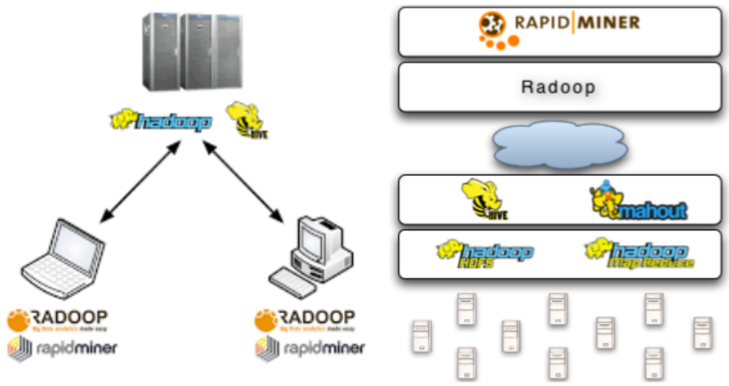


Figure 1. Spark Radoop architecture

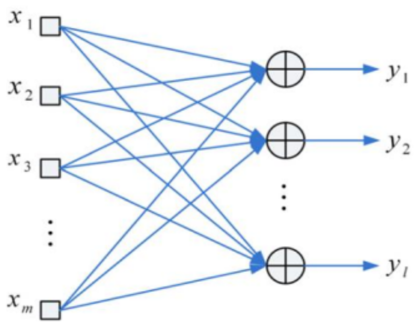


Figure 2. The neural model for the GHA

3. THE PROPOSED GENERALIZED HEBBIAN ALGORITHM

The GHA [75] is a linear feedforward NN framework that is well-suited for unsupervised learning applications and is often used in PCA. It is advantageous in terms of processing efficiency since it can handle the problem of eigenvalue using iterative approaches, which eliminates the need for direct computation of the covariance matrix. Because of the capacity to handle eigenvalue issues iteratively, there is no need to compute and answer eigenvalue issues in a linear fashion. As a result, GHA was created as a solution to memory complexity difficulties, particularly when dealing with large-scale data sets as shown in Figure 2. In order to provide a memory-efficient implementation, GHA is designed to be flexible and adaptable to time-varying distributions. Particle-counting analysis [75]-[81] is regarded as an attribute reduction method that is beneficial when dealing with data that is derived from numerous characteristics and contains some redundancy. Because they are most likely assessing the same concept, redundancy in this circumstance means that there is some type of correlation between some attributes. As a result of this redundancy, it is thought that the observed attributes can be reduced to a smaller number of PCs, each of which will be representative of the fluctuations in the observed characteristics. The PCA method uses orthogonal transformation to convert a set of data with linked qualities into a set of values referred to as principle components (uncorrelated attributes) [82]-[87]. Considering that the number of PCs is typically less than or equal to the number of original attributes, this transformation is defined in such a way that the variance of the first PC, which accounts for the majority of data variability, is as high as possible, and each of the succeeding components has the highest possible variance under the condition that it is orthogonal to the PCs [88].

This section pointed out some of the considerations for the implementation of the proposed algorithm with Radoop. Several steps are involved in the parallelization process. A virtual machine cluster was considered for the tuning and testing of the experimental conditions. The experiments were carried out on three different supercomputers. Because big data is taken into consideration in this work, it is probable that the dataset will contain a huge number of transactions. As a result, some of the large transactional data sets are kept in the HDFS, while numerous data fractions are distributed across the cluster nodes. The execution of jobs on data partitions is carried out in parallel by the Spark engine. We generated and processed a collection of RDDs in order to construct the set of frequently occurring l-iternsets, which were then arranged in descending order. The proposed PGHA is applicable in big data streaming using classification methods. It can be used to reduce all the stored dataset as HDFS files, and handle dataset with numeric features. Figure 3 presents the overall proposed algorithm for data reduction which can be implemented as Map and Reduce functions.

```
Proposed Parallel GHA
Input: S (Dense array)
Output: T (Reduced array)

Begin
    Execute the spark context (Slave)
    Listen to the master connection.
    Receive a dense array of data
    Check the length of the columns M
    Data should be parallelized by Spark (Master)
        N rows of data were collected
        do in parallel
    Set the initial synaptic weights $w_{ij}$ and thresholds j to small random values, such as [0, 1], and
    then repeat the procedure. Assign tiny positive values to the learning rate parameter as well as
    the forgetting factor.
```

