A distributed learning algorithm for Self-Organizing Maps intended for outlier analysis in the GAIA – ESA mission

D. Garabato[1] **C. Dafonte**[1] **M. Manteiga**[1] **D. Fustes**[1] **M. A. Álvarez**[1] **B. Arcay**[1]

[1]Fac. Informática, Universidade da Coruña, Campus de Elviña, 15071 A Coruña, Spain

Abstract

Since its launch in December 2013, the Gaia space mission has collected and continues to collect tremendous amounts of information concerning the objects that populate our Galaxy and beyond. The international Gaia Data and Analysis Consortium (DPAC) is in charge of developing computer algorithms that extract and process astrophysical information from these objects. It organizes its work by means of work packages; one of these packages, Outlier Analysis, is dedicated to the exploration of vast amounts of outlier objects detected during the main classification of the observations. We present a method that is based on Self-Organizing Maps (SOM) and parallelized by means of the Hadoop framework so as to improve its performance. We also compare the execution times of both the sequential and the distributed versions of the algorithm.

Keywords: Gaia mission, Self-Organizing Maps, distributed computing, Hadoop

1. Introduction

Gaia is the most ambitious mission carried out by the European Space Agency (ESA) and it is expected to become a milestone in astronomy. The main purpose of this survey is to provide a six-dimensional map (positions and velocities) of our Galaxy, the Milky Way, unveiling its composition, formation, and evolution [1]. Furthermore, the astrometric accuracy will be unprecedented, up to the level of micro-arcseconds, well beyond ESA's previous astrometric satellite Hipparcos. Gaia will not only observe stars, but also all objects brighter than G = 20 mag (G magnitudes represent Gaia photometric instrument integrated white light in the 330–1050 nm), including many solar-system objects, such as asteroids and minor planets, external galaxies, quasars, supernovae, multiple stars, etc.

The spacecraft, launched in December 2013, is expected to provide approximately 10^{12} observations, related to every object in the sky, over 80 epochs on average during its 5 years of operating time. In order to derive the main astrophysical properties of the astronomical objects, this observations will be processed by a software pipeline at CNES (Centre National d'Etudes Spatiales, France), which is being produced by an international consortium, the Gaia Data Processing and Analysis Consortium (DPAC). DPAC gathers an international collaboration with memberships from all over Europe, which nowadays includes a community of over 500 scientists and software engineers coming from more than 20 countries. This organization consists of several coordination units (CUs) and it is in charge of a well-defined set of tasks in the Gaia data processing effort. In particular, CU8 is devoted to both the classification and the derivation of the main astrophysical parameters of point-like sources [2]. This unit is subdivided into several work packages: DSC (Discrete Source Classifier) is the main package for classification, whereas GSP-Phot (General Stellar Parameterizer - Photometry) and GSP-Spec (General Stellar Parameterizer - Spectroscopy) are the main parameterization packages. There are some other packages that are responsible for more specific tasks, such as Quasar/Galaxy parameterization (QSOC and UGC, respectively) or specific stellar population parameterizers (ESP). Finally, there are two packages dedicated to the unsupervised analysis of the data, OCA (Object Cluster Analysis) and OA (Outlier Analysis). OA is the package on which the present work focuses, aimed at analyzing the observations classified as outliers by DSC and OCA.

CU8 has established a set of well-defined general astronomical classes: STAR, BINARY STAR, NON-PHYSICAL BINARY, GALAXY, and QUASAR, while the specific classes UNKNOWN and UNDEFINED are used to label non-positive results of classification. Such labels are used by DSC and will be used in a near future by OCA. Although the physics of the stars is well understood and there are extensive catalogs, which contain information about the light distribution of most astronomical objects, Gaia is expected to observe such an enormous quantity of sources that many of them could differ significantly from model predictions or previous observations. Gaia will observe a significant sample of peculiar objects, such as supernovae, Wolf-Rayetstars, multiple systems, or high redshift quasars, as well as new kinds of objects unseen before. In addition, low signal-to-noise ratios, cosmic rays, instrument artifacts, and other damaged data will eventually occur, leading to classification errors. Thus, it is estimated that approximately 10^8 objects (10% of the total) will be labeled as UNKNOWN by DSC, which uses an automatic outlier detector based on a one-class SVM that rejects objects that are far from the training data space. Moreover, some objects will receive a set of probabilities that is not decisive in terms of final classification and therefore they will be labeled as UNDEFINED. The OA module will have to deal with both sets of objects (UNKNOWN and UNDEFINED), trying to make an accurate classification which can help scientists to understand and unveil their nature.

