Unsupervised machine learning algorithms as support tools in molecular dynamics simulations.

Daniela Rim[1], Luis G. Moyano[2], and Emmanuel N. Millán[2,3]

[1] Facultad de Ciencias Exactas y Naturales, Universidad Nacional de Cuyo, Mendoza, Argentina,
rim.dan96@gmail.com,
[2] Consejo Nacional de Investigaciones Científicas y Técnicas, CONICET, Mendoza, Argentina
[3] ITIC, Universidad Nacional de Cuyo, Mendoza, Argentina.

Abstract. Unsupervised Machine Learning algorithms such as clustering offer convenient features for data analysis tasks. When combined with other tools like visualization software, the possibilities of automated analysis may be greatly enhanced. In the context of Molecular Dynamics simulations, in particular asymmetric granular collisions which typically consist of thousands of particles, it is key to distinguish the fragments in which the system is divided after a collision for classification purposes. In this work we explore the unsupervised Machine Learning algorithms k-means and AGNES to distinguish groups of particles in molecular dynamics simulations, with encouraging results according to performance metrics such as accuracy and precision. We also report computational times for each algorithm, where k-means results faster than AGNES. Finally, we delineate the integration of these type of algorithms with a well-known analysis and visualization tool widely used in the physics community.

Keywords: machine learning, unsupervised algorithms, molecular dynamics, granular collisions

1 Introduction

Machine Learning provides powerful analysis tools in particular when large amounts of information are involved, as it provides algorithms capable of organizing, summarizing, or finding patterns in vast data volumes. One of the scientific disciplines highly benefited by Machine Learning is Physics and its broad variety of branches, e.g. it has recently been applied to the study of phases and phase transitions in quantum spin models [3, 11] to the prediction of solar flares [7], to measuring masses of galaxy clusters [17] in astrophysics, among others.

In astrophysical contexts, collisions between mass-asymmetric granular aggregates have important applications as they may lead to accumulative growth due to particle agglomeration and may therefore explain certain processes at early stages of planet formation [1], cometary comae [16] or debris disks [9]. To

analyze the fragmentation rate of such granular systems, an efficient simulation software for molecular dynamics modeling, LAMMPS, was developed by Ringl et al. [19] and later ported to GPUs by Millán et al. [8]. This implementation takes into account complex physical inter-cluster particle interactions.

Using this software, collision of micro-metric dust grains are simulated with the aim of understanding the speed limit that separates an agglomeration process from a fragmentation process. Hence, it is essential to determine when and how the system is divided after the collision, which is in strong relation with the general physical parameters determining the simulated sample. Due to the importance of obtaining collision outcomes from such broad variety of parameters, computing time should be reduced for statistical purposes. In most cases, the total simulation time is subjectively chosen by the analyst, as no theoretical information states beforehand the total time steps in which the system will be divided into fragments. Even with HPC resources such as parallel computing with multiple CPU cores or GPUs accelerators [14], an asymmetric collision consisting in more than 121000 particles may take 3 months to be simulated to the point of clearly reaching a distinguishable fragmentation or aggregation state.

As an alternative that could be used for anticipating the identification of such states, Machine Learning algorithms could be integrated to the analysis workflow to process the information obtained at each simulated time step as soon as it is generated, describing general characteristics and providing information to the expert about the possible final configuration of the system long before the simulation concludes.

For such purposes, Machine Learning offers a wide range of algorithms, usually grouped as supervised or unsupervised learners, which are extensively used for -or adapted to perform- classification tasks [2, 4, 6]. For instance, a recent publication by Ceriotti [5] acknowledges the importance of automated analyses of the outcome of a simulation and summarises some of the unsupervised machine learning methods that are aimed toward classification of molecular simulations. These ideas are enhanced even more by applications for statistical purposes and overall improvement of workflow efficiency, as exampled in [20], where 86 billion amino acids without labels are given to the unsupervised algorithms to recover information about protein structure.

Unsupervised algorithms are generally used for identifying meaningful groups among data, summarizing or giving a compressed version of them. This work will be focused around one type of unsupervised algorithms known as clustering algorithms, which are subdivided into two groups: partitioning clustering and hierarchical clustering algorithms. Both of them are used to sort observations, within a data set, into multiple groups based on their similarity. The main general difference is that in partitioning clustering the analyst must pre-specify the number of clusters to be generated, while hierarchical clustering does not require it.

In this work, we build an integrated tool with unsupervised clustering algorithms to process data in lockstep with a given type of molecular dynamics simulations. The clustering algorithms should assist the analyst in the interpre-

tation of the physical system, as they are capable of identifying an agglomeration or fragmentation of the aggregates while the simulations are being executed. For this purpose, a comparison between a partitioning clustering (k-means clustering) and a hierarchical algorithm (Agglomerative Nesting or AGNES) is presented.

