Depth-Based Outlier Detection Algorithm

Miguel Cárdenas-Montes

Centro de Investigaciones Energéticas Medioambientales y Tecnológicas,
Department of Fundamental Research, Madrid, Spain
miguel.cardenas@ciemat.es

Abstract. Nowadays society confronts to a huge volume of information which has to be transformed into knowledge. One of the most relevant aspect of the knowledge extraction is the detection of outliers. Numerous algorithms have been proposed with this purpose. However, not all of them are suitable to deal with very large data sets. In this work, a new approach aimed to detect outliers in very large data sets with a limited execution time is presented. This algorithm visualizes the tuples as N-dimensional particles able to create a potential well around them. Later, the potential created by all the particles is used to discriminate the outliers from the objects composing clusters. Besides, the capacity to be parallelized has been a key point in the design of this algorithm. In this proof-of-concept, the algorithm is tested by using sequential and parallel implementations. The results demonstrate that the algorithm is able to process large data sets with an affordable execution time, so that it overcomes the curse of dimensionality.

Keywords: Outlier Detection, Parallel and Distributed Data Mining, Big Data, Large-Scale Learning, Scalability.

1 Introduction

The modern society is tackling with a deluge of information. The challenge is to convert this huge and increasing volume of information into useful knowledge. In order to achieve this purpose, numerous techniques have been developed to extract pieces of knowledge from large data sets. In the past, this area has been termed as knowledge discover or data mining, and nowadays it is moving toward a blurry set of techniques [1,2] which are termed big data.

One of the typical problems in data mining is to identify in a data set, the tuples or objects that can be considered as special or anomalous elements. Usually these elements are labelled as outliers. Outlier detection is an important task in fraud detection, intrusion detection or simply for cleaning noisy data.

Many techniques have been proposed to detect the tuples or objects which do not belong to groups or clusters in a given data set. For example, there are techniques based on the distance or the angle between the objects. However, these techniques suffer from the curse of dimensionality. This means that for high-dimensional spaces, the objects are by almost equally distant from other object. If the techniques are based on the angles between the objects, then the angle between any two objects is close to 90 degree [3].

Therefore, the curse of dimensionality degrades the performance of certain algorithms for outlier detection. For this reason, alternative strategies are mandatory to process large data sets.

M. Polycarpou et al. (Eds.): HAIS 2014, LNAI 8480, pp. 122–132, 2014.

For other algorithms, the difficulty arises from the computational complexity of the method. Any method with computational complexity larger than linear one results into infeasible for large data sets. In these cases, when incrementing the data size, very large increments in the execution time are produced.

A second source of difficulties arises from the algorithms that analyse the information monolithically. In these cases, the appearance of new tuples forces to perform a new analysis of the complete volume of information. When a large volume of information is involved, this severely impacts over the efficiency of the algorithm. In order to overcome these difficulties, the proposed algorithm performs outlier detection based on a well potential created by each particle in the feature space.

In this algorithm, each tuple is visualized as a particle in a space of N dimensions, being N the number of attributes. Each particle is able to create a squared-well around its position in this N-dimensional space. The well created by each particle is characterized by two parameters: the radius and the depth. All the particles additively contribute to create in the N-dimensional space a potential map. In the positions where more particles concentrate, the potential map has a larger depth.

This model is inspired in the nuclear potential created by nucleons (protons and neutrons) in the nucleus of the atoms. The nucleons in the nucleus create a well potential which bound them together.

The parameters which govern the behaviour of the algorithm are briefly described in the followings:

Radius of the well. This parameter indicates the distance of influence of the presence of the particle. By increasing the radius, distant particles will belong to the same cluster. On the contrary, if the radius is shortened, then more particles will be considered as outliers instead of forming a cluster. The radius gives an notion about the neighbourhood size of a particle.

Depth of the well. This parameter marks the sensitiveness level of the algorithm. If the well formed by a single particle is very depth, then few particles in the neighbourhood of a position will be necessary to form a cluster. Oppositely, if the well is shallow, then more particles in the neighbourhood will be necessary to form a cluster.