2. Gaia simulated datasets

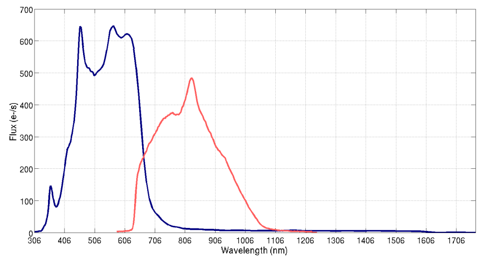
Before Gaia observations are available for processing, DPAC is providing simulated datasets, obtained from both models and earth-based observations, so that the software packages can be tested properly. A powerful simulator was developed to facilitate this process, the Gaia Object Generator (GOG, [3]), which has generated many datasets to test CU8 algorithms. Moreover, some already existent catalogs were transformed by GOG to BP/RP low resolution format and instrumental characteristics, such as the SDSS catalog. However, most of these datasets and catalogs include observations whose classification and parameterization are well known, since they concern typical or classical object categories. Although these types of datasets could be used to validate the OA algorithm presented in Section 3, this algorithm is devoted to the analysis of objects whose nature is unclear. Hence, we decide to use, in this work, a set of objects from the SDSS catalog that were classified as UNKNOWN. Such dataset is composed by 10.125 objects, mostly faint objects, incomplete spectra, and unsuccessful observations. Nevertheless, the size of this dataset is not large enough to measure performance and scalability while using a distributed computing technique (Section 4). To this purpose, we decided to increase the number of observations by putting them together several times, so that we can make further measurements.

2.1. Data preprocessing

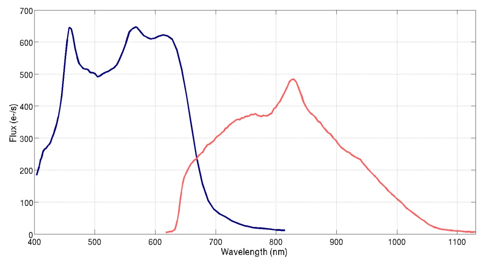
Before processing the data presented above by the OA module described in Section 3, it was necessary to apply some preprocessing to the BP/RP spectrophotometric data. Firstly, we decide to remove the pixels that lie in the extremes of BP and RP spectra, where the sensitivity of the devices decays exponentially, corresponding to the low signal-to-noise ratio (SNR) pixels. Afterwards, the total spectrum area is scaled to a determined quantity, so that objects with different brightness levels can be compared by means of their spectra using common metrics, such as Euclidean distance. This scaling is done by dividing each spectrum pixel by its integral value, following Equation 1.

$$F_i = 1000 \frac{F_i}{\sum_{i \in S}(F_i)} \qquad (1)$$

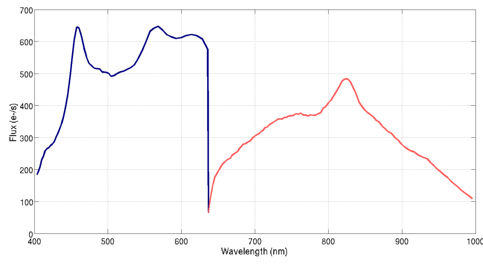
where F_i is the flux of a spectrum in band i and S represents the bands of the spectrum. This method sets the target area at 1000 units. Note that negative values can cause a failure in the previous calculation, so they should be truncated to 0 before scaling. Finally, the overlapping region of BP/RP is removed, avoiding redundant wavelengths, while maintaining the red part of the spectrum, since it is more sensitive in that region [4]. Figure 1 shows these operations.



