Procedia Computer Science

Volume 53, 2015, Pages 9–18

2015 INNS Conference on Big Data

Evolving Classifier TEDAClass for Big Data

Dmitry Kangin*, Plamen Angelov*, **, Jose Antonio Iglesias***, Araceli Sanchis ***

* Data Science Group, Computing & Communications, Lancaster University, UK {d.kangin, p.angelov}@lancaster.ac.uk

** Chair of Excellence, Carlos III University of Madrid, Spain

*** Carlos III University of Madrid, Computer Science Department, Spain

Abstract

In the era of big data, huge amounts of data are generated and updated every day, and their processing and analysis is an important challenge today. In order to tackle this challenge, it is necessary to develop specific techniques which can process large volume of data within limited run times.

TEDA is a new systematic framework for data analytics, which is based on the *typicality* and *eccentricity* of the data. This framework is spatially-aware, non-frequentist and non-parametric. *TEDA* can be used for development of alternative machine learning methods, in this work, we will use it for classification (*TEDAClass*). Specifically, we present a *TEDAClass* based approach which can process huge amounts of data items using a novel parallelization technique. Using this parallelization, we make possible the scalability of *TEDAClass*. In that way, the proposed approach is particularly useful for various applications, as it opens the doors for high-performance big data processing, which could be particularly useful for healthcare, banking, scientific and many other purposes.

Keywords: Big Data, TEDA, AnYa, Evolving Systems for Big Data Analytics

1. Introduction

Huge amounts of data are generated every day in the modern society, mostly in a digital form. This makes the old approach to store all data items and for further processing and analysis impossible and gives rise to the term big data. Big data can be defined as a scale of dataset that goes beyond existing database management tool capabilities of data collection, storage, management, and analysis capabilities [1]. Although, the most common trait of big data is Volume, it is typically defined by more Vs such as Volume, Variety, Velocity, Veracity, Volatility, etc.

There are many different applications in which Big Data techniques are applicable: data mining, predictive analytics, geo-analysis, natural language processing and pattern recognition. Nowadays, since the computational power and reliability of the contemporary computers continue to grow, the data mining problems for big data is becoming widespread. These problems are shifting from the instrument for government, large corporations, banks to mass users. Big Data can be classified taking into account

E-mail addresses: d.kangin@lancaster.ac.uk (Dmitry Kangin), p.angelov@lancaster.ac.uk (Plamen Angelov), jiglesia@inf.uc3m.es (José Antonio Iglesias), masm@inf.uc3m.es (Araceli Sanchis).

Selection and peer-review under responsibility of the Scientific Programme Committee of INNS-BigData2015

doi:10.1016/j.procs.2015.07.274

the data type: (1) Structured (data are stored in fixed field), (2) Semi-structured (data are not stored in fixed field but the data includes metadata or schema), and (3) Unstructured (data are not stored in fixed field).

While processing structured and semi-structured data poses problems related primarily to the storage, retrieval and tagging, when unstructured data is concerned the primary problem is organising and making sense from it. Nowadays, unlike in the past century, vast majority of the data is unstructured. Simpler algorithms, such as retrieval, search and clustering can be approached using Map-Reduce \cite{SinghReddy14jobd} and through parallelisation can be scaled into a number of processing units (PU). More complex machine learning algorithms, however, such as classification, prediction, image processing etc., which are often iterative, are significantly more difficult to parallelise and scale up.

In this paper, we propose an algorithm for classification of Big Data based on the recently introduced TEDAClass (Typicality and Eccentricity based Data Analytics) approach [2]. TEDAClass itself is a neuro-fuzzy classifier which can be of zero or first order. The zero order TEDAClass has the class label as output. The first order version is using a mixture of (linear) regression models combined by a fuzzy weight proportional to their local density [3]. TEDAClass is based on the recently proposed alternative data analytics called TEDA [2] [4] [5].

It is important to stress that the proposed approach has been designed as a context-independent approach. It means that it is not restricted to any particular application. The proposed approach also does not have restrictive prior assumptions that are typical for alternative statistical, fuzzy rule-based and other approaches. This is due to being completely data-driven and based on the data density derived from data items.