2 Materials and Methods

In this section we first discuss the basics of the two unsupervised algorithms used in this work, k-means and AGNES. Then, we describe the molecular dynamics simulations that were used to perform the analysis.

2.1 Unsupervised Machine Learning Algorithms

K-means clustering (MacQueen, 1967) [13] is one of the most frequently used unsupervised clustering algorithms. The main idea of this method is to cleave data points into k pre-specified groups (clusters) to minimize the total within-cluster variation. Each cluster is represented by a centroid, corresponding to the mean of data points assigned to the cluster. The standard k-means method is the Hartigan-Wong algorithm [10], which defines similarity (variation) in terms of distances (Euclidean or others) between the data points x_i and their cluster's centroid μ_j. [12].

The total within-cluster variation ($TWCss$) to be minimized is defined as follows:

$$TWCss = \sum_{j=1}^{k} \sum_{\boldsymbol{x}_i \in C_j} (\boldsymbol{x}_i - \boldsymbol{\mu}_j)^2, \tag{1}$$

where C_j ($j \in \{1, ..., k\}$ is one of the k clusters, x_i are the data points belonging to cluster C_j, represented by centroid μ_j .

On the other hand, Agglomerative Nesting (AGNES) is also a well-known hierarchical algorithm for data clustering [12]. It starts by treating each data point as a singleton cluster and successively merges them into larger clusters given their similarities, until all data is classified into one group. In this case similarity is quantified by Euclidean distances. These distances are arranged in an n-dimensional matrix, where n accounts for the number of predictive variables, and a linkage method is responsible for joining together the data points by their proximity.

Several linkage methods exist depending on how the distances between clusters are calculated. Some of them are briefly summarized below:

- **Complete linkage:** The inter-cluster distance is defined as the maximum value of all pairwise distances between the elements in different clusters.
- **Single linkage:** The inter-cluster distance is defined as the minimum value of all pairwise distances between the elements in different clusters.
- **Average linkage:** The distance is defined as the average distance between the elements in different clusters.

– **Centroid linkage:** The distance is defined as the distance between the centroids (i.e., the same definition as in k-means clustering) of different clusters.
– **Wards minimum variance method:** It minimizes the total intra-cluster variance. At each step the pair of clusters with minimum inter-cluster distance are merged.

The expert will decide which one of the linkage functions is more suitable for the case under consideration.

2.2 Molecular Dynamics simulations

The simulations to be analyzed were granular simulations of porous grains collisions, which were provided by SIMAF's Astrophysics and Cluster Impact researchers [4]. These are Molecular Dynamics simulations, each system composed of a N number of identical SiO_2 particles with $R_g = 0.76\mu m$ granular radius. All simulations consist of two objects composed by grains or particles, a smaller spheroid (the projectile) with initial velocity v_0 along the z-axis, and a larger spherical aggregate (the target), initially at rest. For the value range of the general parameters used in these simulations, once the projectile hits the target, the system almost always ends up divided in two parts: one carried along by the projectile and another one that remains immobile (Fig. 1).

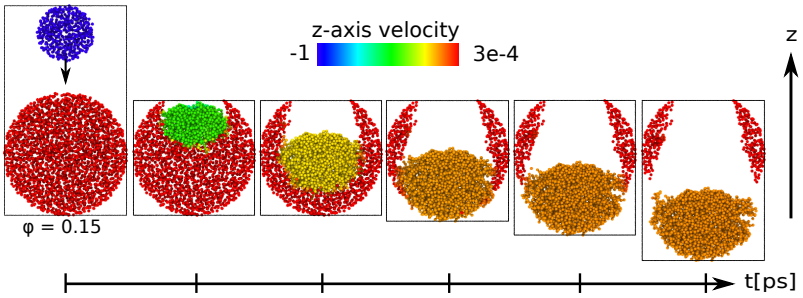

Fig. 1. Example of the temporal development of a collision's simulation. It is shown only a thin z-parallel slice of the system, and the particles are coloured by their vertical (z) velocity.

There are three main general parameters being considered related to the system's evolution and final outcome: the projectile/target size, the filling factor ϕ which is related to the porosity of the sample, and the projectile's initial velocity v_0. The filling factor ϕ is defined as the density of particles inside a sphere of radius $N * R_g$ (N particles times its radius), centered in particle i [18]. This quantity defines how much volume is occupied in both spheroids (projectile and target). The number of grains in each simulation depends on the size of the

[4] `https://sites.google.com/site/simafweb/home`

spheroids and the filling factor. The values used in this work are specified in the table 1.

Table 1. Number of particles for each filling factor ϕ. The initial projectile velocities v_0 used for each ϕ are $0.1\frac{m}{s}$, $0.25\frac{m}{s}$, $0.5\frac{m}{s}$, $0.75\frac{m}{s}$ and $1\frac{m}{s}$.