(a) Original BP/RP spectrum.



(b) Spectrum after low SNR pixels extraction.



(c) Spectrum after overlapping area removal.

Figure 1: BP/RP spectrum preprocessing.

3. Outlier Analysis module

Several studies have investigated approaches for the analysis of big datasets composed of multidimensional outliers [5, 6]. Outliers are, by definition, objects that can not be fitted into the existing models, so the analysis of large outlier datasets must be done by means of unsupervised algorithms. Moreover, the Gaia project presents some additional challenges: a) the size of the expected outlier sample; and b) the variability of properties among individual observations. These challenges will increase the difficulty of discovering interesting objects, classes of objects, or data structures. To work this out, we would like to use a method that satisfies the following conditions: a) it can simplify the original dataset, projecting it into a smaller set of features, thus allowing for manual inspection and detailed analysis; and b) it can discover connections between the features, being able to throw light on their nature.

In the Data Mining field, there are two main approaches to deal with multidimensional data based on unsupervised techniques: Dimensionality Reduction and Clustering. Dimensionality Reduction tries to reduce the number of dimensions (variables, attributes) in the dataset to a level where they can be more reasonably analyzed by domain experts. Principal Component Analysis (PCA) [7] is the most popular algorithm of this type. On the other hand, Clustering is aimed at grouping the data into a number of clusters that share similar properties. A wide variety of clustering algorithms has been proposed, as it is an ill-defined problem [8, 9, 10].

To this effect, we propose an analysis algorithm based on two stages: clustering and labeling. Firstly, during the clustering phase, we try to group the sources into several sets, as homogeneous as possible, in order to facilitate the analysis of the resulting dataset. Then, the labeling stage takes place, trying to describe the groups generated previously by all available means, including both internal data, such as Gaia photometry or astrometry, and external data, such as other astronomical surveys, additional ground observations, or human expert knowledge. This algorithm will have to process an enormous amount of data and it must be run in a limited amount of time and resources, since a complete dataset will be produced by DPAC every 6 months, which is the time that the CUs have to perform their joint processing.

Our choice for analyzing outlier is based on Self-Organizing Maps (SOM) [11], which have been used extensively in a number of specific fields. However, they have been used sparingly thus far in Astronomy [12, 13, 14, 15, 16]. SOM are a type of Artificial Neural Networks (ANN) that project the input data space into a fixed set of clusters (called neurons in the ANN field) usually arranged in a 2D (or 3D) grid specified by a matrix with N rows by M columns. Each cluster has a representative, called prototype (neuron weights in the ANN field), which is a virtual pattern that better represents or resembles the set of input patterns belonging to this cluster. To achieve this solution, an unsupervised learning or training procedure, based on a competition between the clusters to fit the data as best as possible, is carried out. Such a procedure is described in Section 3.1.

3.1. SOM training algorithm

During the learning stage, the algorithm tries to find the best prototypes (weights) for the SOM clusters. Since it is a NP-hard problem, an iterative optimization procedure is followed to reach an acceptable solution from a randomly initialized set of neuron weights. There are several learning algorithms that can be used to train the SOM: online, batch, fuzzy, etc. We have tried many of them, obtaining the best results for both the online and the batch algorithms. Nevertheless, the batch algorithm was found to be more stable than the online algorithm, as its results do not depend on the order in which the input patterns are presented to it [17]. Moreover, the online learning algorithm can not be parallelized because the neuron weights have to be updated for each input item; however, we can parallelize the batch learning algorithm, achieving a scalable solution. Therefore, the integration of this algorithm in an industrial environment is more feasible, because it is deterministic, reproducible, and scalable, so it was decided to use this algorithm at the Gaia software pipeline.

Before presenting this algorithm we need to define some concepts:

The neighbourhood When dealing with ANN, the neighbourhood determines how many neurons are updated during the learning phase and it is obtained by a function, called the neighbourhood function. This neighbourhood is usually large during the first iteration, but shrinks as the iterations succeed themselves (typically, a Gaussian function).

