

A New Clustering Boids Algorithm for Data Mining

Marcio Frayze David and Leandro Nunes de Castro

Mackenzie University
Rua da Consolação, 930 - São Paulo - SP - Brazil
mfdavid@gmail.com, lnunes@mackenzie.br

Abstract. This paper presents a multi-agent flocking approach for data clustering. The new algorithm, called cBoids, is proposed based on the classic Boids model with a few changes on the Boids behavior. In this new algorithm, each Boid represents an object from the database and the original three rules from the Boids model were changed so that the values in the database have influence on their behavior and two other rules were added. The new rules are responsible for the creation and destruction of centroids, which represent the formed clusters. In the preliminary experiments described here the algorithm was successfully tested on the Ruspini, Animals and Iris datasets.

Keywords: Data clustering, natural computing, Boids.

1 Introduction

This paper introduces a new clustering algorithm, named cBoids (Clustering Boids), which was designed based on the pioneer Boids flocking model by Reynolds. The cBoids algorithm is able to group a set of data in an autonomous and dynamic way using a variation of the three behavioral rules from the original Boids model with the addition of other two probabilistic rules. This new algorithm was preliminary tested on the Ruspini, Animals and Iris datasets, showing very promising results.

The paper is organized as follows. First, the paper introduces the motivation that inspired the creation of this new clustering algorithm, its main objectives and reviews the related works. In Section 2 it is presented a brief introduction to data clustering. In Sections 3 and 4 a brief introduction to the Boids model and the proposed cBoids algorithm, respectively, are presented. Preliminary experimental results with the cBoids algorithm and comparisons with the k-means algorithm are described in Section 5. The paper is concluded in Section 6.

1.1 Motivation and Main Objectives

Bio-inspired algorithms have been used for various purposes for a long time. Artificial neural networks, generic algorithms, artificial immune systems and the

Boids model are just some examples of bio-inspired systems that can be applied to several engineering problems, for instance, data grouping (clustering), optimization and autonomous navigation [1]-[4]. These algorithms are known for their good results in solving complex problems. Therefore, new proposals for bio-inspired algorithms continue to emerge and the algorithms already proposed continue to be improved. This paper aims at proposing another bio-inspired algorithm for clustering data.

During the initial phase of this project some theories for the creation of mechanisms to perform data clustering based on the behavior of birds were studied and tested. After some testing and tweaking, the Boids model was selected to be used as a basis for the creation of this new algorithm. Although the Boids model has been initially created to perform computer animation, it proved to be robust enough so that, with adequate modifications, it could be used for data clustering. This paper will propose and detail the operation of the cBoids algorithm and its implementation in the Java language (as an Applet application) and presents the results achieved so far.

1.2 Related Works

This section briefly reviews the use of the Boids algorithm for data clustering.

G. Folino and G. Spezzano [5] created an algorithm, called SPARROW (SPAtial ClusteRing AlgoRithm thrOugh SWarm Intelligence), as a multi-agent system that uses some modified Boids' rules so as to transform the Boids into hunters foraging for clusters in spatial data. Each agent searches the clusters in parallel and changes their colors as signals to other flock members for the presence or lack of significant patterns in the data. The entire flock then moves towards the agents (attractors) that have discovered interesting regions in order to help them and avoid the uninteresting areas marked as obstacles.

The Multiple Species Flocking (MSF) model proposed by Cui and Potok [6] extends the Boids model to perform Data Clustering in Dynamic databases. In addition to the three basic rules from the original Boids model, the MSF model introduces a fourth rule (a feature similarity rule) that influences the Boids movements. With this new rule, the Boids try to stay closer to similar Boids and distant from dissimilar Boids. The 'attraction' and 'repulsion' intensity is proportional to the similarity or dissimilarity among Boids, respectively.

Youssef et al. proposed a new algorithm, called Particle Swarm-Based Clustering [7], in which every agent is assigned to a data object. This agent follows the original three Boids rules and move around the space. The population decides to stop moving in the solution space if the maximum change in its position is less than a pre-defined threshold. This means that all the agents are stable and almost stop moving. The fitness value is measured every complete cycle to decide if the population made a good solution or a new population should be generated to get better results.