Threshold outlier level. This parameter gives to the practitioners the capacity to select the threshold level for outlier-cluster discrimination. The threshold value is subtracted to the final potential in all the feature space to determine which particles form clusters and which are outliers. By defining a low value for the threshold, with the appropriate values for the two former parameters, few particles will be necessary to form a cluster. On the contrary, if the threshold level has a high value, many particles grouped in the neighbourhood of a position will be mandatory to be considered as a cluster.

Although the depth of the well and the threshold outlier level seem to have a similar function, they have been proposed for different purposes. The depth of the well is conceived for the creation of the potential map in the feature space. By varying the radius and the depth, different potential maps are generated. However, this task is considered as computationally intensive. Therefore, for big data, the generation of the potential map is a task which should be performed once.

On the other hand, the threshold outlier level is a parameter that ought to be handled by the practitioner for discriminating clusters and outliers. This task can be performed many times, with different threshold outlier levels, over a previously generated potential map; so that, it provides different sensitiveness in the outlier detection.

This approach provides diverse advantages and potential capacities to process the information in distributed computing infrastructures (grid and cloud computing), by processing the information by parts and later by gathering the partial information pieces; or to process in parallel platform such as clusters or accelerator cards (GPU or Xeon Phi). Besides, this can be executed with horizontal data fragmentation, in subsets of tuples; or with vertical data fragmentation, in subsets of attributes [4]. These advantages are developed in the following points.

- Series of data sets can be compared against a potential map conceived as standard or correct. This allows distinguishing changes in data.
- The potential maps can be compared among them. This comparison allows distinguishing if the clusters or the outliers locations evolve.
- In most of the problems, the data set is composed by a large number of objects with multiple attributes. The approach allows processing the data set by taking into consideration only a sub-set of the attributes, and when finishing the task to continue with the remaining attributes. If some attributes are considered as more important than others, the approach supports to tackle the analysis as a "feature selection" mode. Firstly, the more important attributes are analysed, and the objects marked as outliers based on this initial consideration. Thus, the practitioners can start to extract conclusions from the subset of data analysed. In parallel, the second order of importance attributes can generate their potential maps, and the remaining outliers analysed against these new maps.
- Taking into account that the objective is to process large data sets, the practitioners aim to execute the analysis in high-performance computing platforms (clusters or accelerator cards) or in distributed computing infrastructures (grid or cloud computing). The modularity of the approach allows easily adapting the algorithm to distributed computing infrastructures. Subsets divided by attributes can be processed in different nodes. The execution of the work can be performed in an asynchronous and elastic way (cloud or grid platforms). As far as the partial units are processed, information about the outliers found in this partial processing can be presented as partial conclusions. Moreover, the variation of the number of computational resources available does not impede to process the data set, but it will accelerate or slow down.
- If the main bulk of data but not the complete set is accessible by the practitioners at determined time, the data can be analysed, and the remaining part be incorporated later when possible. This kind of disaggregate analysis is supported by the approach. The potential maps are additive, so that when a set of data are available its potential map can be created, and later be incorporated. The work done with the initial part of the data is not wasted since the potential map does not need to be completely regenerated, only the contribution of the new objects are processed.

The final aim for designing this algorithm is to process the log-files of the cluster Tier-2 at CIEMAT. This infrastructure is devoted to analyse the data of the CMS detector.

This cluster is composed of 1300 cores (roughly 8 cores per node) with 1 PB storage capacity on disk. The analysis of these logs-files might provide clues about fails in the systems, types of job rejected, errors in connections, etc, which currently are concealed in the huge volume of data generated.

The rest of the paper is organized as follows: Section 2 summarizes the Related Work and previous efforts done. An in-depth description of the implementation of the algorithm is presented at Section 3. The Analysis is presented in Section 4, including the asymptotic analysis of the algorithm proposed (Section 4.1) and the study of an MPI implementation (Section 4.2). And finally, the Conclusions and Future Work are presented in Section 5.