$$\mathbf{W} = \begin{bmatrix} w_{11} & \cdots & w_{1j} & \cdots & w_{1n} \\ \vdots & & \vdots & & \vdots \\ w_{i1} & \cdots & w_{ij} & \cdots & w_{in} \\ \vdots & & \vdots & & \vdots \\ w_{n1} & \cdots & w_{nj} & \cdots & w_{nn} \end{bmatrix} \text{// e.g., } \mathbf{W} = \mathbf{I} = \begin{bmatrix} 1 & \cdots & 0 & \cdots & 0 \\ \vdots & & \vdots & & \vdots \\ 0 & \cdots & 1 & \cdots & 0 \\ \vdots & & \vdots & & \vdots \\ 0 & \cdots & 0 & \cdots & 1 \end{bmatrix}, \boldsymbol{\theta} = \begin{bmatrix} 0.4940 \\ 0.2661 \\ 0.0907 \\ 0.9478 \\ 0.0737 \end{bmatrix}$$

```
    Calculate the output of the neuron at iteration T.
    Update the weights in the network: $w_{ij}(p + 1) = w_{ij}(p) + \Delta w_{ij}(p)$, // i, j = 1, 2, ..., n
    Send reduction array T
    Close connection
    End
```

Figure 3. Overall proposed algorithm for data reduction

4. MATERIAL AND METHOD

A number of supervised classification approaches were considered in this study, including Nave Bayes, K-Nearest Neighbours, NN, and Random Forest, among others. To begin, Table 1 has a description of the system, while Table 2 has a description of the six datasets that were used in the study. The computation times for parallel GHA and parallel PCA on the identical hardware arrangement were used to present the results. With respect to the six large datasets that were used for the analysis in this study, the performance of Apache Spark and MLlib 2.0 was compared. The six datasets used in this study were obtained from the UCI ML repository. The experiments in this research are comprised of a Spark cluster that runs on Apache Zeppelin 0.7.1 and an HDFS, which is described in detail in the paper. The Spark cluster is made up of four nodes: a master node that executes the driver application, three worker nodes, and a cluster manager. The three nodes were configured in a manner that was similar to that shown in Figure 1. The three worker nodes were each given a memory allocation of 48 GB and were configured with four executors (each with a memory allocation of 4 GB) and two cores. Each worker was allotted three executors (each with a memory size of five gigabytes) while the master node was allotted two cores. A total of 16 GB of RAM has been assigned to the driver process. Scala 2.11.8 was used as the programming language for MLlib execution in the Spark 2.2.1 cluster, with Hadoop 2.7.3 serving as the distributed storage system. The amount of memory available to the executors in each worker node was changed in order to get the best possible performance.

Table 1. Description of the system

Operating system	Windows10
CPU	Intel® CoreTM i7-6700 processor running at 3.40 GHz with eight cores
Memory	16 GB
No. of workers	3
Computational framework	Apache Spark 2.2.1
Compatible framework	Radoop
DSS	HDFS (Hadoop 2.7.3)
Code development editor	Apache Zeppelin 0.7.1
Coding language	Scala 2.11.8

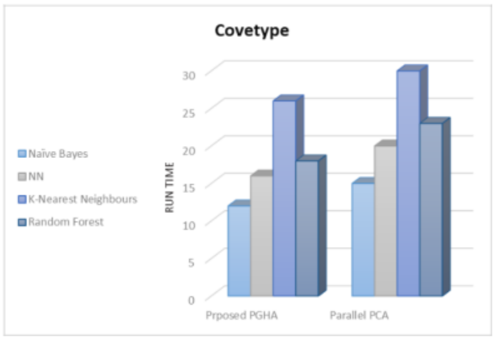
Table 2. Datasets description

Data	No of record	No of attributes	No of classes
Covtype	581012	54	7
Covtype-2	581012	54	2
Higgs	11,000,000	28	2
Botnet Attacks	7,062,606	115	10
Dota2	102944	116	2
SUSY	5,000,000	18	2