ϕ	N_{part}
0.15	11200
0.25	18785
0.35	26206

The granular simulations use input data generated with a model presented in [19], adapted by members of SIMAF. The simulations were executed using LAMMPS (http://lammps.sandia.gov) in GPUs (Graphics Processing Unit). The granular pair style used was ported by Millán et al. [8] to run in NVIDIA GPUs with CUDA from the original one presented by Ringl et al [19].

The simulations were configured to generate one dump file every 10000 steps. A dump file is a text file (with a format similar to the *comma separated values* extension *.csv*) containing the complete configuration of the simulated system at the given time step, which includes information about positions and velocities of each grain. The number of dumps files generated varies between the different general parameters of the simulations. The main idea of this work is to process each of these dump files with a clustering algorithm immediately after they are generated.

First, each dump file was filtered to only use some input variables to facilitate its processing, such as velocity component in the **z** axis and *particle ID*. For both k-means and AGNES it was specified that 2 clusters should be identified. The algorithm processed the data, classifying each particle to a certain cluster by its **z**-axis velocity. For this type of simulations, the main interest is finding the amount of target grains that remain still and the number of grains that agglomerate to the ones belonging to the projectile. In all the simulations made (more than 50 simulations with different input parameters) the outcome was similar, only two clusters of grains can be identified after the collision.

For the AGNES algorithm the distance matrix was calculated, also based on the **z** velocity of each particle. Then, clusters were formed by successively grouping together particles in accordance to Ward's method. In this way, a tree-like structure is formed. The algorithm was set to return the two biggest clusters, that is, the penultimate grouping made by AGNES. This analysis was done independently for 100 equally time-spaced dump files for each simulation. The AGNES linkage function used for this analysis is Ward's minimum variance method [15].

Because these simulations were prototypes, they were actually executed until the final two-cluster state was clearly distinguishable by inspection. Thus, the

actual labels for each particle are known, enabling us to compare them with the two clustering outputs (k-means and AGNES) and thus calculate performance metrics such as accuracy and precision, as if the algorithms were performing a classification task. It is necessary to note that the final particles' labels are only used in this work to quantify the performance of the algorithms but are not a requisite for the actual simulation analysis.

2.3 Predictor Variable Selection

The selection of the predictor variable to perform the classification requires an exploration of the data that has to be classified. To aid in the selection of this variable, we use the Ovito[5] analysis and visualization software (extensively used in the physics community). We first observed the features of the simulations:

1. The projectile collides the target, initially at rest in all simulations, with a downward velocity $\boldsymbol{v_0}$.
2. As it penetrates the target, the projectile drags some of its particles downwards until they detach completely from the target. This results in a two-clustered system.
3. The top cluster (cluster 1) has only particles that belonged originally to the target, which remain almost steady throughout the collision.
4. The bottom cluster (cluster 2) has an overall downward velocity v_z which is relatively larger than zero in magnitude, and it's composed by all projectile particles and the target particles dragged by them.

The output files written by LAMMPS (the Molecular Dynamics software) have 11 variables for each grain: **id** unique identification of each grain (fixed for all the simulations steps), **type of particle** (1 for the projectile and 2 for the target at the beginning of the simulation), position coordinates in **x, y, z** for each grain, velocity coordinates in **x, y, z** and angular velocity coordinates in **x, y, z**. As it can be seen in Fig. 1, one of the features that allows to clearly identify the two clusters is the velocity in the **z** direction. No other variables could portray better this classification, as the grains have almost no displacement in the **x** and **y** directions, which means that **x** and **y** positions and velocities in the same directions have nearly no change at all during the entire simulation. Taking this into account, we identified that the velocity in the **z** direction was the best predictor variable to perform the classification.

3 Results and Discussion

In this section we first discuss the hardware and software used to execute the simulations and the classification analysis. Then, the performance of the k-means and AGNES clustering algorithms is shown and a comparison between both is performed for a set of 15 simulations. Additionally, we show the results of the

[5] `https://www.ovito.org/`

computational cost of executing k-means and AGNES. Finally, we propose to integrate these algorithms in a software heavily used by physics community to aid in the classification process and the interpretation of the results.

3.1 Hardware and Software Description

The Molecular Dynamics simulations and the Machine Learning algorithms were executed in the following hardware and software infrastructure:

- Workstation FX-8350 with: AMD FX-8350 8 cores running at 4 GHz and 32 GB of DDR3 RAM memory. MD simulations were executed with one NVIDIA GeForce GTX Titan X (Maxwell GM200 architecture) with 12 GB of memory. Running Slackware Linux 14.2 64 bit OS with kernel 4.4.14, OpenMPI 1.8.4, GCC 5.3.0, R language version 3.5.1 and CUDA 6.5.
- Cluster Toko at Universidad Nacional de Cuyo: one node with two AMD EPYC 7281 CPU, with 16 CPU cores at 2.1GHz each (32 cores total), 128 GB of RAM and Gigabit Ethernet. Running Slackware Linux 14.1 64 bit OS with kernel 4.19.23, OpenMPI 1.8.8, GCC 4.8.2 and R language version 3.5.1.