This paper is organized as follows: In the next section, the background and related work to the proposed problem are discussed. In addition, the TEDA framework on which the proposed approach is based on is also outlined. Section 3 details the structure of the proposed parallelization approach (called $\text{TEDAClass}_{\text{BDp)}}$. Section 4 describes initially an intuitive illustrative example (IRIS classification data), and one larger realistic approach (ETL1 data set) presents the experimental settings and the obtained results. Finally, Section 5 makes the conclusions and outlines the future work.

2. Background and Related Work

Different scientific fields are becoming increasingly data-driven which also requires new approaches to be developed within the Computer Science to reflect this. For example, social computing [6] is becoming a discipline on its own; same is true for the bioinformatics [7], econometrics, astronomy is increasingly data-driven [8], etc. are examples of these fields. Big data require new type of computational approaches and techniques in order to be able to process efficiently large volumes of data within limited run times.

Various approaches where proposed specifically for Big Data recently, like those described in [9]. Perhaps the most popular approach is Hadoop and Map-reduce [10], but it is applicable to simpler problems such as information storage and retrieval, clustering, samples-centred approaches such as SVM etc. Most of the Big Data specific approaches are adaptations of previously existing and well known machine learning approaches, for example, k-means clustering [11], SVM [12], fuzzy and probabilistic clustering Fuzzy logic algorithms [13].

In the following subsections the two important aspects of the proposed approach are explained in more detail: i) the parallelization structure, and ii) the *TEDA* classifier.

2.1.1. Data processing parallelization

The parallelization concept is about partitioning the data into chunks and processing them by several independent (possibly distributed) processors in parallel. Partial results of different processors are then merged in order to obtain the overall result. It is clear that simpler problem such as counting words in a document (sum of sums) or finding a maximum or mean value (max of maxes or mean of means) that are needed for storage and retrieval, clustering or SVM are easy to parallelise. More complex problems, however, such as classification, prediction, image processing etc. which require iterative solutions are significantly more difficult to parallelize. The specific manner different processors work and their results are merged is essential for the final result.

We propose two different types of architecture for data processing (Figure1) and use one of them which is based on parallelization of the TEDAClass classification algorithm. The Data Distributor node distributes the data in blocks (data chunks). The Data Processor nodes obtain a result from the data chunk provided to them. Functioning of all the nodes is exactly the same (the only difference is the data that are being provided). Finally, the Fusion Centre node merges the partial results obtained by the different Processor nodes.

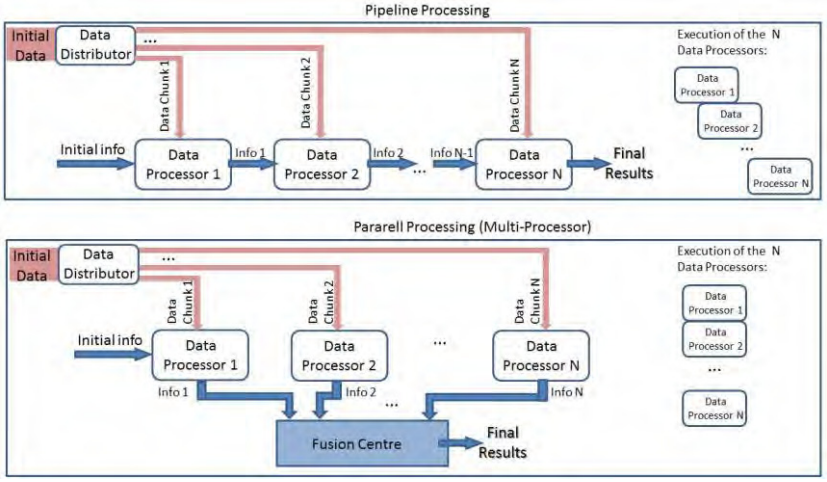

Figure 1. Different data processing architectures

In what follows the two architectures are briefly explained:

1. Pipeline Processing (Figure 1 - top): A pipeline is configured as a series of processing nodes. The data is divided into chunks and each node receives sequentially its corresponding data chunk. Each node receives the meta data (partial result from the predecessor node, e.g. means or cluster centres, sum scalar products, etc). It then updates it using its own data chunk and forwards the results to the next node. The last node produces the overall result. This scheme preserves the order of processing in the sequential algorithm. It reduces the amount of data being processed by and, thus, the computational complexity and burden on each node. However, it does not reduce the time needed to solve the problem. Therefore, such a pipeline (sequential) architecture can be useful in solving a number of problems in parallel. Let's imagine, we have a number of different problems (or data sets) where this number is not bigger than the number of processors (N). Then the pipeline architecture can be used to solve all these problems at the same time by

the first processor starting to process the first data chunk of the first problem; the second processor then starts to process the first data chunk of the second problem and so on. When the first data processor is ready with the partial result of the first problem (using the first data chunk) and the second processor is ready with the partial data of the second problem etc. the processors swap the meta data between them in such a way that the second processor now continues to process the first data chunk, the first processor – the Nth chunk etc. In this way, the pipeline architecture can be used to reduce both the computational complexity and time required to solve (up to) N different problems. In addition, this architecture can guarantee obtaining exactly the same result as if all the data is being processed by a single processor [14].

2. Parallel Processing (Figure 1- bottom): In contrast to the pipeline architecture, in this case we have a full parallelization. Initially, the data is divided and the different chunks are sent to the individual processing nodes in a parallel manner. The nodes sent their partial result when ready to the Data Fusion Centre. This architecture offers full parallelisation, but does not involve a synchronisation and obtaining exactly the same result as if all the data is being processed by a single processor which is the case for the pipeline (sequential) architecture. Instead, the result can be approximately the same gaining reduction in time and computational complexity.

Further in this paper we will consider TEDAClass classifier benefiting from the parallel architecture. We will expand on the sequential architecture for TEDAClass and other algorithms in subsequent publications.

2.1.2. TEDAClass

In this section, we will outline the main characteristics of the TEDA framework. More details about this alternative data analytics framework can be found elsewhere [2] [4] [5].

The TEDA framework is based on the spatially-aware concepts of typicality and eccentricity which represent the density and proximity in the data space. The Typicality and Eccentricity-based Data Analytics (TEDA), is a data driven, parameters and assumptions free approach which differs from the traditional probability theory, the approaches based on belief and possibility, or those based on subjective experts such as fuzzy logic. In what follows, it is briefly outlined.

Let us consider the data samples sequence: $\{\boldsymbol{x}_1, \boldsymbol{x}_2 \dots \boldsymbol{x}_k \dots\}, \boldsymbol{x}_i \in \mathbb{R}^n, i \in \mathbb{N}$, where k is a time index within the data sequence. We can define some particular distance $d(\boldsymbol{x}, \boldsymbol{y})$ (Mahalanobis, cosine or any other which seems to be appropriate within the problem context) between the vectors of the data sample set. We can also calculate the accumulated proximity: $\pi^k(\boldsymbol{x}) = \sum_{i=1}^{k} d(\boldsymbol{x}, \boldsymbol{x}_i), k \geq 1$. Here we shortly denote $\pi_j^k = \pi^k(\boldsymbol{x}_j)$.

Figure 1: Possible architectures for data processing parallelisation.

The eccentricity (ξ) of a particular $\mathrm{j}^{\mathrm{th}}(\mathrm{j} > 1)$ data sample is defined as:

$$\xi^k(\boldsymbol{x}) = \frac{2\pi_j^k}{\sum_{i=1}^{k} \pi_j^k} = \frac{2\sum_{i=1}^{k} d(\boldsymbol{x}_j, \boldsymbol{x}_i)}{\sum_{i=1}^{k}\sum_{l=1}^{k} d(\boldsymbol{x}_i, \boldsymbol{x}_l)}, \sum_{i=1}^{k} \pi_j^k > 0. k > 2 \quad (1)$$

The typicality (τ) of a particular $j^{th}(j > 1)$ data sample is defined as:

$$\tau_j^k = 1 - \xi_j^k, \mathrm{k} > 2, \sum_{i=1}^{k} \pi_i^k > 0 \quad (2)$$

These definitions can be normalized (see [2] [4] [5]) with the advantage that they do not require any prior assumption such as having infinite amount of data, independence of the individual data vectors, smooth pre-defined distributions, having positive likelihood of infeasible values, etc. which are typical for the traditional probability theory [15]. For example, the normalized typicality (t) of a particular j^{th} ($j > 1$) data sample is defined as:

$$t_j^k = \frac{\tau_j^k}{k-2}, k > 2. \quad (3)$$

In addition, and it is very important for the use in big data problems, these definitions can be calculated recursively by updating only the global or local mean (μ) and scalar product (*X*).