Similarity distance This similarity distance is used to determine which neuron fits best into an input pattern, according to its prototype. Usually, the squared euclidean distance is used.

Convergence criteria This criteria determines whether the training stage must continue or not. If the weights of the neurons have not changed in the current iteration or the maximum number of iterations is reached, then the procedure is finished.

The batch algorithm consists of the following steps, which can be observed in Figure 2:

1. Set up properly the configuration parameters: the topology of the output grid, the neighbourhood function, the similarity distance, etc.
2. Initialize, randomly, the neurons and its weights.

3. If it is necessary, then update the neighbourhood function, according to the current iteration.

4. For each pattern in the input space, calculate the winner neuron, depending on the specified similarity distance.

5. For each neuron, update its weights, according to the observations assigned to it, the observations assigned to its neighbours, and the neighbourhood function.

6. Check if the algorithm has converged, then the learning phase is over. Otherwise, go back to step 3.

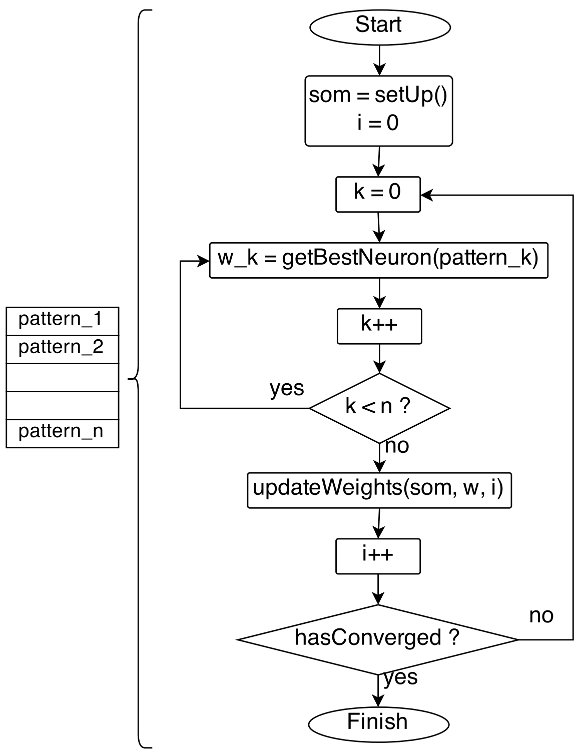


Figure 2: SOM batch learning algorithm.

4. Distributed computing for OA

The large amount of data that the OA module will have to process makes the OA module unfeasible to be run on one single machine, due to exceedingly long execution times and physical memory requirements. To this effect, we decided to start thinking about improving OA so that it can be run on a set of machines or cluster, so as it will be done once OA is integrated in the Gaia software pipeline, at CNES. According to DPAC software developments, which are carried out using Java, we decided to use Hadoop because it offers a solution for both distributed storage (HDFS) and distributed processing (Map-Reduce) [18]. HDFS is the file system used by Hadoop to distribute input files, after being splitted into many blocks of data (chunks), among the nodes included in the cluster. Hadoop Map-Reduce is a programming model that allows the user to distribute data processing through a cluster. Mainly, it consists of two tasks: a) the Map, which takes several key/value pairs from the input file(s), processes each, and outputs zero or more key/value pairs; and b) the Reduce, which receives the output given by the map procedure, sorted and grouped by key (shuffle), and processes each, generating zero or more outputs.

These key/value pairs are just two Java special objects (the types of which may be chosen by the programmer) that can be written to a file by Hadoop framework, so they can be used for communication between the different tasks as well as general input/output. The programmer only needs to specify the behaviour of two functions, the map function and the reduce function, which are invoked by the Map and the Reduce tasks. Commonly, the Map task is invoked by Hadoop using each one of the items contained in the input file (the value component) and the line number of the input file related to that item (the key component). Figure 3 illustrates this idea. The number of Map and Reduce tasks, so as many other parameters, can (and must) be configured properly in order to obtain the best possible performance.