The unsupervised algorithms were executed with an R language script (available at `https://github.com/machine-learning-and-physics/jaiio2019`) using various packages: *dplyr* and *plyr* to manipulate input data, *caret* to calculate the confusion matrix, *fastcluster* and *parallelDist* to compute the AGNES algorithm, *tictoc* to benchmark portions of code, *stats* to compute kmeans, and *doParallel* and *foreach* to perform the analysis of different input data in parallel to improve compute time.

3.2 Comparison between k-means and AGNES

One of the main outputs that k-means and AGNES return is the number of particles labeled as belonging to one of the two clusters they are set to recognize. In Fig. 2, we show the proportion of particles (relative to the total number of particles) classified as part of cluster 2, which is the one that after the collision is placed below cluster 1, with non-zero velocity. We find that both k-means and AGNES present an almost identical particle recognition. For the dump files corresponding to earlier times in the collision, both algorithms label all projectile particles as belonging to cluster 2, due to their initial velocity v_0. For dumps corresponding to timesteps when the projectile penetrates the target, more particles are labeled as belonging to cluster 2, as they gain energy and are dragged downwards. For dumps corresponding to late simulation time steps, both algorithms properly recognize all particles.

From Fig. 2 it can be inferred that the system is most likely fragmented (divided into two distinguishable well-formed clusters) for lower filling factors, while almost every particle agglomerates with the projectile for larger ones.