TEDA framework applies to various types of underlying models, but in this paper we will consider the specific neuro-fuzzy approach recently introduced by Angelov and Yager called *AnYa* [16]:

$$F = \{R_i\}, i = 1, N, R_i(\boldsymbol{x}) : IF\ (\boldsymbol{x} \sim \boldsymbol{x}_i^*)\ THEN\ y_i = \overline{\boldsymbol{x}}^T \Theta_i \quad (4)$$

where F is a fuzzy rule set, R_i is a particular rule of this set, $y_i \in C, \boldsymbol{x} \in \mathbb{R}^n, F: \mathbb{R}^n \to C, \mathbb{R}^n$ is a feature space, C is a class space, $\boldsymbol{x}_i^*$ is cluster representative point (focal point), Θ_i is a design matrix for linear regression, $\sim$ is a closeness relation (fuzzy association/membership) of the point $\boldsymbol{x}$ to the fuzzy rule $R_i(\boldsymbol{x})$.

The procedure of classification using *TEDAClass,* as well as a learning algorithm, is the same as for *eClass* [17] and *AutoClass* [3]. The main difference of *TEDAClass* algorithm is that the inference is based on the ratio of the local *typicality* and *eccentricity* rather than on the de-fuzzification as it is in *eClass*. The firing rate of each *AnYa* type fuzzy is derived as a ratio of the normalised typicality:

$$w_i^k = \frac{t_i^k}{\sum_{j=1}^N t_j^k} \quad (5)$$

where normalised typicality t_i^k is given over all the elements assigned to this fuzzy rule.

3. The Proposed Approach TEDAClass$_{BDP}$

The approach presented in this paper (*TEDAClass$_{BDp}$* which stands for *TEDAClass*, Big Data parallelized) is based on the *TEDAClass* classifier executed by each data processor node in parallel manner (Figure 1-bottom). Using the parallelization it is possible to scale up the classification algorithm.

The proposed parallel architecture reminds the well-known client-server architecture [18]. The 'Fusion Centre' block has a role similar to that of a server, whilst the 'Data Processor' nodes have a role similar to that of clients.

Despite the original *TEDAClass* approach [5] being online and dynamically evolving, in this paper we use a much more limited offline version of *TEDAClass* in order to avoid the expensive communication between the processing nodes and the Data Fusion Centre. In our realization, the Data Fusion Centre controls the processing of the entire data and holds all the meta data such as means, sum scalar products, number of data clouds formed, etc.

The data is separated into chunks and passed to the Processors where the data clouds are being formed and parameters of the regression models in the consequent parts are being updated; then these partial results are being then passed to the Data Fusion Centre which merges the data clouds and updates the parameters.

3.1.1. Our algorithm description

In this section, the algorithm description is presented. Here we consider the case of Mahalanobis distance without loss of generality.

1) *Fusion Centre initialisation*:$\mu^k = \emptyset, \mu^k_{x^T x} = \emptyset$, N is a count of data processors, N_D is the cardinality of the data set, N_c is the cardinality of a data chunk, X is a data set with cardinality N_D.

2) *Fusion Centre*: Select first $N_0 = \min(N_c/N\ , N_D)$ objects from the data set and transmit first N_0 data to the data processor.

3) *Data Processor 1:* calculate the parameters of the clusters (the mean $\{\mu_i^{N_{1,i}}\}$, covariances $\left\{\mu_{(xx^T)_i}^{N_{1,i}}\right\}, i = 1 \dots N_{Cl}$, where N_{Cl} is the number of clouds and i is a current cloud, $N_{1,i}$ is a support of the i-th cloud) as in traditional *TEDAClass* and transmit it back to the Fusion Centre.