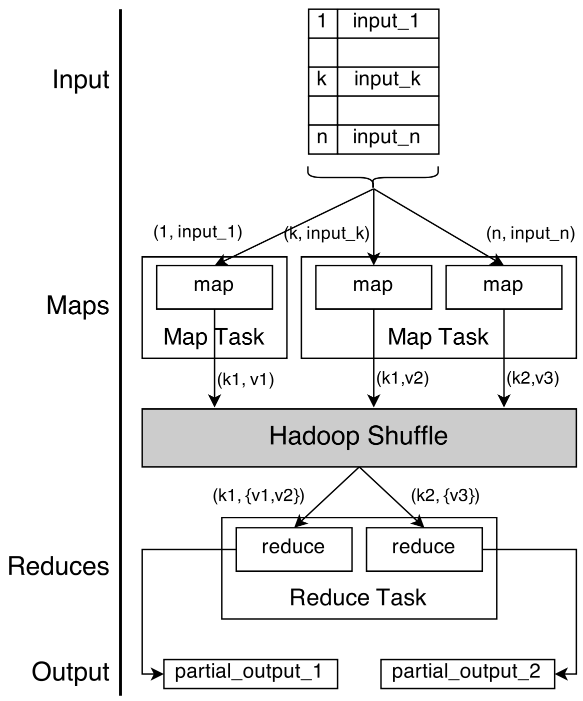


Figure 3: Hadoop Map-Reduce job.

4.1. Hadoop-based SOM training algorithm

In this section, we propose a Hadoop-based implementation for the SOM learning algorithm described in Section 3.1. Since it is an iterative algorithm and the result produced by one iteration depends on the previous one, we can not run multiple iterations simultaneously. So, we decided to parallelize the content of each iteration.

Firstly, we store all the spectra on a single file (usually a Hadoop sequence file) which is distributed through all the machines in the cluster via HDFS.

Then, we define both the Map and the Reduce tasks in the following manner:

Map This procedure receives a spectrum (the value) and the line number of the input file associated with it (the key), calculates the winner neuron for the given spectrum, and outputs a key/value pair which consists of the winner neuron (the key) and the input spectrum (the value).

Reduce This task receives a neuron (the key) and a collection of spectra (the value), whose winner neuron has been determined to be the neuron specified by the key. Partial updates for the weights are worked out and produced as output.

Finally, we have to use a main application or driver where we can launch and configure the training process. Moreover, it invokes the Hadoop job where Map and Reduce tasks take place and commits the weight update to the network by gathering the partial results from the Reduce tasks.

5. Scenario

The next section compares the performance of the sequential algorithm and the Hadoop-based algorithm. As such we are describing the scenario that was used to carry out some experiments.

5.1. Single machine description

To execute the sequential algorithm we used a machine whose details are shown in Table 1. However, certain issues related to memory requirements had to be solved, since we could not load the full dataset into memory. To work this out, we decided to implement a custom collection object which loaded partial sets of data in advance by using a background thread to fetch it.

CPU	*Intel Core i7 950 @ 3.07GHz × 8*
Mem.	*11.8GB DDR2 (2GB × 6)*
OS	*Debian 8 Jessie - 3.16.0-4-amd64*

Table 1: Single machine details.

5.2. Cluster description

To execute the Hadoop-based algorithm we used a cluster that consists of 6 machines running under *Debian 7.7 Wheezy - Linux 3.2.0-4-amd64* Operating System and 64-bit compiled *Hadoop 2.5.2* whose particular details can be found in Table 2.