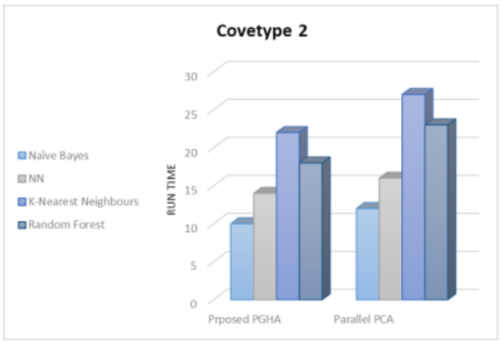
5. RESULTS AND DISCUSSION

The initial step of the classical GHA is loading the whole dataset to memory. Note that the data size must be within the limit that can fit within the memory size of the computer. Memoryup is a performance metric that assesses a parallel clustering algorithm's ability to efficiently utilize the available memory space on each node. It is possible to compute the memoryup by changing the memory size of each node while keeping the dataset and the number of nodes the same. The concept of the new GHA approach is to use the idea of data scanning by rows. The GHA approach can still implemented even when the data exceeds the computer memory size. A significant amount of CPU time is frequently lost in large datasets as a result of the unnecessary processing of redundant and non-representative data. The deletion of this type of data can frequently result in a significant increase in processing performance. A further benefit of eliminating nonrepresentative data from huge datasets is that storage and transmission of these datasets become less difficult. The computational advantages of the proposed new system were evaluated using numerical examples. The computation was performed on a third generation Intel core-i7 2.8GHz processor with 16GB DDR3 memory. The programming language for all the algorithms in the proposed big data GHA was C++. Through a thorough examination of the PGHA's running time utilizing the Radoop method and Parallel PCA, the comparison seeks to assess the speed performance of the algorithm. For the purposes of this example, we will suppose that the degree of support varies while the number of computer nodes remains at 3. Runtime with different support degrees for the datasets Covtype, Covtype-2, Higgs, Botnet Attacks, Dota2, and SUSY are depicted in Figures 4 (a), (b), (c), (d), (e), and (f).

The algorithms is shown by the x-axis, while the running time is represented by the y-axis. The two techniques appear to be more efficient when the support degree is increased, as can be seen in the graph. Remember that our approach appears to be faster than parallel PCA when running on all datasets, which is a significant advantage. The performance of the proposed based was evaluated based on a single processor because if a parallel algorithm is used, the performance may be over-shadowed by the performance of the other algorithms. Execution times and speed-up ratios are depicted in Figure 4 in relation to the number of objects in the datasets for different numbers of processors.

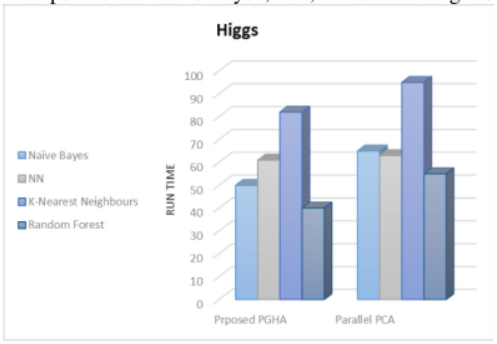


(a)

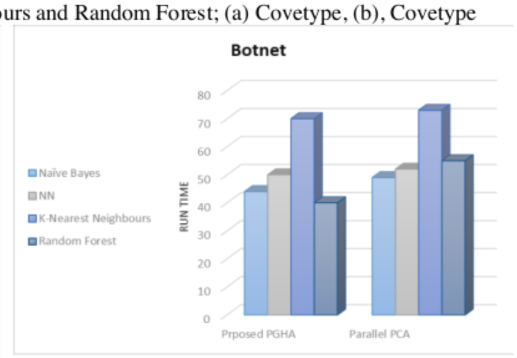


(b)

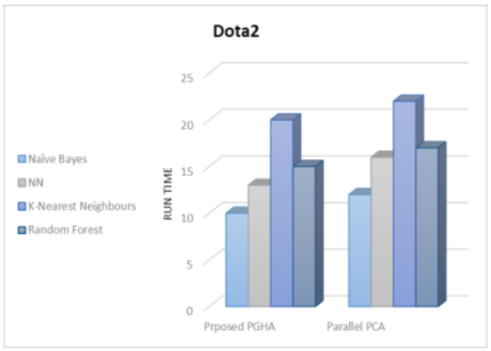
Figure 4. Running time of PGHA using Radoop compared to parallel PCA under different classification experiments. Naïve Bayes, NN, K-Nearest Neighbours and Random Forest; (a) Covetype, (b), Covetype



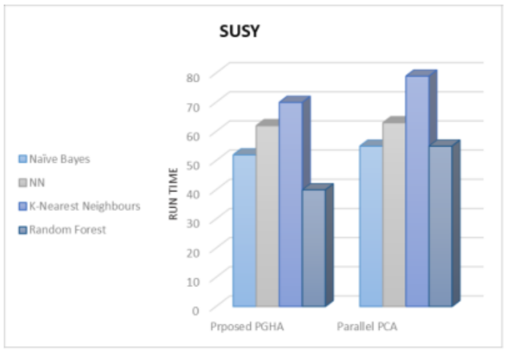
(c)