2 Related Work

In [3] a brief sub-section describing "depth-based methods" for outlier detection is presented. In this book, the authors express as follows for presenting this type of methods: "The idea is that the points in the outer boundaries of the data lie at the corners of the convex hull". These points in the outer boundaries are identified as likely to be outliers. Although the word "depth-based" is employed in this description of the convex hull algorithm, it is not related at all with the approach presented in the present article. Different meanings to the concept "depth" are given in both works.

Some other studies [5,6] underline that the main drawback of this convex hull-based approach is the large execution time. So that, it makes impractical for processing large data set, and therefore, in big data. Furthermore, by studying the pseudo-code of the algorithm, it is not intuitive how it can be parallelized in order to accelerate the execution.

In [7] an excellent recap of the clustering strategies as well as a discussion about the curse of dimensionality can be found. This is relevant because one of the main drawbacks of outlier detection is the degradation of the performance when incrementing the data size. Other classification of the clustering methods can be found in [8]. Further comparison with the categories presented in these books will be done when describing in-depth the proposed algorithm. In the clustering algorithms, the outlier detection depends on the efficiency of the clustering method, in such way that the outliers are considered as a sub-product of the cluster structure.

Finally, the classifications presented in [4] have been used along this work to categorize both the algorithm and the data fragmentation.

3 Implementation

The schema of the algorithm (Algorithm 1) is as follows: from a zero-level potential map, iteratively for each particle the depth is subtracted in the points inner to the radius of influence of the particle. This process is executed for all feature-space and all particles. The visual result of this process for an one-dimensional example is presented at Fig. 1(a) (continuous line).

At Fig. 1(a), a set of objects (points at horizontal axis) is drawn. Each object generates a squared-well potential of radius 10 and depth 0.1. The dotted horizontal line

Algorithm 1. The outlier detection algorithm pseudocode

```
for Each object in the data set do
    while The distance to the object is lower than the radius do
        Subtract to the potential map, the value of the depth of the well;
    end
end
Once the potential map has been created, add in all the feature space the threshold
outlier level;
for Each object in the data set do
    if The object is below the zero level then
        This object is member of a cluster;
    end
    else
        This object is an outlier;
    end
end
```

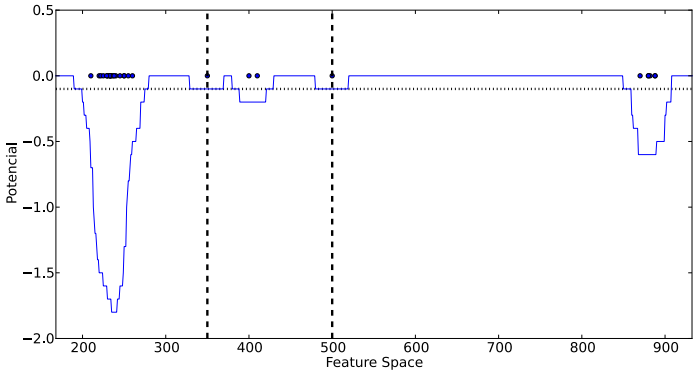
marks the threshold outlier level. This level discriminates the objects which are member of a cluster from the outliers (marked with stripped vertical lines at Fig. 1(a)).

Visually it can be identified the effect over the analysis of the variation of the algorithm parameters. If the absolute value of the threshold outlier level is incremented, then the horizontal line will go down, and as a consequence more objects will be classified as outlier (Fig. 1(b)). As an example, the two points around the position 400 will be classified as outliers if this threshold goes slightly down. In essence, the threshold outlier level marks how many objects in a close area are necessary to be classified as members of a cluster, or on the contrary, as outlier.