2 Data Clustering

Data Clustering aims at separating or categorizing data into different classes, what is achieved usually using some mechanism to calculate the similarity (or dissimilarity) degree among objects [8]. Thus, after the execution of a data clustering algorithm, it is expected that the objects from the resulting groups have high similarity among one another and low similarity with objects from other groups. The clustering techniques are used to discover natural groups in the data set and identify abstract structures that may reside in these groups [9]. Data clustering is a useful process to extract meaning from sets of unlabeled data or to perform data exploration for pattern recognition [8]. It is, therefore, a task that aims at separating heterogeneous datasets into homogeneous subgroups or segments, where the objects are grouped according to some similarity among them [10]. The data clustering goal is to group objects that are similar to one another and separate those that are not. Unlike the classification task, the set of labels are not known in advance [11].

3 The Boids Model

Boids is an artificial life model proposed in 1987 by Craig W. Reynolds and aims to model, quite realistically, the coordinated movements of flocks of animals such as birds and fish. The main purpose of this model is to assist the production of computer-based animations [12]-[14].

Reynolds was fascinated with the way flocks of animals are formed and move through the environment in a way that seems random, but at the same time synchronized. The route taken by birds, for example, is quite complex. Writing a script to create a computer animation of flocks containing several birds and their paths in the traditional way is a labor-intensive task and hardly produces a realistic result. It is necessary to specify every detail of the path that each individual will follow to avoid the collision and provide realism. Also, if a change in the route traveled by the flock is required, all work must be redone. Reynolds decided to seek a new way to solve this problem and created a model which is based on three simple behavioral rules for each of the “virtual agents” (Boids) describing how one should behave according to their condition site (position and velocity of other agents around). The three rules that characterize this model are:

- *Separation*: avoid collision with other Boids;
- *Alignment*: align the angle with the neighbors;
- *Cohesion*: move toward the average position of the neighbors.

When a Boid encounters another Boid within its vision field, the separation rule causes it to move to the opposite direction, thus avoiding the collision. To accomplish this, the average angle value of the Boid’s neighbors is calculated and then the Boid’s angle is altered to the opposite direction. The strength of separation is inversely proportional to the distance between the Boids; that is, the closer the individuals, the stronger the separation among them.

When one Boid sees another Boid within its vision field, the alignment rule seeks to change the angle and speed values so that they tend to form groups flying in approximately the same direction and speed.

The cohesion rule brings the Boids closer. This rule makes each Boid move toward the average position of the Boids in the neighborhood, thereby making them even closer to one other.

As a result, to create an animation of birds flying around using the Boids algorithm, it is just necessary to provide the starting coordinates for the Boids and make each of them follow the three behavioral rules. An emergent coordinated behavior will emerge, avoiding collisions and promoting the formation of groups.

Other rules may be added to achieve specific goals. For instance, one can add rules to make the Boids move to a certain region of the environment, avoid collision with static objects, and so on. It is difficult to foresee the exact path to be taken by each Boid, but by complying with the rules, they will eventually reach the goal. These emergent, self-organized behaviors generate realistic group behavior. Thus, a few simple rules, which only account for local conditions, can, after several iterations, achieve complex tasks on the global stage. Solitary and smaller groups of Boids tend to group together forming larger groups and, in the presence of obstacles, larger groups may temporarily divide into smaller groups [13].

4 The cBoids Algorithm

In the cBoids algorithm, the Boids are arranged in a 2D toroidal environment of limited size. Each object in the database to be analyzed is represented by a Boid. If there are n objects to be grouped, then n Boids are initially created to represent the dataset. Each Boid can be seen as a projection of a multidimensional object from the database. This is because the Boids move about the 2D environment, whilst the objects may be in a space of much larger dimension. Thus, each object from the database is indexed by a Boid, in a process akin to the projection of the objects into the 2D Boids' space.

The affinity between two Boids indicates quantitatively the similarity between the corresponding objects: the more similar the objects, the greater the affinity between the corresponding Boids. The cBoids algorithm uses the affinities of Boids as one of the main parameters for the task of grouping data. It is therefore important that the implementation of the algorithm is able to effectively compare the attributes of two Boids (similarity between two objects from the database). The calculation of the affinity A_{ij} between Boids \mathbf{x}_i and \mathbf{y}_j in the implementation proposed in this paper is equal to the inverse Euclidean distance between the objects from the database that these Boids represent (normalized between 0 and 1):

$$A_{ij} = \left(\sqrt{\sum_{k=1}^L (x_{ik} - y_{jk})^2} \right)^{-1}$$

The cBoids algorithm uses a variation of the three behavioral rules proposed in the original Boids model and also adds two new behavioral rules: the *merging* and the *splitting* of centroids, as detailed in the following. These are important to allow for vector quantization and also the automatic determination of the number of clusters n in the data set.