2×	**CPUs**	*Intel Xeon CPU E5420 @ 2.50GHz × 4*
		Intel Xeon CPU E5420 @ 2.50GHz × 4
	Mem.	*24GB DDR2 @ 667MHz (4GB×6)*
2×	**CPUs**	*Intel Xeon CPU E5472 @ 3.00GHz × 4*
		Intel Xeon CPU E5472 @ 3.00GHz × 4
	Mem.	*32GB DDR2 @ 667MHz (4GB×8)*
1×	**CPUs**	*Intel Xeon CPU X5550 @ 2.67GHz × 4*[1]
		Intel Xeon CPU X5550 @ 2.67GHz × 4[1]
	Mem.	*24GB DDR3 @ 1333MHz (4GB×6)*
1×	**CPUs**	*Intel Xeon CPU X5660 @ 2.80GHz × 6*[2]
		Intel Xeon CPU X5660 @ 2.80GHz × 6[2]
	Mem.	*32GB DDR3 @ 1333MHz (8GB×4)*

Table 2: Cluster details.

Hadoop has been also parameterized, so that an acceptable performance was achieved. Hence, we decided to use a block size of 64Mb, which determines the number of splits of the input file and therefore the number of Map tasks launched. Finally, the number of Reduce tasks was set up to 4.

5.3. SOM configuration

To perform the experiments we decided to use the following configuration for the SOM: firstly, both the topology and the number of clusters to use was determined empirically by means of several processes of train/analysis, resulting in a 30×30 square map, which is a trade-off between retrieval error and data compression. On the other hand, it was decided to run the algorithm up to 200 iterations, since it was found that the SOM usually converges around the 200th iteration. In the same way, we also need to choose a suitable neighbourhood function, so we have tested some of them (Bubble, Gaussian, Epanechicov), getting better results for the Gaussian one, whose width shrinks for the 200 learning iterations with a minimum size of 1 (the immediate neighbourhood, Figure 4). Finally, we need to use a function to decide which neuron is activated for each input pattern. Among different distance functions such as Euclidean, Chebyshev or Manhattan distance functions, we selected the Minimal Euclidean distance, as it was the one which offered a better performance.

6. Results

Once the algorithms, the scenario, and the dataset are defined, the proposed experiments can be carried out and its results presented.

Firstly, we briefly address the results of the classification achieved by the SOM while dealing with the original dataset of outliers provided by the SDSS classification pipeline, so as to show the power

[1]It provides up to 8 threads.
[2]It provides up to 12 threads.

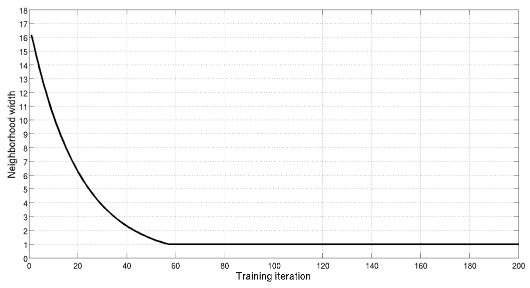


Figure 4: Neighborhood width decay with iterations. The plot was cut at 200 iterations since the width of the neighbourhood remains 1 up to the end of the training phase.

and usefulness of this method. As it was stated in Section 2, we are trying to proccess a dataset which contains observations whose nature is mostly unknown, so we can not measure the accuracy given by the SOM algorithms over this dataset (Using a reference dataset, where all the observations are appropriately labeled, we were able to obtain an accuracy near to the 75%). Figure 5 shows such classification, where we can observe that the clusters have been labeled after performing some template matching across a predefined set of well known spectral templates. Thus, a scientist can explore each cluster deeply in order to clarify, even more, the nature of the observations. This discussion was thoroughly addressed in previous works [15, 16].

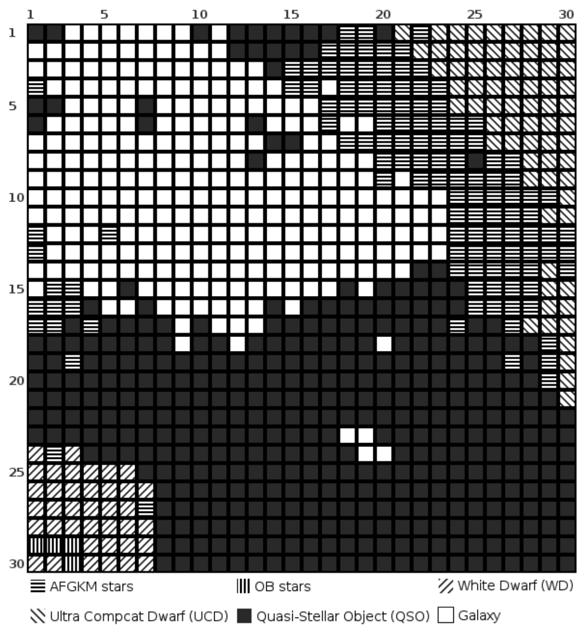


Figure 5: Results of the SDSS outliers classification obtained by the SOM.