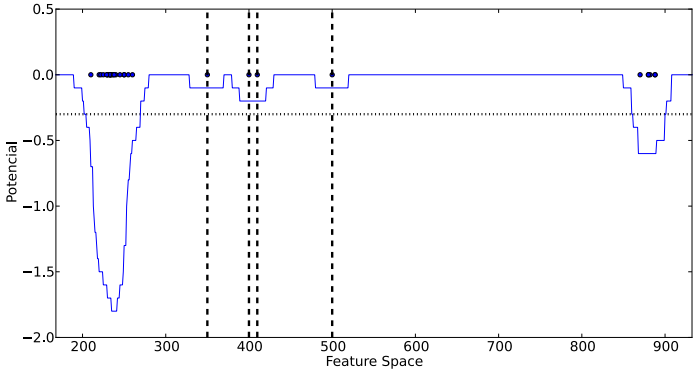
Both the threshold outlier level and the depth of the well are connected. If both parameters are modified in the same direction (both are incremented or both are reduced), then no net effect will be produced. For this reason, it is useful to define the threshold outlier level as a multiple of the depth of the well. Then, it can be visualized as the minimum number of objects in a reduced neighbourhood area to conform a cluster. At Fig. 1(c), the reduction of the radius produces an increment of the objects labelled as outlier. This effect is specially significant in the outer border of the clusters.

Finally, the radius of the well created by an object can visually understood as the size of the area to be considered as the neighbourhood of an object. By enlarging this parameter, scatter objects might be considered as being member of a cluster. Oppositely, by reducing the radius, the objects should be really close to be considered as a cluster.

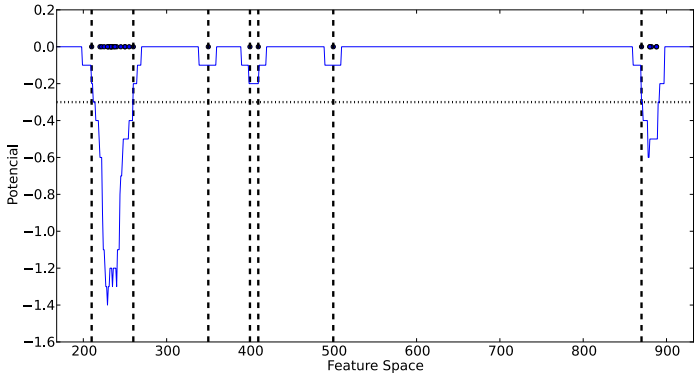
If the problem is computationally intensive (large number of tuples and large number of attributes), then the process can be performed iteratively for each feature. In this case, for one feature and all particles, the one-dimensional potential map is created. Next, all objects are checked against this one-dimensional potential map while retaining the outliers list after the attribute analysed. Once again, iteratively for each feature and the remaining list of outliers, coming from the previous steps, the mentioned procedure is followed, so that, at the end of the process all the features have been checked.



(a) The configuration is: radius = 20; depth =0.1; and threshold outlier level = -0.1.



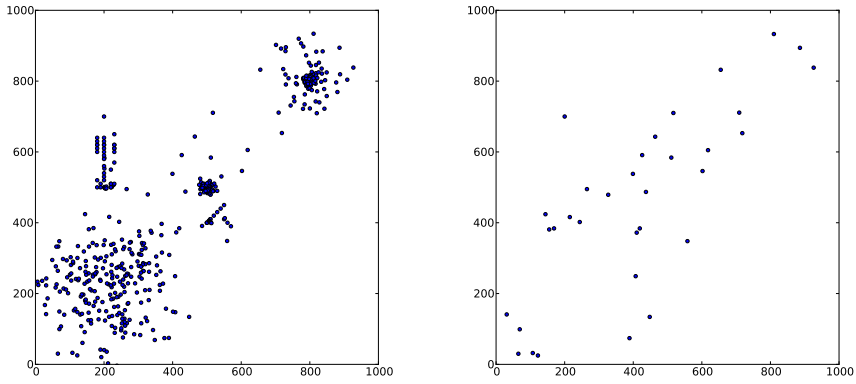
(b) The configuration is: radius = 20; depth =0.1; and threshold outlier level = -0.3.



(c) The configuration is: radius = 10; depth =0.1; and threshold outlier level = -0.3.

Fig. 1. In this plot, the potential map for a set of points (points in x-axis), the threshold outlier level (dotted horizontal line), and the outlier positions (dashed vertical lines) for one-dimensional problem are presented

The algorithm proposed for outlier detection can be roughly categorized as a density-based method. The potential map can be assimilated to a density map in the feature space. Besides, as the proposed algorithm allows searching in subsets of attributes, it can be labelled as subspace clustering method [8]. This type of algorithm is suitable for outlier detection independently of the shape of the cluster.



