Facing Big Data by an Agent-Based Multimodal Evolutionary Approach to Classification

Mauro Giampieri, Luca Baldini, Enrico De Santis, Antonello Rizzi
Department of Information Engineering, Electronics and Telecommunications
University of Rome "La Sapienza"
Via Eudossiana 18, 00184 Rome, Italy
{mauro.giampieri, luca.baldini, enrico.desantis, antonello.rizzi}@uniroma1.it

***Abstract*—Multi-agent systems recently gained a lot of attention for solving machine learning and data mining problems. Furthermore, their peculiar divide-and-conquer approach is appealing when large datasets have to be analyzed. In this paper, we propose a multi-agent classification system able to tackle large datasets where each agent independently explores a random small portion of the overall dataset, searching for meaningful clusters in proper subspaces where they are well-formed (i.e., compact and populated). This search is orchestrated by means of a genetic algorithm able to act in a multi-modal fashion, since meaningful clusters might lie in different subspaces. Furthermore, since agents operate independently one another, their execution is parallelized across different computational units. Tests show that the proposed algorithm, E-ABC^2, is able to deal with large datasets, returning satisfactory results in terms of scalability and performances, especially when compared with our previous baseline versions.**

I. Introduction

In the Big Data era, designing powerful data mining and machine learning algorithms able to extract useful information from huge amount of data is of paramount importance. Furthermore, such big datasets are often noisy and/or with plenty of redundant information, making the data mining task and the interpretability of the learning algorithm output rather difficult.

In a general sense, data mining algorithms can be roughly divided in *supervised* and *unsupervised*: in the former case, the dataset is labelled (i.e., patterns are mapped with ground-truth class labels, in case of classification problems) and the learning algorithm exploits input-output pairs to track the decision boundary that separates different classes. In the latter case, patterns are unlabelled and regularities have to be discovered by taking into account mutual dissimilarities between patterns.

Recently, multi-agent systems [1] emerged as powerful strategies to solve data mining problems. Broadly speaking, multi-agent systems are composed by atomic units (agents) that somehow cooperate in order to reach a common goal, where the agent's definition is vague and algorithm-dependent. For example, in [2] a multi-agent approach has been used for local graph clustering. In [3] each agent runs a different clustering algorithm in order to return the best one for the data set at hand. In [4] agents negotiate one another rather than being governed by a master/wrapper process (e.g. evolutive algorithm). In [5] each agent consists in a set of data points and agents link to each other, thus leading to clustering. In [6] a genetic algorithm has been used where the agents' genetic code is connection-based: each agent is a clustering result whose genetic code builds a subgraph and, finally, such subgraphs can be interpreted as clusters. In [7] the multi-agent approach collapses into two agents: a first agent runs a cascade of principal component analysis, self organizing maps and k-means in order to cluster data and a second agent validate such results. Finally, in [8] a multi-agent algorithm has been proposed in which agents perform a Markovian random walk on a weighted graph representation of the input data set. Each agent builds its own graph connection matrix amongst data points, weighting the edges according to the selected distance measure parameters, and performs a random walk on such graph in order to discover clusters. This algorithm has been employed in [9] to identify frequent behaviours of mobile network subscribers, starting from a set of call data records.

In this work, we consider Evolutive Agent Based Clustering Classifier (E-ABC^2), an agent-based algorithm developed both for solving clustering [10], [11] and classification [12] problems. In E-ABC^2 each agent runs a very simple clustering procedure on a small sub-sample of the whole dataset. A genetic algorithm [13] orchestrates the evolution of such agents in order to return a set of well-formed clusters, thus discovering possible regularities in the data set at hand. Many clustering algorithms deal with a global metric. This means that the dissimilarity measure between patterns weights, in a suitable fashion, each feature leading to a global feature selection procedure. In the proposed approach weights are valid locally. The property of "locality" is ensured by a weighted dissimilarity measure alongside the not straightforward clustering procedure adopted and, practically, it is valid in the region around the cluster representatives. In other words, the evolutionary clustering process selects, through the agents, the subspace where these clusters are well-formed.

Three main concepts are well addressed with E-ABC^2: i) the possibility of conceiving an evolutionary learning algorithm capable of dealing with local metrics, instead of global metrics, hence learning which features are important in characterizing input space regions, possibly belonging to different classes for the classification problem at hand [14]; ii) the use of the multi-agent paradigm that allows to face the pattern recognition problem by relying on possibly distributed and scalable architectures [15], [16], suitable in Big Data contexts,

978-1-7281-6926-2/20/$31.00 ©2020 IEEE

by solving a complex problem by dividing it in small (simple) sub-problems [17]; iii) the adoption of a clustering algorithm, as the main core of agent's job, that allows building a gray-box model useful for further data discovery analysis [18].
The contribution of this paper is two-fold:

1) we propose a multimodal variant of E-ABC^2: in fact, in previous works [10]–[12], a standard genetic algorithm took care of orchestrating the swarm of agents. However, this can limit the exploration capabilities, especially as the local metric learning task is concerned;
2) since each agent independently operates on a specific dataset shard randomly drawn from the dataset, their jobs can be embarrassingly parallelized.

