A Distributed Dendritic Cell Algorithm for Big Data

Zaineb Chelly Dagdia
Department of Computer Science, Aberystwyth University, Aberystwyth, United Kingdom
LARODEC, Institut Supérieur de Gestion de Tunis, Tunis, Tunisia,
chelly.zaineb@gmail.com,zaineb.chelly@aber.ac.uk

ABSTRACT

In this work, we focus on the Dendritic Cell Algorithm (DCA), a bio-inspired classifier, and its limitation when coping with very large datasets. To overcome this limitation, we propose a novel distributed DCA version for data classification based on the MapReduce framework to distribute the functioning of this algorithm through a cluster of computing elements. Our experimental results show that our proposed distributed solution is suitable to enhance the performance of the DCA enabling the algorithm to be applied over big data classification problems.

CCS CONCEPTS

• **Computing methodologies → Artificial intelligence**;

KEYWORDS

Dendritic Cell Algorithm, Classification, Big Data, Distributed Processing, Scalability.

ACM Reference Format:
Zaineb Chelly Dagdia. 2018. A Distributed Dendritic Cell Algorithm for Big Data. In *GECCO '18 Companion: Genetic and Evolutionary Computation Conference Companion, July 15–19, 2018, Kyoto, Japan.* ACM, New York, NY, USA, 2 pages. https://doi.org/10.1145/3205651.3205701

1 INTRODUCTION

In this work, the main objective is to further enable bio-inspired algorithms to be applied on big data. In this concern, we focus on the Dendritic Cell Algorithm (DCA) [2] which is derived from behavioral models of the natural Dendritic Cells (DCs). The DCA has caught the attention of many researchers due to its worthy characteristics as it exhibits numerous advantageous features for classification problems [2]. Despite the emergence of the DCA, in the current literature, the practical application of the algorithm was limited to problems with moderated size only. The reason behind this arise from the necessity to use an antigen multiplier to generate at once several copies of antigens, referring to data instances, to process them in turn to finally perform the classification task. More precisely, the DCA requires multiple instances of identical antigens, so processing across a population can be performed in order to generate the classification results for each antigen. However, as the number of data instances is increasing this task becomes challenging and this is where the DCA inadequacy arises. It is quite unmanageable to generate the set of all antigen copies based on the huge number of data instances due to hardware and memory constraints. In this work, we propose an efficient distributed dendritic cell algorithm for large-scale datasets which solves the standard DCA mentioned computational inefficiencies. Our novel DCA version is characterized by its distributed implementation design based on both Scala and the MapReduce Apache Spark framework [3].

GECCO '18 Companion, July 15–19, 2018, Kyoto, Japan

ACM ISBN 978-1-4503-5764-7/18/07.
https://doi.org/10.1145/3205651.3205701

2 THE DISTRIBUTED DENDRITIC CELL ALGORITHM

To deal with high dimensional datasets it appears mandatory to store all the data in a distributed environment and ensure computations in a parallel way. With respect to this, we first partition the entire DCA algorithmic processes into elementary tasks, each executed independently, and then conquer the intermediate results to finally acquire the ultimate output; the classes of the antigens.

For antigen classification, Sp-DCA has to go through its distributed phases run on the original high dimensional input database which corresponds to the data stored in the given DFS as a single file. To operate on the given DFS in a parallel way, a Resilient Distributed Dataset (RDD) is created. We may formalize the latter as a training set, of a determined size N, which corresponds to the antigen dataset defined as T_{RDD}, where universe $U = \{x_1, \ldots, x_N\}$ is the set of antigen identifiers, the attribute set $C = \{c_1, \ldots, c_V\}$ contains every single feature of the T_{RDD} and the decision attribute D corresponds to the class label of each T_{RDD} sample. As Sp-DCA is based on the standard DCA concepts, and since DCA is applied to binary classification problems; then the decision attribute, D, of our Sp-DCA is defined as: $D = \{d_{normal}, d_{anomalous}\}$. The universe set U presents the pool from where the antigens will be multiplied by an antigen multiplier m generating a pool of antigens $AntigensPool = \{an_{1_i}, \ldots, an_{1_m}, \ldots, an_{N_i}, \ldots, an_{N_m}\}$.

In order to make our algorithm scalable with the high number of both training data and antigens and within the Apache Spark perspective, Sp-DCA partitions the given T_{RDD} into p data blocks based on splits from the universe set U. Indeed, Sp-DCA creates an RDD from the generated antigens pool, $AntigensPool_{RDD}$, and splits it into a a number of disjoint subsets. Both of these RDDs are accessible from any computer of the cluster independently of their size. In such a way, $T_{RDD} = \bigcup_{i=1}^{p} \bigcup_{j=1}^{N} (x_j) T_{RDD_{(i)}}$ and $AntigensPool_{RDD} = \bigcup_{i=1}^{a} \bigcup_{x,y=1}^{m,N} (an_{x,y}) AntigensPool_{RDD_{(i)}}$. To ensure scalability, rather than applying Sp-DCA to T_{RDD} including all antigens and to $AntigensPool_{RDD}$ including all the copies of antigens, the distributed algorithm will be applied to every single $T_{RDD_{(i)}}$ and $AntigensPool_{RDD_{(i)}}$ that at the end all the intermediate results will be gathered from the different p and a partitions. In such a way, we can guarantee that Sp-DCA can deal with the large number of the antigens and hence solving the DCA limitations.