(a) Raw data. (b) Outliers detected.

Fig. 2. Example of application of the outliers detection algorithm to a two-dimensional data set (600 objects). The configuration employed is: radius = 15; depth =0.2; and threshold outlier level = 0.4.

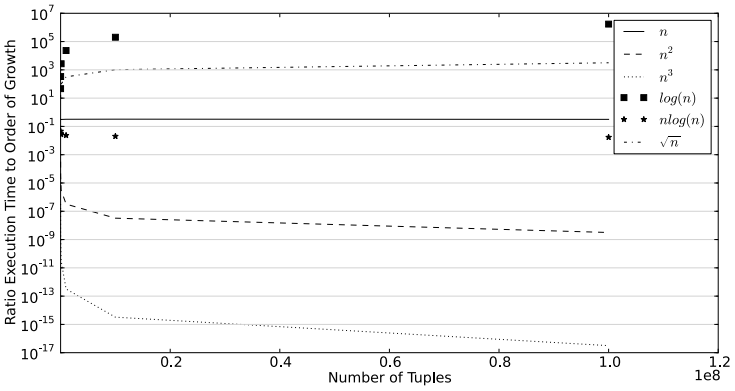
At Fig. 2, an example of data processing for a two-dimensional data set is presented. On the left figure (Fig. 2(a)), the original data set is plotted, whereas the tuples labelled as outliers are plotted at Fig. 2(b). As be can appreciated, both compact and scatter clusters, large and small clusters, spherical and linear are well detected. By varying the configuration of the algorithm, a different number of objects will be marked as outlier.

4 Analysis

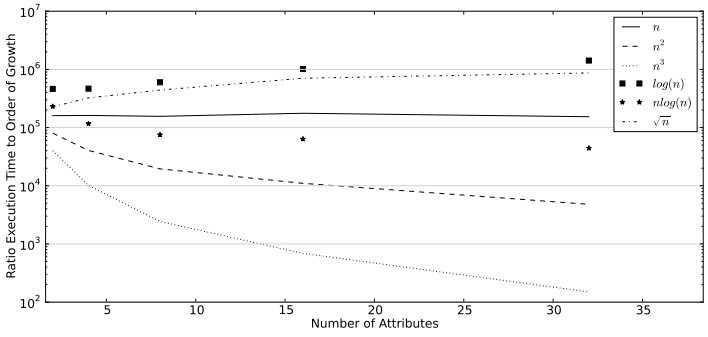
In this section, the proposed implementation is scrutinized in order to in-depth characterize. The numerical experiments of this section have been performed in a cluster composed by 144 nodes with two Quad-Core Xeon processors at 3.0 GHz with 8 GB.

4.1 Asymptotic Analysis

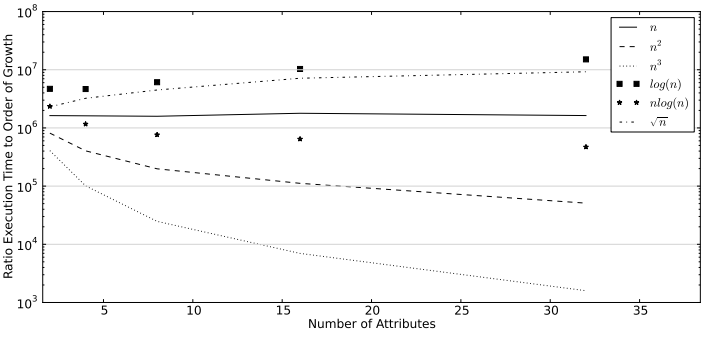
One of the main objectives of this approach is to propose the fastest possible algorithm. This premise was considered in the initial design phase. For this reason, the implementation is tested against an incremental number of tuples (Fig. 3(a) for 2 attributes) and an incremental number of attributes (Fig. 3(b) for 1 million objects, and Fig. 3(c) for 10 million objects). For each configuration, 20 executions are performed and the worst execution time divided by the varying parameter (tuples or attributes) is plotted.