4) For $k = N_0 + 1, N_0 + 1 + N_C, \dots N - N_C$ do

   a. Fusion Centre: Divide $\mathrm{X_k} = \{\mathrm{x_k} \dots \mathrm{x_{k+N_C}}\}$ into $\mathrm{N_C}$ groups $\{\mathrm{G_1}, \mathrm{G_2}, \dots \mathrm{G_N}\}$, preferably of equal size.

   b. Data Processor ($\mathrm{j = 1{:}\,N}$): update the means $\{\mu_{\mathrm{i,j}}^{\mathrm{N_{i,j}}}\}$, covariances $\left\{\mu_{(\mathrm{xx^T})_\mathrm{i},\mathrm{j}}^{\mathrm{N_{i,j}}}\right\}, \mathrm{i} = 1 \dots \mathrm{N_{Cl}^j}$, where $\mathrm{N_{Cl}^j}$ is the number of the data clouds found by each Data Processor j, i is the current data Cloud according to the equation (6) and (7) and $\mathrm{N_{Cl}^j}$ is a number of data items to process at each Data Processor j. Update the fuzzily-weighted RLS parameters according to (11) and (12).

   c. Fusion Centre: Update the means $\{\mu_{\mathrm{i}}^{\mathrm{N_i}}\}$, covariances $\left\{\mu_{(\mathrm{xx^T})_\mathrm{i}}^{\mathrm{N_i}}\right\}, \mathrm{i} = 1 \dots \mathrm{N_{cl}}$, where $\mathrm{N_{cl}}$ is the number of data clouds, according to the equations (8) and (9), and then merge the data clouds that are close to each other.

After the step 4.2, the Data Processor gets back the meta data about all data clouds from the Fusion Centre. Then, it performs the TEDAClass algorithm as in [2]. When adding a new data item $\mathrm{x_k}$ to the j-th data cloud with support $\mathrm{N_j}$, the means are accumulated for the i-th data clouds for j-th Data Processor are given as follows:

$$\mu_{i,j}^0 = 0, \mu_{i,j}^{N_{i,j}} = \frac{N_{i,j} - 1}{N_{i,j}} \mu_{i,j}^{N_{i,j}-1} + \frac{x_k}{N_{i,j}}, \tag{6}$$

$$\mu_{(xx^T)_{i'}j}^0 = 0, \mu_{(xx^T)_{i'}j}^{N_{i,j}} = \frac{N_{i,j} - 1}{N_{i,j}} \mu_{(xx^T)_{i'}j}^{N_{i,j}-1} + \frac{x_k x_k^T}{N_{i,j}}, \tag{7}$$

At step 4.3, the Fusion Centre obtains the data from all the data clouds, and then merges them when the following criteria is satisfied, or otherwise adds it to the set of Fusion Centre clouds:

$$R_d = \left\{R_i : j > i, R_j \in F, t_j^k\left(\mu_i^k\right) > T(k)\right\},$$

i.e. typicality in given instant of time for any rule does not exceed a threshold $T(k)$, depending on the instant of time, that shows that the data clouds are too close to each other. The merger is carried out according to the following equations:

$$\mu_l^0 = 0, \mu_l^k = \frac{k - N_{i,j}}{k} \mu_l^{k-N_{i,j}} + \frac{N_{i,j}}{k} \mu_{i,j}^{N_{i,j}}, \tag{1}$$

$$\mu^{0}_{(xx^T)_l} = 0, \mu^{k}_{(xx^T)_l} = \frac{k - N_{i,j}}{k}\mu^{k-N_{i,j}}_{(xx^T)_i,j} + \frac{N_{i,j}}{k}\mu^{N_{i,j}}_{(xx^T)_i,j}, \quad (9)$$

where $\mu^{N_{i,j}}_{i,j}$ is the mean of i-th cluster of j-th data processor, and l is the index of the cluster within the Fusion Centre.