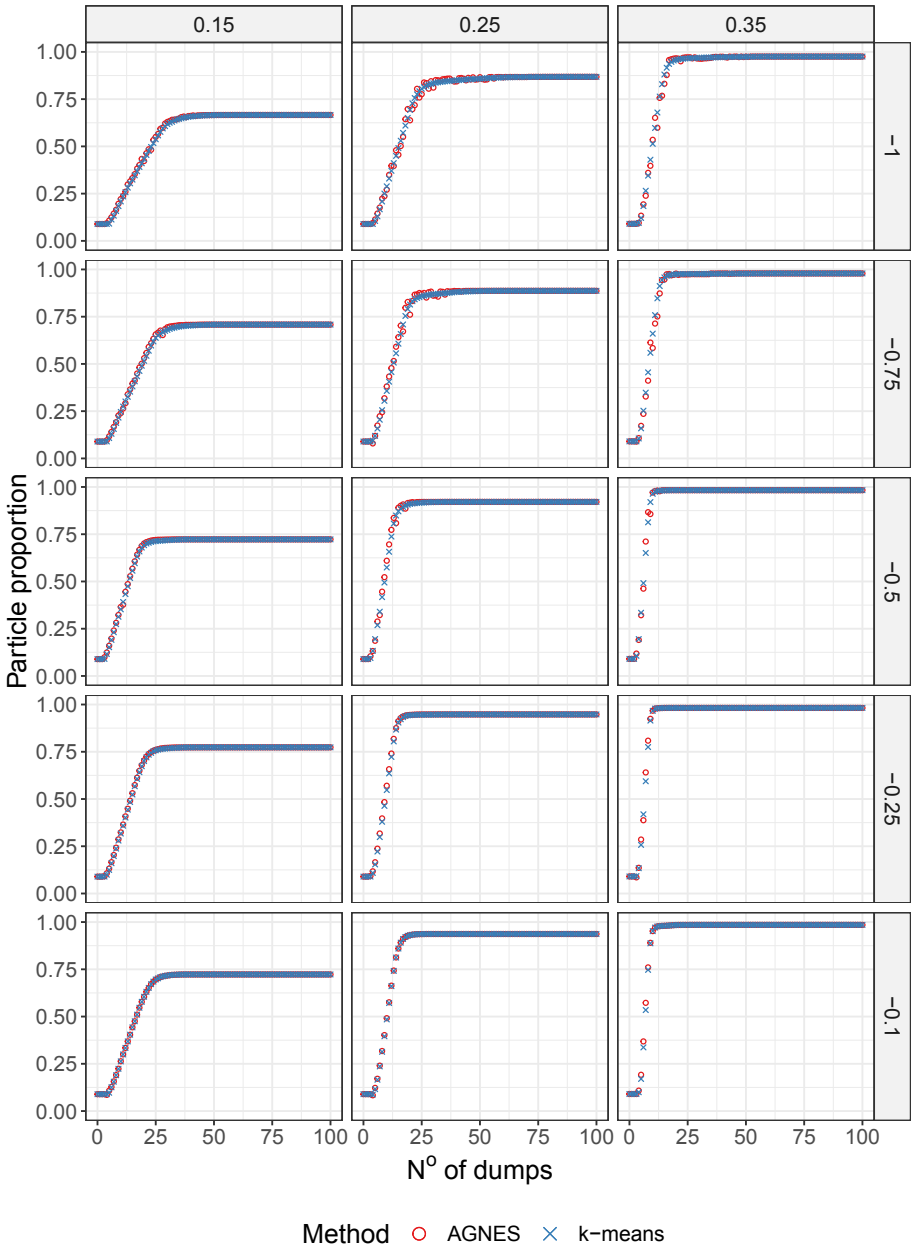

Fig. 2. Comparison between the number of particles that the algorithms classify as belonging to cluster 2 (the particles drawn by the projectile) relative to all particles, in each of the 100 dump files, for 15 simulations with different filling factors (0.15, 0.25. 035) and different initial projectile velocities.

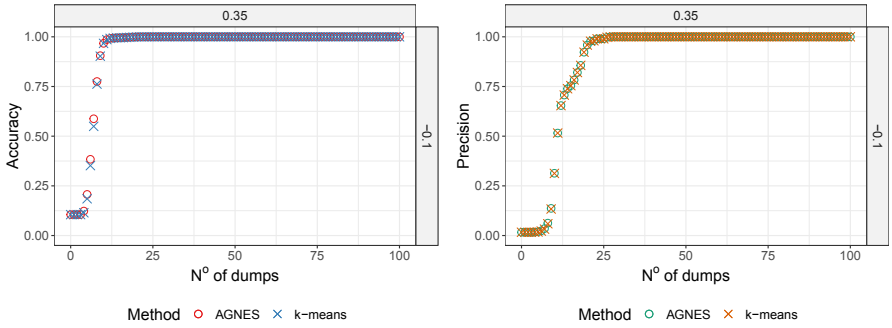

Fig. 3. Example of a simulation's accuracy and precision performance by k-means and AGNES per dump file. The simulation has a filling factor ϕ of 0.35 and an initial projectile velocity v_0 of $-0.1\frac{m}{s}$.

3.3 Algorithm's labeling performance

We measured the accuracy and the precision of the classification performed by k-means and AGNES per output file by comparing it with the true particle labels. An example of the evolution of both metrics for a particular simulation is shown in Fig. 3. The accuracy has its lower values within the first 10 dumps, as the projectile has yet not collided with the target, and the system's configuration is the most disparate with the final one. As the collided particles acquire their final velocity (future cluster 1 particles remain steady, while the rest gain an uniform non-zero v_z velocity), the classification becomes more accurate, reaching its maximum once the system stabilizes to its final values.

On the other hand, the precision of the metrics follows a similar evolution. There is a little fluctuation of its values before it stabilizes, showing that the classifications are not as precise even though an accuracy of 100% has been reached.

The 15 simulations analyzed exhibited this same behavior, and overall no significant differences where found between k-means and AGNES correct predictions.

Some minor differences between k-means and AGNES classifications' accuracies in some dump files appear, specially for the first half of each simulations. We show in Fig. 4 the percentage for which AGNES was more accurate than k-means (positive values) and the opposite (negative values). These variations between accuracies over each dump do not surpass a 10% value, and are larger for bigger filling factors and initial projectile velocities magnitudes.

3.4 Computational performance

In this section we discuss the performance implications of running k-means and AGNES algorithms in the hardware infrastructure detailed in subsection 3.1. The k-means algorithm is considerably less resource intensive than AGNES. To