We then focus on the performance achieved by the SOM for both the sequential and the Hadoop-based algorithms. Table 3 show the execution times that were measured for several datasets with different sizes. Note that the Hadoop-based algorithm takes more time to be executed than the sequential approach when the size of the dataset is fairly small, since we have to take into account the processes initialization and communication overhead while using the Hadoop approach. However, when the volume of the dataset increases up to a huge number of observations we can reduce the execution times significantly. Such a behaviour can be observed in Figure 6.

Size	**S. machine (h)**	**Hadoop cluster (h)**
10125	10.75	14.05
20250	21.44	25.08
81000	84.57	39.23
324000	338.11	43.09
1296000	1364.55	55.47
5184000	5454.51	224.08
10368000	10909.03	426.45

Table 3: Obtained performance for SOM algorithms over different datasets.

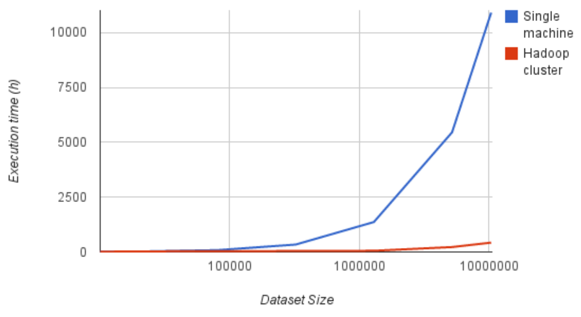


Figure 6: Obtained performance for SOM algorithms over different datasets.

7. Discussion and future work

This paper addresses the situation that the Outlier Analysis module from the CU8 of DPAC for the Gaia mission will encounter. We need to process huge amounts of data (the outliers detected by the main classification module from CU8, DSC) by means of unsupervised techniques. To work this out, we propose the use of one type of ANN that can reduce the input dimensionality of the data while discovering relationships between the features of the input observations, the Self-Organizing Maps (Section 3). Such method needs a learning or training algorithm, so that the similarity between the observations can be found and then they can be grouped. To this effect, we propose the use of the batch learning algorithm because it is more stable than the online algorithm. However, it is not feasible to execute this algorithm in a sequential way, since it is expected to process over 10^8 observations (10% of the total) and we can not manage such an amount of memory in one single machine. Moreover, the DPAC software pipeline will use a Hadoop cluster to carry out this processing. We have therefore proposed a distributed version of the algorithm based on such a framework (Section 4). Execution times were measured (Section 6) in order to compare the performance between both the sequential and the Hadoop-based algorithm.

On the basis of the results we can say that the Hadoop-based algorithm offers a very interesting solution for the training of the SOM, since we can reduce the amount of time required for execution in a significant manner for large datasets. Furthermore, the achieved solution is scalable, since the more number of Map/Reduce tasks one uses, the better performance one obtains. Indeed, the number of map tasks was determined by the number of splits into which the input file was divided, so the number of map tasks will be larger for larger input datasets.

Finally, although the Hadoop-based algorithm achieves much better performance than the sequential algorithm, we should look forward for new optimizations, such as considering a straight neighbourhood as the iterations succeed themselves, trying to exploit the fact that the SOM is getting ordered. We can also explore new training algorithms for the SOM or even new methods, such as Growing Neural Gas (GNG) or Adaptive Resonance Theory Maps (ART-Maps). On the other hand, we can continue this work and try to combine the Hadoop framework with some other computing techniques, obtaining even better performance.