To obtain the overall classifier, we apply fuzzily weighted recursive least squares (RLS) method to build up the linear regression for each of the data clouds. This regression is used to estimate Θ_i and resolves the following least-squares problem:

$$\left(Y_i^k - \left[\Psi_i^{\mathrm{k}}\right]^T \Theta_i^{\mathrm{k}}\right)^T \left(Y_i^k - \left[\Psi_i^{\mathrm{k}}\right]^T \Theta_i^{\mathrm{k}}\right) \to \min_{\Theta_i}, \quad (10)$$

where Y_i is the matrix of ground-truth output results for the i-th fuzzy rule, Θ_i is a design matrix, $\Psi_i = \{\boldsymbol{x} w_i^k(\boldsymbol{x})\}$ is a matrix of the RLS inputs with the membership weights $w_i^k(\boldsymbol{x})$. The RLS regression [3]. is updated at each of the Data Processors as:

$$\Theta_i^k = \Theta_i^{k-1} + C_i^k \Psi_i^{\mathrm{k}} \left(y_i^k - \Psi_i^{\mathrm{k}}\, \Theta_i^{\mathrm{k}-1}\right), \Theta_i^k = 0, \quad (2)$$

$$C_i^k = C_i^{k-1} - \frac{C_i^{k-1}\Psi_i^{\mathrm{k}}\left[\Psi_i^{\mathrm{k}}\right]^T C_i^{k-1}}{1 + \left[\Psi_i^{\mathrm{k}}\right]^T C_i^{k-1}\Psi_i^{\mathrm{k}}}, C_i^1 = \Omega \mathrm{I}. \quad (3)$$

and similarly on the Fusion Centre using the Data Processors regression values C_i^k and Θ_i^k as an input.

4. Experimentation

This section is divided in two subsections: first, on the simple illustrative example we demonstrate how our approach works. This is then followed by a more realistic example.

4.1.1. Illustrative Example

To illustrate the different stages of the algorithm we use the well-known Iris dataset [19] for classification. It contains 50 samples/vectors for each of the 3 different classes, representing types of the Iris flower. Figure 2 demonstrates how the proposed *TEDAClass*$_{BDp}$ approach works. On the top of the figure, we can see the projection on the first 2 axes of all data samples with different classes shown with different colour and symbol. In this particular case, we used 5 Data Processors because the data set is small and the aim is simply to illustrate the work process and to proof the concept. Thus, the Data Distributor node divides the data in 5 different chunks. Each of the Data Processors then applies *TEDAClass* to the specific data chunk and obtains the AnYa type rules. Data clouds [3] are then formed which do not have a specific shape or parameters. However, for illustrative purposes only we depicted on the Figure 2 the 2D ellipses of the respective covariance matrices (only those having support ≥2 are presented on the images). Then the Data Fusion Centre node merges the data clouds (we can observe in the bottom right of the figure the result of this process again simplified for a 2D illustration).

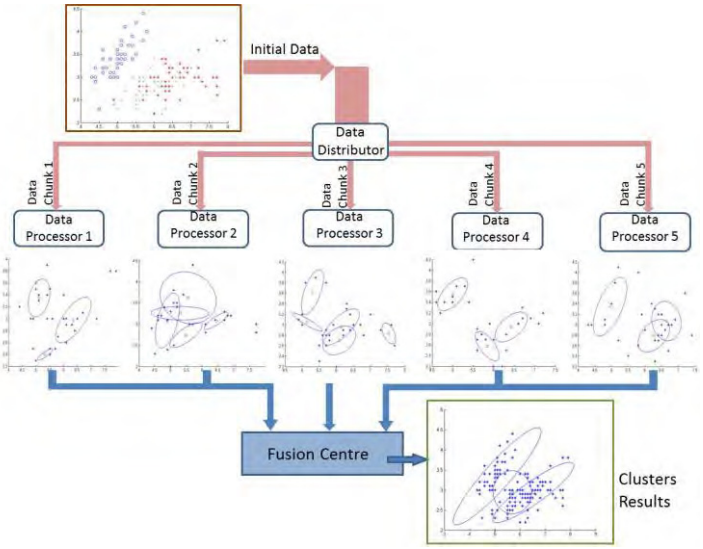

Figure 2: Example – How the proposed approach works.

4.1.2. Experimentation Results

In order to evaluate the newly proposed TEDAClass$_{BDp}$ algorithm, we applied it *to the so called ETL1 data set (*http://projects.itri.aist.go.jp/etlcdb/etln/etl1/etl1.htm) which consists of 12,888 different images of handwritten digits labelled with its corresponding digit. Each sample is represented by a 64 x 64 image. For the training, 1,857 digits were selected, and 11,031 images were selected for validation. The Matlab software was written to evaluate the proposed approach. .