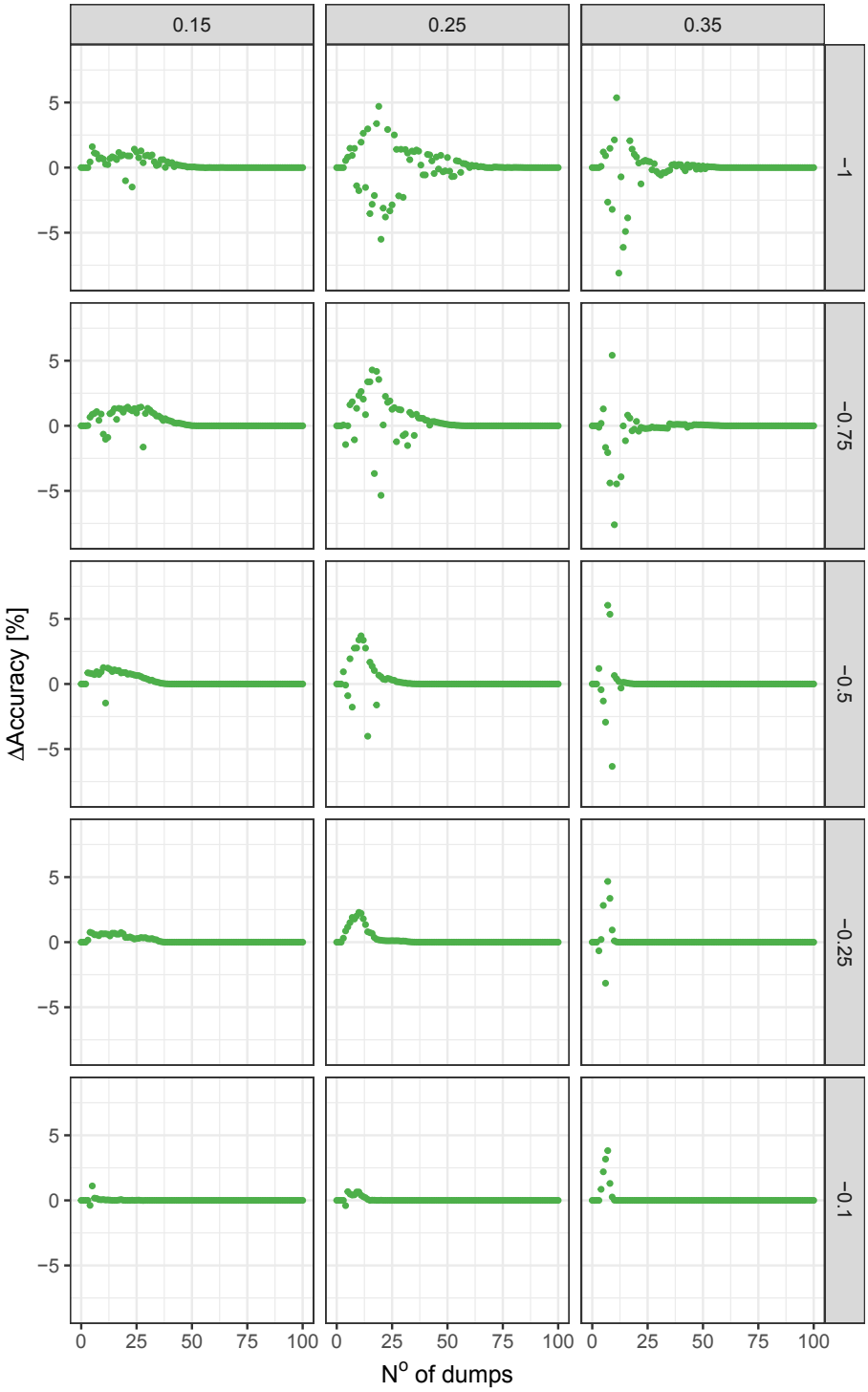

Fig. 4. δAccuracy, proportion of particles properly classified by AGNES when compared to k-means. Positive values refer to better accuracy from AGNES over k-means, and negative values for the opposite.

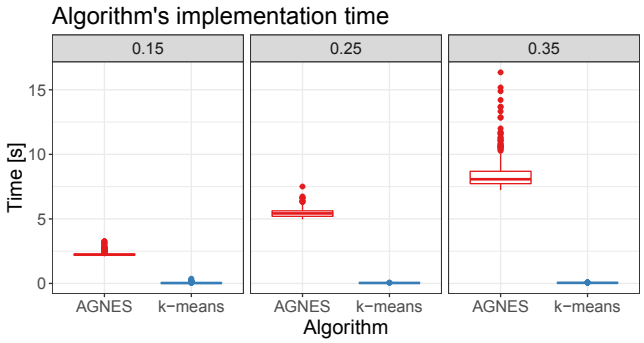

Fig. 5. Compute time in seconds of the k-means and AGNES algorithms for each dump file, executed in the Epyc node of the Toko cluster.

be able to analyze one dump file from the simulation, k-means needs $\sim$ 135 MB of RAM, the AGNES algorithm (from the *fastcluster* package) uses an average of 600 MB of RAM (both for 11200 grains and filling factor value set to 0.15). AGNES algorithm uses up to 4.4 times more RAM than k-means.

In Fig. 5 we show the computational time that each algorithm consumes for the three filling factors. K-means is considerable faster than AGNES, in some cases up to two orders of magnitude faster.

Another fundamental step of the automated analysis is to read each dump file generated by the LAMMPS software used to perform the simulations. In the FX-8350 workstation with an standard hard drive, the time it takes R to read each file is on average 0.2 ± 0.048 seconds.

Even though there were some dump files where AGNES was more accurate than k-means, the latter had an overall better performance regarding computing time. Unfortunately, for larger system sizes than the ones specified in Table 1 (for example, more than 31000 particles), k-means has yield incorrect particle classifications whereas AGNES hasn't (Fig. 6).