The exploration concepts of cBoids, borrowed from the original Boids model, have an important role. The cooperative and distributed behavior of these agents with the help of probabilistic functions and a stopping criterion, allowed the creation of an algorithm which is able to explore the solution space effectively and automatically define the number of clusters.

4.1 Centroids' Rules

Each Boid flying in the environment is treated as a centroid or prototype for a cluster, and each centroid can group many objects (other Boids). For instance, in a database with 200 objects, the algorithm will initially have 200 Boids, and each one is seen as one centroid of an existing cluster (each cluster containing a single object). A probabilistic rule is then used to merge centroids, thereby creating clusters of Boids (objects). Another probabilistic rule is responsible for creating new centroids, splitting the groups previously merged and forming two distinct groups (one containing $ng-1$ objects and the other containing 1 object, where ng is the number of objects that had previously been grouped).

The Merging Centroids rule goal is to merge two Boids (objects) into one centroid. When the algorithm starts, each group has only a single object in the database. Whenever two Boids are within the vision area of the other, there is a probability Pm_{xy} for the two groups x and y to merge into a single group. This probability is proportional to the affinity between the two Boids (which represent two centroids):

$$Pm_{xy} = \frac{1}{\sqrt{(x_1 - y_1)^2 + (x_2 - y_2)^2 + \dots + (x_n - y_n)^2}} = \frac{1}{\sqrt{\sum_{i=1}^n (x_i - y_i)^2}}$$

The two Boids merge and one disappears from the environment, whilst the other continues to fly around normally. Along with the Boid that keeps on flying there will be a reference to that Boid that was merged and stopped flying, and all the objects that it may have grouped previously.

The Merging Centroids rule is probabilistic and sometimes can merge two objects that should not be part of the same group. To minimize this problem, a new rule that creates new centroids was proposed. When two or more objects are grouped, there is a

probability that one of them leaves the group and returns to fly around in the environment, looking for a new group. The probability of an individual leave its cluster is proportional to the difference between its affinity with its current group and its greater affinity with the other groups. Initially, the current affinity with its group ($cgaf$) is calculated, which is the affinity between the Boid and the centroid of the group it belongs to. Then, it is calculated the affinity of this Boid with the centers of all remaining groups; the highest one is named $ggaf$ (greatest affinity group). The probability Pl of a Boid leaving the group it belongs to is directly proportional to the difference $ggaf - cgaf$:

$$cgaf = \sqrt{\sum_{i=1}^n (\mathbf{b}_i - \mathbf{c}_i)^2} \quad ggaf = \sqrt{\sum_{i=1}^n (\mathbf{b}_i - \mathbf{g}_i)^2} \quad Pl = ggaf - cgaf \quad ,$$

where \mathbf{b} is the object that the Boid represents, \mathbf{c} is the Centroid (average values) of the current group and \mathbf{g} is the Centroid of the group with greatest affinity with \mathbf{b} .

If the value of an object changes, this rule will reflect this change and may exclude this object from its present group. Therefore, this rule may play an important role when applied to dynamic datasets.

4.2 The New Separation, Alignment and Cohesion Rules

Under the cBoids algorithm, the strength of the separation between two Boids is inversely proportional to the affinity between the Boids: the lower the affinity, the greater the separation strength between them. The degree of alignment varies with the affinity between the Boids: the greater the affinity between the Boids, the stronger the alignment between them. The cohesion rule of the cBoids algorithm also works similarly to the original model rule, but the strength of cohesion varies with the Boids' affinity: the greater the affinity, the stronger the cohesion, and vice-versa.

4.5 Stopping Criterion

The algorithm stops running when a maximal number of cycles passes without changes in the groups formed. If c cycles pass with no change in the groups; that is, no new centroid is proposed or no two centroids merge to form a single one, then the algorithm assumes the system has stabilized and stops running. The number of cycles is important: if too little, the algorithm may stop before finding the best solution, and if too large, the algorithm may merge groups that should be separated resulting in a solution with less groups than it should. Tests with the Ruspini, Animals and Iris datasets indicate that 1000 iterations without changes in the groups is an adequate limit in most situations.