To make the image data tractable for the vector classification algorithm, the feature extraction is first carried out by the method proposed in [20]. This method trivially combines Gabor filter based gist descriptor [21] combined with Haar features [22]

The results of applying *TEDAClass$_{BDp}$* to this data set are shown in (Table 1 and figure 3) and they demonstrate that approximately the same high accuracy can be achieved by each Data Processor working with only part of the data (up to 1/5th) and thus the data being processed much faster since we are using several (up to 5) Data Processors. Although, the number of data clouds is increasing significantly, they are being reduced by merging and other techniques subsequently.

The proposed classifier is tailored for parallel processing of Big Data by splitting the data into chunks and processing them independently yet achieving approximately the same classification result (using 1 or 5 data processors). The proposed parallelisation technique allows reducing both, the computational complexity and time required to process all the data tremendously (in the order of *N*, *O(N)*, where *N* is the number of Data Processors). From the update equations we can see, that the update depends linearly from the number of fuzzy rules. We also observed that the more data we process, the more complex patterns we can find. Therefore, more data clouds are being created. Therefore, the overall gain can be even more because of the growing complexity of the data patterns within the large chunks (one can see it from the table 1 and figure 3, where the relative computational time is the part of the training time comparing to the training time for one data processor). In the overall solution, we accumulate all the patterns in the final node, and in this case we do not operate with a huge amount of data, but only with fuzzy rule merger. This means, that we do not deal with Big Data directly in the Data Fusion Centre, but delegate it to the Data Processors, and the increase of the d ata chunks gives the parallelisation at least in number of nodes times.

4.2. Table 1: ETL1 Performance dependency of # nodes

#Data Processors	Accuracy	#Data Clouds	Computational time compared to one node
1	0.9540	79	1
2	0.9517	103	0.3835
3	0.9491	227	0.2504
4	0.9490	232	0.2371
5	0.9428	454	0.1256

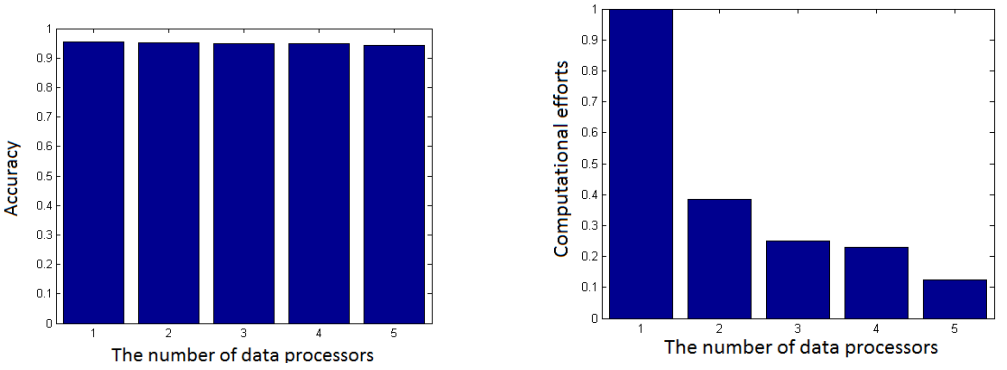

Figure 3: Accuracy (left) and Computational Time (right) depending of the number of nodes.

5. Conclusions and Future Work

In this article, a new algorithm for processing Big Data is proposed. The approach is based on the classifier *TEDAClass* and it gives new opportunities for application of this algorithm for huge amounts of data. It is obtained by adapting the original algorithm to a block-wise processing of the data. This approach gives new opportunities to distribute the data processing for the algorithm and to bring new applications to this method. The results give a strong evidence of the proposed concept, showing no significant reduction in the accuracy while the computational complexity and time required are reduced by a factor of N. Same as the original *TEDAClass*, the proposed $TEDAClass_{BDp}$ approach does not need prior assumptions (except the number of the Data Processors we will use). In addition it can be recursive and thus online and real-time (not completely exploited in this paper). These characteristics make this method a good candidate for the problems in which thousands or millions of data units need to be processed fast.