(a) Asymptotic behaviour (worst execution time divided by the number of tuples) of the algorithm when incrementing the number of tuples (two attributes).



(b) Asymptotic behaviour (execution time divided by the number of attributes) of the algorithm when incrementing the number of attributes (1M objects).



(c) Asymptotic behaviour (worst execution time divided by the number of attributes) of the algorithm when incrementing the number of attributes (10M objects).

Fig. 3. Asymptotic analyses of the proposed algorithm for outlier detection. The worst execution time of 20 runs is employed. The plots demonstrate that the computational complexity is $O(tuples \times attributes)$.

As can be appreciated at Fig. 3, the algorithm behaves linearly with the number of the tuples and attributes in the range analysed. Therefore, it can be concluded that the most adequate computational complexity for the proposed algorithm is $O(tuples \times attributes)$. This behaviour is very suitable for processing big data in an affordable execution time.

Additionally to the plot presented at Fig. 3, the execution time of other large input sizes was checked. From tiny data set with 1,000 objects and 2 attributes, to data sets with 100 million of objects and 16 attributes (23 GB), the same number of objects with 8 attributes (12 GB) or 4 attributes (5.6 GB); or 10 million of objects with 32 attributes (4.5 GB), the execution time is measured and divided by the number of objects and attributes. The aim of this calculation is to check the variations in the unitary execution time per information piece (objects and attributes). The proposed algorithm for outlier detection maintains a ratio $\frac{Execution\,time}{Objects \times Atrributes}$ in the range [0.15–0.18] for the inputs processed. This stability in the unitary execution time is an excellent feature when intending to process large data sets. As a consequence, the execution times[1] keeps affordable even for large input sizes: 74 hours for the 23 GB input size, 36.8 hours for the 12 GB, 17.7 hours for the 5.6 GB, and 14.5 hours for the 4.5 GB input size.

4.2 Parallel Implementation

The two data fragmentation models produce two different parallel model. Firstly, the objects can be distributed among all the nodes. Iteratively the nodes process the attributes of the objects assigned. This model has been followed in the implementation described in this section. A second implementation stems from the fragmentation of the attributes. Each node processes one single attribute of all the objects.

In order to test the behaviour of the proposed algorithm when producing a parallel implementation, an MPI implementation is coded, and the execution time tested for diverse number of objects and attributes, and number of cores (Table 1). As can be appreciated in this survey, the algorithm behaves differently in function of the problem size, and number of nodes used to process it.

In general, the speedup obtained is close to the theoretical maximum speedup (number of nodes) provided that a large volume of data is supplied to each node. When distributing the data among too much nodes or the input size is not large, the time of the data transference becomes relevant compared with the processing time in the node, and, as a consequence, the performance degrades. This degradation is clearly observed when using 16 or more than 16 nodes for analysing the input with one million of objects and up to 8 attributes.

Without considering the cases where the speedup degrades appreciably, the algorithm is able to process efficiently files in the order of GB with a limited execution time budget and by using a humble number of nodes. For example, the sample of 10^7 objects and 16 attributes has a size of 2.3 GB, and it can be analysed in less than one hour in 8 cores.

[1] The worst execution time after 20 runs is employed.

Table 1. Mean execution times in seconds (20 executions) and speedup of the MPI implementation of the outlier detection algorithm for diverse input sizes: number of tuples from 10^6 to 10^7 and number of attributes from 2 to 16