3.5 Integration of unsupervised algorithms in analysis tools

One of the tools extensively used by the physics community in order to perform visualization and analysis of Molecular Dynamics simulations is an open-source software called Ovito [21]. This tool has different *modifiers* that enable the analyst to manipulate each output file obtained from a Molecular Dynamics simulation such as the ones used in this work. For instance, these *modifiers* are able to change color, select and filter particles with different criteria, slice and analyze the particles with various algorithms, among other transformations useful for the analysis of the simulations. Currently, this tool does not include Machine Learning algorithms.

Ovito allows users to add *modifiers* to enhance the software analysis or visualization features. Adding unsupervised algorithms such as the ones used in

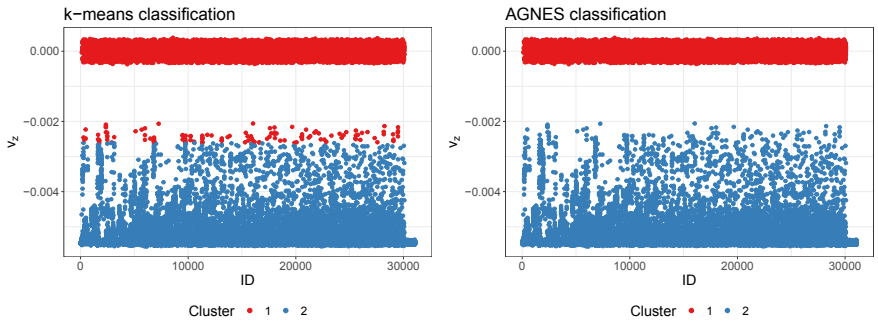

Fig. 6. Comparison between a miss-classification performed by k-means and the same dump file of the same simulation ($\phi = 0.15$, $v_0 = 0.1\frac{m}{s}$, 31087 particles) classified by AGNES.

this work as modifiers would allow all Ovito users to take advantage of these tools, enabling them to extract more information and improve cluster analysis in an integrated fashion, without the need to perform the analysis outside the software. Additionally, the amount of time that takes to run k-means or AGNES algorithms is within the time that takes to run most Ovito modifiers. For these reasons, we are taking steps forward to include these algorithms in the Ovito software.

Figure 7 shows three different simulations colored by cluster identification performed by our R script using the AGNES algorithm. After the analysis is performed, our R script writes a new output file in LAMMPS format, adding a new column with the cluster identifier for each grain.

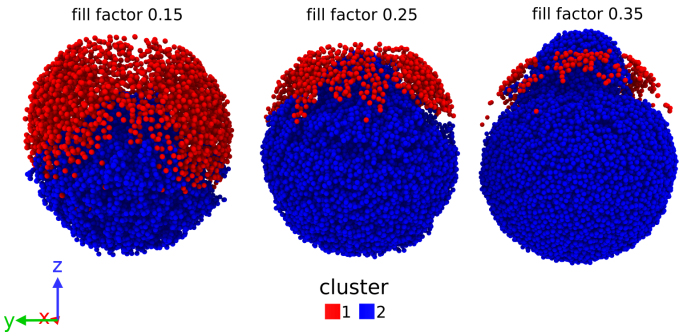

Fig. 7. Ovito visualization of the classification performed by the AGNES algorithm for three different simulations (for three filling factors with $v_0 = 0.1m/s$) in the same instance of dump file. Color indicates the cluster classification performed by the algorithm.

4 Conclusions and Future Work

Overall, k-means and AGNES were able to classify the particles in a similar way. Both algorithms were precise, and even though AGNES was on average a little more accurate than k-means, k-means computation time was relatively better. Besides, k-means needs approximately a quarter of the RAM used by AGNES. Out of the two proposed clustering algorithms, the results suggest k-means as the best suited algorithm for this classification task. However, AGNES cannot be discarded, as it correctly recognizes well delimited clusters when larger system sizes than the ones specified are analyzed, whereas k-means fails to do so.

Despite this, the results were encouraging as possible Ovito modifiers, and an integrated tool to help analyst performing this kind of simulations is a very feasible possibility.

As future work we will further explore the possibility of saving computational time via predictive analysis, i.e. identifying the fragmentation outcome as earliest as possible in the simulation run. The main idea is to test algorithms that could be used to predict collision outcomes before they actually occur in the simulation. With this information (the amount of grains present in each cluster), the analyst could decide not to continue the simulation potentially saving substantial computational time. Furthermore, along the lines described in this work, we plan to develop a fully integrated mode for particle classification and outcome prediction using supervised Machine Learning algorithms. In this line, we are currently working with the Support Vector Machine and the Random Forest families of algorithms.