4.6 Overview of the cBoids Algorithm

1. Start the system with n Boids, where n is the number of objects in the database. Each Boid is initially a centroid Boid. After some cycles a centroid Boid will be the one that represents more than a single object, what will occur as a consequence of the merging rule.
2. For each centroid Boid, do:
 - a. Calculate the separation angle
 - b. Calculate the average angle of neighbours
 - c. Calculate the cohesion angle
 - d. Calculate the average affinity with neighbours
 - e. Update the Boid's angle considering the average affinity with its neighbours
 - f. For each Boid previously grouped do:
 - i. Calculate cgaf (affinity with the current group)
 - ii. Calculate ggaf (greatest affinity with a group)
 - iii. Calculate the ungrouping probability
 - iv. If the rule decides to ungroup, do:
 1. Remove the Boid group
 2. Return the Boid to the environment in a random 2D position, velocity and angle
3. For each of the other centroids, do:
 - a. Calculate the probability of merging, proportional to the affinity between the centroids
 - b. If the rule decides to merge the groups, do:
 - i. Merge groups (copy all elements from one to the other)
 - ii. Remove one of the centroids from the grid
 - c. If stopping criterion is met
 - i. Stop the algorithm

5 Experiments and Results

The cBoids algorithm was implemented so that it could be tested in practice. The implementation was built on the Java 1.6 language and initial tests were conducted using the Ruspini, Animals and the Iris datasets. For each dataset the algorithm was executed 10 times and the results to be shown are average over this number of experiments. The same datasets were tested with the k-means algorithm and a brief comparison is presented for each dataset.

5.1 Animals Database

The Animals dataset contains 16 objects with 13 binary attributes each, and has been used to test some self-organizing networks for clustering [15].

Initially, the cBoids algorithm generates 16 groups (16 centroids), where each group contains a single object.

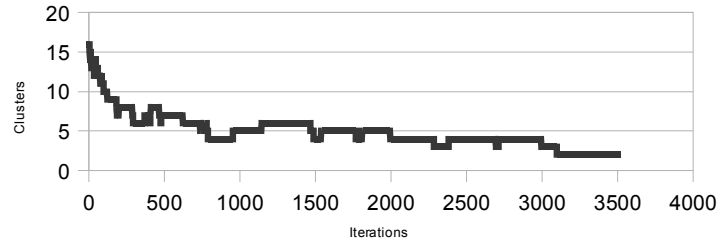


Fig. 1. Number of clusters over time for the Animals database.

Figure 1 shows the number of clusters over time in a typical execution of cBoids for the Animals database. Initially there is a rapid reduction in the number of clusters, followed by periods of bursts and reductions in the number of centroids, eventually reaching a more stable dynamics converging to the proposed solution. The most usual solution proposed by the algorithm for this database was:

- Cluster 1: dog, cow, fox, wolf, zebra, lion, horse, tiger and cat.
- Cluster 2: duck, chicken, eagle, dove, owl, hawk and goose.

By examining the results provided by the algorithm it can be observed that the mammals and the birds were separated.

During the tests, some executions resulted in less stable clusters. Sometimes the algorithm proposed three groups and sometimes it proposed two groups, but mixed one or two Boids (objects) in the groups. The algorithm was executed 10 times and, considering the above result as the correct clustering, the average accuracy was 94.38%. In the worst case the accuracy was 87.50% and in the best case it was 100%.

The resulting clustering of the k-means algorithm with the same dataset and the parameter k (number of clusters) set to two was the same as the one presented by cBoids.

5.2 Ruspini Database

Ruspini is a database with 72 integer-valued objects that has been broadly used to assess the performance of novel clustering algorithms [16].

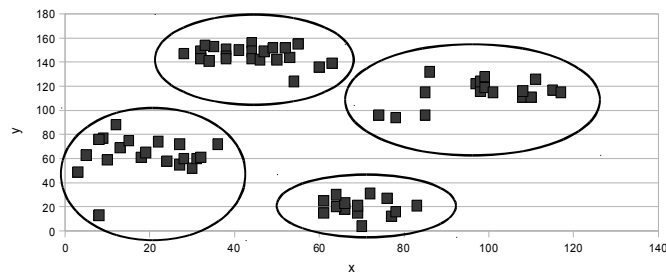


Fig. 2. The Ruspini database objects and clusters.

During the execution of the tests under the Ruspini dataset, one of the tests resulted in only three clusters as a solution, but the usual result was four clusters with perfect clustering as presented under Figure 2. The algorithm was executed 10 times and the average accuracy was 96.70%. In the worse case, the accuracy was 77.80% and at the best cases the accuracy was 100%.

Running the k-means algorithm for this dataset and setting the k parameter to four resulted, in most cases, the same results proposed by the cBoids algorithm.