3 EXPERIMENTAL SETUP

To demonstrate the effectiveness of our Sp-DCA we chose the Supersymmetry Particles (SUSY) dataset [1]. The data includes 5 million data items referring to the simulated collision events described through 19 features. The first feature refers to the class (1 for signal, 0 for background) followed by 8 features which are low-level features then 10 features which are high-level features. Aiming to investigate the scalability of our Sp-DCA, we have created 4 synthetic different versions of the SUSY dataset by generating 10, 20, 30 and 40 million of instances of the original dataset. We will denote these versions as SUSY10M, SUSY20M, SUSY30M and SUSY40M. The databases are named according to the number of antigens contained, i.e. SUSY10M contains 10 million data items and SUSY5M contains 5 million items.

Our experiments are performed on the High Performance Computing Wales (HPC Wales) which provides a distributed computing facility. Under this testbed, we used dual 12 core Intel Westmere Xeon X5650 2.67 GHz CPUs and 36GB of memory to test the performance of Sp-DCA which is implemented in Scala 2.11 within Spark 2.1.1. The Sp-DCA parameter setting is as follows: In the initialization phase, the class and the 10 high-level features are selected among the 19 features. The first and the second high-level features are used to be mapped as a PAMP and SS, respectively, while the rest of the features are used to represent the DS. The weights used in the context assessment phase are 2, 0, 2, 2, 2, 2, 1, 0.9 and -0.9 for $W_{PAMP,CSM}$, $W_{PAMP,smDC}$, $W_{PAMP,mDC}$, $W_{SS,CSM}$, $W_{SS,smDC}$, $W_{SS,mDC}$, $W_{DS,CSM}$, $W_{DS,smDC}$ and $W_{DS,mDC}$, respectively. The DC migration threshold is set to 10, each data item is mapped as an antigen with the value of the antigen equals to the data ID of the item and an antigen multiplier $m = 9$ is used to derived the $AntigenPool_{RDD}$ resulting in 45, 90, 180, 270 and 360 million antigens for the SUSY5M, SUSY10M, SUSY20M, SUSY30M and SUSY40M datasets, respectively.

4 RESULTS AND ANALYSIS

4.1 Analysis of the Speed-Up

Considering the speed-up of Sp-DCA, we keep the size of the dataset constant and increase the number of nodes [4]. We plot the average speed-ups needed to run a single iteration within the Sp-DCA (over the 10 iterations) and their corresponding average times. From our experiments, we see that Sp-DCA has a good speed-up performance. The more the size of the database increases, the more the speed-up becomes closer to linear. This performance is almost the same for SUSY20M, SUSY30M and SUSY40M databases. However, we notice that the SUSY10M database has a slightly lower speed-up curve. This is explained by the fact that based on the size of SUSY10M the partitioning of the data is concentrated on the total number of nodes used resulting in a big communication cost. Therefore, the skew in this case is higher than in the other datasets and the total speed-up is lower. Nevertheless, once the size increases starting from 20 million of instances we clearly observe a difference in the algorithm speed-up performance. This observation is also supported by the execution times which decreases with increasing the number of nodes while for the SUSY10M we observe hardly any improvement when it comes to more than 4 nodes.

4.2 Analysis of the Size-Up

Sizeup analysis holds the number of nodes constant, and grows the size of the databases by the factor m. The size-up measures how much longer it takes when the database size is m-times larger than the original database [4]. To measure the performance of size-up, we have fixed the number of nodes to 1, 2, 3, 4, 8, 12 and 16 respectively. Our method has a very good size-up performance as it is able to process large datasets efficiently while keeping the number of nodes constant and increasing the size of the data. We can clearly see that a 2 times larger problem for instance (SUSY40M) needs about 2 times more time (SUSY20M).

4.3 Analysis of the Scale-Up

Scale-up is defined as the ability of an m-times larger system to perform an m-times larger job in the same run-time as the original system [4]. To demonstrate how well Sp-DCA handles larger datasets when more nodes are available, we measure the scale-up where we increase the size of the databases in direct proportion to the number of nodes. For instance, 10 million antigens are classified on 1 node and 40 million antigens are classified on 4 nodes. Our experiments show that Sp-DCA scales very well as the scale-up values are all close to 1.

5 CONCLUSION

In this paper, we have developed a distributed bio-inspired DCA for large-scale data classification under the Spark framework. The experimental study has shown that our Sp-DCA is an efficient distributed and scalable bio-inspired classification technique. The algorithm is applied to large databases of 40 million of instances and 360 million of antigens and it guarantees satisfactory classification results.

ACKNOWLEDGMENTS

This work is part of a project that has received funding from the European Union's Horizon 2020 research and innovation programme under the Marie Skłodowska-Curie grant agreement No 702527. The author would additionally like to thank the support of the Supercomputing Wales project, which is part-funded by the European Regional Development Fund (ERDF) via Welsh Government.

REFERENCES

[1] http://mlearn.ics.uci.edu/MLRepository.html , UCI Machine Learning Repository. [n. d.].
[2] Zeineb Chelly and Zied Elouedi. 2016. A survey of the dendritic cell algorithm. *Knowledge and Information Systems* 48, 3 (2016), 505–535.
[3] James G Shanahan and Laing Dai. 2015. Large scale distributed data science using apache spark. In *Proceedings of the 21th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. ACM, 2323–2324.
[4] Xiaowei Xu, Jochen Jäger, and Hans-Peter Kriegel. 1999. A fast parallel clustering algorithm for large spatial databases. In *High Performance Data Mining*. Springer, 263–290.