Number of Nodes	Number of Objects			
	10^6		10^7	
	Execution Time	Speed-up	Execution Time	Speed-up
2A				
1	319.17		3099.46	
2	167.09	1.91	1560.92	1.99
4	90.56	3.52	791.85	3.91
8	53.13	6.01	408.75	7.58
16	34.76	9.18	218.66	14.17
32	25.96	12.30	136.20	22.76
64	23.55	13.55	88.42	35.05
4A				
1	631.32		6220.18	
2	326.40	1.93	3138.17	1.98
4	173.02	3.65	1598.82	3.89
8	94.82	6.66	805.38	7.72
16	57.03	11.07	421.27	14.77
32	37.57	16.81	234.24	26.56
64	30.46	20.73	142.44	43.67
8A				
1	1285.33		12368.90	
2	648.60	1.98	6206.87	1.99
4	333.40	3.86	3130.69	3.95
8	175.36	7.33	1592.99	7.76
16	99.15	12.96	830.10	14.90
32	50.81	25.30	457.38	27.04
64	42.29	30.39	274.92	44.99
16A				
1	2530.92		25961.52	
2	1313.64	3.71	12532.01	2.07
4	682.46	3.71	6500.17	3.99
8	342.07	7.40	3208.67	8.09
16	180.78	14.00	1602.08	16.20
32	108.00	23.44	895.33	29.00
64	71.29	35.50	522.27	49.71

5 Conclusions and Future Work

In this paper, a new approach to detect outliers based on squared-well has been proposed. This approach has been designed to be able to deal with large data sets, at the same time that it keeps an affordable execution time. Besides, in the design phase, it has been taken into account to be suitable for diverse computational infrastructures.

The proposed approach is fully additive. This allows disaggregating the calculation by objects and by attributes. This property eases the adaptation of the algorithm to distributed computational infrastructures, such as: cloud and grid; as well as to parallel platforms, such as: cluster and GPU.

Furthermore, the approach has demonstrated that the sequential implementation behaves linearly with the number of objects and the number of attributes, $O(objects \times attributes)$, which is an excellent feature when intending to process large data sets. For testing the approach, artificial very large data sets (in the order of GB input size) have been successfully processed while maintaining a limited execution time budget.

Additionally to the sequential version, a parallel implementation (MPI) has been created and tested. The speedup achieved for large data sets is close to theoretical maximum speedup. This demonstrates that the approach is able to deal with these large data sets.

As future work, more comparative works are proposed. On the one hand, the approach will be tested for other computational platforms, preferentially cloud and GPU. On the other hand, other data sets are considered for comparing with other algorithms devoted to outlier detection. Finally, the aim of this approach is to process the log-files of the cluster Tier-2 at CIEMAT for processing the data of CMS detector. So that, once the approach has been successfully tested against artificial data sets, to analyse the data sets generated by the CMS Tier-2 infrastructure.

Acknowledgement. The research leading to these results has received funding by the Spanish Ministry of Economy and Competitiveness (MINECO) for funding support through the grant FPA2010-21638-C02-02, together with the European Community's Seventh Framework Programme (FP7/2007-2013) via the project EGI-InSPIRE under the grant agreement number RI-261323.

References

1. Corchado, E., Wozniak, M., Abraham, A., de Carvalho, A.C.P.L.F., Snásel, V.: Recent trends in intelligent data analysis. Neurocomputing 126, 1–2 (2014)
2. Abraham, A.: Special issue: Hybrid approaches for approximate reasoning. Journal of Intelligent and Fuzzy Systems 23(2-3), 41–42 (2012)
3. Aggarwal, C.C.: Outlier Analysis. Springer (2013)
4. Peteiro-Barral, D., Guijarro-Berdiñas, B.: A survey of methods for distributed machine learning. Progress in AI 2(1), 1–11 (2013)
5. Johnson, T., Kwok, I., Ng, R.T.: Fast computation of 2-dimensional depth contours. In: Agrawal, R., Stolorz, P.E., Piatetsky-Shapiro, G. (eds.) Proceedings of the Fourth International Conference on Knowledge Discovery and Data Mining (KDD 1998), pp. 224–228. AAAI Press (1998)
6. Struyf, A., Rousseeuw, P.J.: High-dimensional computation of the deepest location. Computational Statistics and Data Analysis 34, 415–426 (1999)
7. Rajaraman, A., Ullman, J.D.: Mining of massive datasets. Cambridge University Press, Cambridge (2012)
8. Han, J., Kamber, M.: Data Mining: Concepts and Techniques. Morgan Kaufmann (2000)