5.3 Iris Database

In order to assess the performance of the algorithm on a real-world dataset, tests were performed using the Iris dataset of Fisher (1936). This dataset is composed of 150 objects distributed in three different classes of plants, being one of them linearly separable from the others and the remaining two classes non-linearly separable. After running cBoids and k-means for this dataset it was possible to observe that none of these algorithms was capable of separating the non-linearly separable classes. Furthermore, cBoids merged Boids until two remained in the population, suggesting that the dataset contains only two groups of data. Further experiments and modifications with this dataset should be performed in order to assess if cBoids can be successfully applied to datasets in which the number of classes is greater than the number of natural clusters available, which is the case with the Iris dataset.

6 Conclusions and Future Investigations

The cBoids algorithm proposed in this paper demonstrated to be capable of solving simple clustering problems. For the two preliminary experiments performed it was able to correctly group two sets of synthetic data with high accuracy most of the time. The tests showed that cBoids and k-means results are very similar, but cBoids has two advantages over k-means: it does not require the specification of the number of clusters, which is defined automatically, and it is also able, in principle, to handle dynamic datasets. It is also important to note that cBoids is efficient for finding natural clusters, that is, clusters where there is always an intra-cluster distance sufficiently greater than the inter-cluster distance [17].

There is still a lot of work to be done yet. First, the algorithm has to be tested on larger and real-world datasets, such as bioinformatics and text data. Further, the algorithm has to be compared with other proposals from the literature, including the method proposed by Cui and collaborators [6]. Last, but not least, the dynamic nature of the algorithm suggests that it can be applied to cluster time-varying datasets; and this will also be investigated.

7 Acknowledgements

The authors thank CAPES, CNPq, Fapesp and Mackpesquisa for the financial support.

References

1. de Castro, L. N., Von Zuben, F. J.: An Evolutionary Immune Network for Data Clustering. In: Proceedings of 6th Brazilian Symposium on Neural Networks, pp. 84--89. IEEE Computer Society, Brazil (2000).
3. Handl, J., Meyer, B.: Ant-based and swarm-based clustering. *Swarm Intelligence* 1(1), pp. 95--113. (2007)
4. Mondada, F., Guignard, A., Bonani, M., Bär, D., Lauria, M., Floreano, D.: SWARM-BOT: From Concept to Implementation. Autonomous Systems Lab, Swiss Federal Institute of Technology in Lausanne (EPFL) (2003).
5. Folino, G., Spezzano, G.: An Adaptive Flocking Algorithm for Spatial Clustering. *Lecture Notes in Computer Science*, 2439, pp. 924--933 (2002).
6. Cui, X., Potok, T. E.: A Distributed Agent Implementation of Multiple Species Flocking Model for Document Partitioning Clustering. *Lecture Notes in Artificial Intelligence*, LNAI 4149, pp. 124--137 (2006).
7. Youssef, S. M., Rizk, M., El-Sherif, M.: Dynamically Adaptive Data Clustering Using Intelligent Swarm-like Agents. *International Journal of Mathematics and Computers in Simulation*, 1(2), pp. 108-118 (2007).
8. de Sá, M.: *Pattern Recognition: Concepts, Methods and Applications*. Springer, first edition (2001).
9. Kogan, J.: *Introduction to Clustering Large and High-Dimensional Data*. Cambridge University Press, University of Maryland, Baltimore (2007).
10. Jain, A. K., Murty, M. N., Flynn, P. J.: Data Clustering: A Review. *ACM Computing Surveys*, Vol. 31, No. 3 (1999).
11. Amigó, E., Gonzalo, J., Artiles, J., Verdejo, F.: A comparison of Extrinsic Clustering Evaluation Metrics based on Formal Constraints Technical Report. Departamento de Lenguajes y Sistemas Informaticos UNED, Madrid, Spain (2008).
12. Reynolds, C. W.: Flocks, herds and schools: A distributed behavioral model. *ACM SIGGRAPH Computer Graphics*, v.21 n.4, p.25-34 (1987).
13. de Castro, L. N.: *Fundamentals of Natural Computing: Basic Concepts, Algorithms, and Applications*. Chapman & Hall/CRC (2006).
14. Reynolds, C. W.: *Big Fast Crowds on PS3*. Sony Computer Entertainment, US R&D (2006).
15. Haikyn, S.: *Neural Networks: A Comprehensive Foundation*, Prentice- Hall, Old Tappan, N. J. (1999).
16. Ruspini, H. R.: Numerical Methods for Fuzzy Clustering, *Information Sciences*, 2, pp. 139-350 (1970).
17. Fisher, R. (1936), "The Use of Multiple Measurements in Taxonomic Problems", *Ann. of Eugenics*, 7, pp. 179-188.