Acknowledgments

We thank the support provided by Belen Planes and Eduardo Bringa of the SiMaF group for the molecular dynamics simulations used in this work. We acknowledge support from CONICET and SIIP UNCuyo grants. LGM acknowledges support from PICTO- 2016-0054 UNCuyo-ANPCyT. This work was performed in the Toko Cluster from FCEN-UNCuyo, that is part of SNCAD, MinCyT, Argentina.

References

1. Armitage, P.J.: Astrophysics of planet formation. Cambridge University Press (2010)
2. Banerji, M., Lahav, O., Lintott, C.J., Abdalla, F.B., Schawinski, K., Bamford, S.P., Andreescu, D., Murray, P., Raddick, M.J., Slosar, A., Szalay, A., Thomas, D., Vandenberg, J.: Galaxy zoo: reproducing galaxy morphologies via machine learning. Monthly Notices of the Royal Astronomical Society 406(1), 342–353 (jul 2010)
3. Broecker, P., Assaad, F.F., Trebst, S.: Quantum phase recognition via unsupervised machine learning. arXiv (2017)

4. Carrasquilla, J., Melko, R.G.: Machine learning phases of matter. Nature Physics 13(5), 431–434 (feb 2017)
5. Ceriotti, M.: Unsupervised machine learning in atomistic simulations, between predictions and understanding. The Journal of Chemical Physics 150(15), 150901 (apr 2019)
6. Ch'ng, K., Vazquez, N., Khatami, E.: Unsupervised machine learning account of magnetic transitions in the hubbard model. Physical Review E 97(1) (jan 2018)
7. Colak, T., Qahwaji, R.: Automated solar activity prediction: A hybrid computer platform using machine learning and solar imaging for automated prediction of solar flares. Space Weather 7(6), n/a–n/a (jun 2009)
8. E. N. Millán, Ringl, C., Bederián, C.S., Piccoli, M.F., Garino, C.G., Urbassek, H.M., Bringa, E.M.: A gpu implementation for improved granular simulations with lammps. In: HPCLatAm 2013. pp. 89–100 (2013), http://hpc2013.hpclatam.org/papers/HPCLatAm2013-paper-10.pdf
9. Gáspár, A., Rieke, G.H., Balog, Z.: The collisional evolution of debris disks. The Astrophysical Journal 768(1), 25 (2013)
10. Hartigan, J.A., Wong, M.A.: Algorithm as 136: A k-means clustering algorithm. Series C (Applied Statistics) 28(1), 100–108 (1979)
11. Hu, W., Singh, R.R.P., Scalettar, R.T.: Discovering phases, phase transitions, and crossovers through unsupervised machine learning: A critical examination. Physical Review E 95(6) (jun 2017)
12. Kassambara, A.: Practical guide to cluster analysis in R: Unsupervised machine learning, vol. 1. STHDA (2017)
13. MacQueen, J., et al.: Some methods for classification and analysis of multivariate observations. In: Proceedings of the fifth Berkeley symposium on mathematical statistics and probability. vol. 1, pp. 281–297. Oakland, CA, USA (1967)
14. Millán, E.N., Ruestes, C.A., Wolovick, N., Bringa, E.M.: Boosting materials science simulations by high performance computing. In: ENIEF 2017. AMCA (Nov 2017)
15. Murtagh, F., Legendre, P.: Wards hierarchical agglomerative clustering method: which algorithms implement wards criterion?. Journal of classification 31(3), 274–295 (oct 2014)
16. Musiolik, G., de Beule, C., Wurm, G.: Analog experiments on tensile strength of dusty and cometary matter. Icarus 296, 110–116 (nov 2017)
17. Ntampaka, M., Trac, H., Sutherland, D.J., Battaglia, N., Póczos, B., Schneider, J.: A machine learning approach for dynamical mass measurements of galaxy clusters. The Astrophysical Journal 803(2), 50 (2015)
18. Ringl, C., Bringa, E.M., Bertoldi, D.S., Urbassek, H.M.: Collisions of porous clusters: A granular-mechanics study of compaction AND fragmentation. The Astrophysical Journal 752(2), 151 (Jun 2012), http://dx.doi.org/10.1088/0004-637X/752/2/151
19. Ringl, C., Urbassek, H.M.: A lammps implementation of granular mechanics: Inclusion of adhesive and microscopic friction forces. Computer Physics Communications 183(4), 986–992 (Apr 2012), http://dx.doi.org/10.1016/j.cpc.2012.01.004
20. Rives, A., Goyal, S., Meier, J., Guo, D., Ott, M., Zitnick, C.L., Ma, J., Fergus, R.: Biological structure and function emerge from scaling unsupervised learning to 250 million protein sequences (apr 2019)
21. Stukowski, A.: Visualization and analysis of atomistic simulation data with OVITO–the open visualization tool. Modelling and Simulation in Materials Science and Engineering 18(1), 015012 (dec 2009)