References

[1] J. H. J. de Bruijne. Science performance of gaia, esa's space-astrometry mission. *Astrophysics and Space Science*, page 68, March 2012.

[2] D. Ordóñez-Blanco, B. Arcay, C. Dafonte, M. Manteiga, and A. Ulla. Object classification and outliers analysis in the forthcoming Gaia mission. *Lecture Notes and Essays in Astrophysics*, 4:97–102, December 2010.

[3] Y. Isasi, F. Figueras, X. Luri, and A. C. Robin. GUMS & GOG: Simulating the Universe for Gaia. In Jose M. Diego, Luis J. Goicoechea, J. Ignacio González-Serrano, and Javier Gorgas, editors, *Highlights of Spanish Astrophysics V*, Astrophysics and Space Science Proceedings. Springer Berlin Heidelberg, 2010.

[4] Jordi, C., Gebran, M., Carrasco, J. M., de Bruijne, J., Voss, H., Fabricius, C., Knude, J., Vallenari, A., Kohley, R., and Mora, A. Gaia broad band photometry. *A&A*, 523:A48, 2010.

[5] Xiaohui Liu, Gongxian Cheng, and J.X. Wu. Analyzing outliers cautiously. *Knowledge and Data Engineering, IEEE Transactions on*, 14(2):432–437, mar/apr 2002.

[6] X. Liu, G. Cheng, and J.X. Wu. Identifying the measurement noise in glaucomatous testing: An artificial neural network approach. *Artificial Intelligence in Medicine*, 6(5):401–416, 1994.

[7] I. T. Jolliffe. *Principal Component Analysis.* Springer, second edition, October 2002.

[8] Rui Xu and D. II Wunsch. Survey of clustering algorithms. *Neural Networks, IEEE Transactions on*, 16(3):645–678, may 2005.

[9] A. Baraldi and P. Blonda. A survey of fuzzy clustering algorithms for pattern recognition. I. *Systems, Man, and Cybernetics, Part B: Cybernetics, IEEE Transactions on*, 29(6):778–785, dec 1999.

[10] T. Warren Liao. Clustering of time series data-a survey. *Pattern Recogn.*, 38(11):1857–1874, November 2005.

[11] T. Kohonen, M. R. Schroeder, and T. S. Huang, editors. *Self-Organizing Maps.* Springer-Verlag New York, Inc., Secaucus, NJ, USA, 3rd edition, 2001.

[12] A. Naim, K. U. Ratnatunga, and R. E. Griffiths. Galaxy Morphology without Classification: Self-organizing Maps. *ApJS*, 111:357, August 1997.

[13] J. E. Geach. Unsupervised self-organized mapping: a versatile empirical tool for object selection, classification and redshift estimation in large surveys. *MNRAS*, 419:2633–2645, January 2012.

[14] M. J. Way and C. D. Klose. Can Self-Organizing Maps Accurately Predict Photometric Redshifts? *PASP*, 124:274–279, March 2012.

[15] Diego Fustes, Carlos Dafonte, Bernardino Arcay, Minia Manteiga, Kester Smith, Antonella Vallenari, and Xavier Luri. SOM ensemble for unsupervised outlier analysis. Application to outlier identification in the Gaia astronomical survey. *Expert Syst. Appl.*, 40(5):1530–1541, April 2013.

[16] Fustes, D., Manteiga, M., Dafonte, C., Arcay, B., Ulla, A., Smith, K., Borrachero, R., and Sordo, R. An approach to the analysis of SDSS spectroscopic outliers based on self-organizing maps. *Astronomy & Astrophysics*, 559:A7, 2013.

[17] Jean-Claude Fort, Patrick Letrémy, and Marie Cottrell. Advantages and drawbacks of the batch kohonen algorithm. In Michel Verleysen, editor, *ESANN 2002, 10th Eurorean Symposium on Artificial Neural Networks, Bruges, Belgium, April 24-26, 2002, Proceedings*, pages 223–230, 2002.

[18] Tom White. *Hadoop: The Definitive Guide.* O'Reilly Media, Inc., 3rd edition, may 2012.