(d)



(e)



(f)

Figure 4. Running time of PGHA using Radoop compared to parallel PCA under different classification experiments. Naïve Bayes, NN, K-Nearest Neighbours and Random Forest; (c) Higgs, (d) Botnet, (e) Dota2, (f) SUSY datasets *(continue)*

The analysis of the dataset Covetype is only possible when the number of processors in our computer cluster system is equal to or greater than eight. When dealing with large amounts of data, the advantages of a distributed memory system are readily apparent. Based on the experimental results, it has been demonstrated that this parallel reduction method has superior speed-up and linear scaling behavior (time complexity), and that it may be used to overcome space complexity limits by using the aggregate memory of the reduction system. The performance evaluation of the new approach was based on the method of inducing the base classifier. The results of the experiments showed that the PGHA, as a data reduction tool, minimized the run time compared to parallel PCA algorithm as shown in Figure 4. However, our partial reduction method outperformed full reduction methods in many real-world data analysis and data reduction applications.

6. CONCLUSION

The concept of parallel computing and parallel dimensionality reduction algorithms was introduced in this study. This article proposed the parallel algorithm concept based on the classical DR algorithm for effective handling of the issue encountered in big data mining. The proposed framework in this work was based on the previous studies with the aim of reducing the high volume of input data features while retaining the relevant information. To achieve this aim, both GHA and the proposed parallel algorithm were used to improve the DR and reduce features complexity. The evaluation results showed that GHA was better in reducing redundant features of datasets. In the future studies, effort will be focused on combination of the proposed parallel GHA in this work with other ML methods, as well as improving the performance of the latest datasets using some evolutionary optimization techniques.

ACKNOWLEDGEMENTS

The authors would like to thank ICCI, Informatics Institute for Postgraduate Studies, Iraqia University and Al Salam University College for their facilities, support and cooperation during this research; and Universitas Ahmad Dahlan to support this collaborative research. Special thanks to the anonymous reviewers for their valuable suggestions and constructive comments.

3135 An effective classification approach for big data with parallel generalized Hebbian algorithm

ORIGINALITY REPORT

8%
SIMILARITY INDEX

PRIMARY SOURCES

1	Ahmed Hussein Ali, Mahmood Zaki Abdullah. "A Parallel Grid Optimization of SVM Hyperparameter for Big Data Classification using Spark Radoop", Karbala International Journal of Modern Science, 2020 Crossref	91 words — 3%
2	www.stat.purdue.edu Internet	41 words — 1%
3	Mo Hai, Yuejing Zhang, Haifeng Li. "A Performance Comparison of Big Data Processing Platform Based on Parallel Clustering Algorithms", Procedia Computer Science, 2018 Crossref	28 words — 1%
4	s3.amazonaws.com Internet	20 words — 1%
5	www.deepdyve.com Internet	18 words — 1%
6	Royida A. Ibrahem Alhayali, Mohammad Aljanabi, Ahmed Hussein Ali, Mostafa Abdulghfoor Mohammed, Tole Sutikno. "Optimized machine learning algorithm for intrusion detection", Indonesian Journal of Electrical Engineering and Computer Science, 2021 Crossref	16 words — < 1%

7	docplayer.net Internet	14 words — < 1%
8	Leandro P. F. Rodriguez, Marco V. Cedeño, Mabel C. Sánchez. "Sensor Location for Enhancing Fault Diagnosis", Industrial & Engineering Chemistry Research, 2016 Crossref	12 words — < 1%
9	Sheikh Kamaruddin, Vadlamani Ravi. "Big data regression via parallelized radial basis function neural network in Apache Spark", Institution of Engineering and Technology (IET), 2021 Crossref	9 words — < 1%
10	Dong Qiu, Bing Yang. "Text summarization based on multi-head self-attention mechanism and pointer network", Complex & Intelligent Systems, 2021 Crossref	8 words — < 1%
11	Yasunori Endo. "On principal component analysis for data with tolerance", 2011 IEEE International Conference on Granular Computing, 11/2011 Crossref	8 words — < 1%
12	link.springer.com Internet	8 words — < 1%
13	Liang-Hwa Chen, Shyang Chang. "An adaptive learning algorithm for principal component analysis", IEEE Transactions on Neural Networks, 1995 Crossref	7 words — < 1%

EXCLUDE QUOTES	OFF	EXCLUDE MATCHES	OFF
EXCLUDE BIBLIOGRAPHY	OFF		