Acknowledgements

This work has been supported by the Spanish Government under Projects: TRA2011-29454-C03-03 and TRA2013-48314-C3-1-R, as well as by the Chair of Excellence programme by Santander Bank for 2015.

References

[1] M. G. Institute, “Big data: The next frontier for innovation, competition, and productivity,” 2011.

[2] P. Angelov, “Outside the box: an alternative data analytics framework,” *Journal of Automation, Mobile Robotics and Intelligent Systems,* vol. 8, pp. 29-35, 2014.

[3] P. Angelov, Sense and Avoid in UAS: Research and Applications, John Wiley and Sons, 2012.

[4] P. Angelov, "Anomaly detection based on eccentricity analysis," in *Evolving and Autonomous Learning Systems (EALS), 2014 IEEE Symposium on*, 2014.

[5] D. Kangin and P. Angelov, "Evolving Clustering, Classification and Regression with TEDA," in *The International Joint Conference on Neural Networks*, 2015.

[6] W. Mason, W. J. Vaughan and H. Wallach, "Computational social science and social computing," *MAchine Learning,* vol. 95, no. 3, pp. 257-260, 2014.

[7] N. Savage, "Bioinformatics: Big data versus the big C," *Nature,* vol. 509, no. 7502, pp. 566-567, 2014.

[8] K. J. Edwards and M. M. Gaber, Astronomy and Big Data: A Data Clustering Approach to Identifying Uncertain Galaxy Morphology, Springer Publishing Company, Incorporated, 2014.

[9] D. Singh and C. K. Reddy, "A Survey on Platforms for Big Data Analytics," *Journal of Big Data,* vol. 2, no. 8, pp. 8:1-8:20, 2014.

[10] J. Dean and S. Ghemawat, "MapReduce Simplified Data Processing on Large Clusters," *Commun. ACM,* vol. 51, no. 1, pp. 107-113, 2008.

[11] P. Anchalia, A. Koundinya and N. Srinath, "MapReduce Design of K-Means Clustering Algorithm," in *2014 International Conference on Information Science And Applications (ICISA)*, 2013.

[12] Z. Sun and G. Fox, "Study on Parallel SVM Based on MapReduce," in *International Conference on Parallel and Distributed Processing TEchniques and Applications*, 2012.

[13] R. J. Hathaway and J. C. Bezdek, "Extending Fuzzy and Probabilistic Clustering to Very Large Data Sets," *Comput. Stat. Data Anal.,* vol. 51, no. 1, pp. 215-234, 2015.

[14] P. Angelov, "Machine Learning (Collaborative Systems)". US Patent US 8250004, 21 8 2012.

[15] T. Hastie, R. Tibshirani and J. H. Friedman, The elements of statistical learning: data mining, inference, and prediction, New York: Springer-Verlag, 201.

[16] P. Angelov and R. R. Yager, "A new type of simplified fuzzy rule-based system," *Int. J. General Systems,* vol. 41, no. 2, pp. 163-185, 2012.

[17] P. Angelov and X. Zhou, "Evolving Fuzzy-Rule-Based Classifiers From Data Streams," *Fuzzy Systems, IEEE Transactions on,* vol. 16, no. 6, pp. 1462-1475, 2008.

[18] C. Leopold, Parallel and Distributed Computing: A Survey of Models, Paradigms and Approaches, John Wiley And Sons, Inc., 2001.

[19] R. A. Fisher, "The Use of Multiple Measurements in Taxonomic Problems," *Annals of Eugenics,* vol. 7, no. 7, pp. 179-188, 1936.

[20] P. Angelov, D. Kangin, Z. Xiaowei and D. Kolev, "Symbol recognition with a new autonomously evolving classifier AutoClass," in *IEEE Conference on Evolving and Adaptive Intelligent Systems*, 2014.

[21] A. Oliva and A. Torralba, "Building the Gist of a Scene: The Role of Global Image Features in Recognition Visual Perception," in *Progress in Brain Research*, 2006.

[22] A. Haar, "Zur Theorie der orthogonalen Funktionensysteme," *Mathematische Annalen,* vol. 69, no. 3, pp. 331-371, 1910.