The remainder of the paper is structured as follows: in Section II we introduce the E-ABC^2 basic algorithm, along with the two aforementioned variants in Section II-D and II-E. Section III shows the experimental results, whereas Section IV concludes the paper.

II. E-ABC^2

E-ABC^2 is a classification algorithm which takes advantage of the approach proposed for E-ABC algorithm and uses it to extract meaningful clusters from the data set, which will later be used for building the classification model. Hereinafter, let us consider the overall dataset $\mathcal{S}$ to be split into three non-overlapping training, validation and test set ($\mathcal{S}_{\text{tr}}$, $\mathcal{S}_{\text{vl}}$ and $\mathcal{S}_{\text{ts}}$).

A. E-ABC^2 Agent Behaviour

The key role of E-ABC^2 agents is to construct well-formed clusters, starting from a subset of patterns $\mathcal{R}$ randomly sampled from training set, used to build the decision regions of the classification model $\mathcal{M}$. The underlying clustering algorithm, individually performed by each agents, is based on the well-known Basic Sequential Algorithmic Scheme (BSAS). This method scans sequentially the input data and assigns a pattern to an existing cluster if the pattern-to-representative distance is below a given threshold θ. If the pattern cannot be assigned, it will serve as a centroid for a new cluster. However, especially for low θ values, BSAS is likely to return a huge number of small clusters: to limit this issue, a maximum number of allowed clusters Q can be defined by the end-user and new clusters can be spawned provided that the number of already-available clusters is below Q. A major drawback of BSAS is its sensitivity to pattern order and outliers due to the sequential approach followed by the procedure. In order to mitigate this issue, in E-ABC^2, a Reinforcement Learning-based BSAS (RL-BSAS), has been considered [19]. In this variant, an energy value S is assigned to the each cluster. When a cluster C receives a new pattern, its energy is increased by a value $\alpha \in [0,1]$, whereas the energies of all other clusters are diminished by a value $\beta \in [0,1]$. As consequence, relevant clusters will survive, giving high level of energies, whereas badly-formed clusters will eventually vanish when their energies approach to zero. Each E-ABC^2 agent relies on a set of parameters:

- a binary mask $\mathbf{w} \in [0,1]^n$ is in charge of selecting only relevant features, defining the specific (local) metric used by the agent in searching for well-formed clusters;
- the maximum radius θ for the BSAS procedure;
- the ratio parameter $r = \alpha/\beta \in (0,1)$ involved in the RL-BSAS.

Each agent exploits the aforementioned parameters in order to cluster the input shard $\mathcal{R}$. Specifically, it first retains features from patterns in $\mathcal{R}$ belonging to 1's in $\mathbf{w}$, then runs the RL-BSAS parameter with threshold θ, with penalty and reward factors $\alpha = r \cdot \beta$. For the sake of simplicity, α has been set to 1, in such a way that β can easily be evaluated as $\beta = \alpha/r$. Hence, the basic agent task is to return a partition $\mathcal{P} = \{C_1, \ldots, C_{|\mathcal{P}|}\}$. Before returning $\mathcal{P}$ the agent merges overlapped clusters which may occur due to the RL-BSAS architecture. The single agent behaviour is summarized in Algorithm 1.

Algorithm 1 E-ABC^2 Agent Behaviour

$\mathcal{P}$	-	Partition of clusters $\{C_1, \ldots, C_{\lvert\mathcal{P}\rvert}\}$
θ_k	-	radius of k^{th} cluster in $\mathcal{P}$
$d(\cdot;\cdot)$	-	suitable distance between entities
$\mathcal{R}$	-	agent data shard

Input: $\mathcal{R}$
Output: $\mathcal{P}$
procedure AGENTEXECUTION
  $\mathcal{P} = \text{RL-BSAS}(\mathcal{R})$
  repeat
    for each pair C_i, C_j in $\mathcal{P}$ **do**
      Evaluate $D = d(C_i, C_j)$
      if $D < \theta_i$ && $D < \theta_j$ **then**
        Merge C_i with C_j
  until a merge occurs

B. Evolutive Orchestration and Model Synthesis

As stated in Section I, a genetic algorithm is in charge of orchestrating the swarm of agents and of returning a suitable model. In this evolutive scenario, each agent acts as an individual for the evolving population. As for Section II-A, the agent behaviour is determined by several parameters driving the subspace in which clusters shall be found ($\mathbf{w}$) and the RL-BSAS per se (θ, r). These parameters compose the genetic code, that reads as

$$\begin{bmatrix} \theta & \mathbf{w} & r \end{bmatrix} \tag{1}$$

Each agent exploits the parameters written in its genetic code in order to return a partition $\mathcal{P}$ of the random data shard $\mathcal{R}$ as described in Section II-A. As $\mathcal{P} = \{C_1, \ldots, C_{|\mathcal{P}|}\}$ is returned, each cluster C in the partition is evaluated thanks to a quality index $f_{\text{cc}}(C)$ that considers both its compactness $f_{\text{co}}(C)$ and cardinality $f_{\text{ca}}(C)$, i.e.

$$f_{\text{cc}}(C) = \lambda \cdot f_{\text{co}}(C) + (1-\lambda) \cdot f_{\text{ca}}(C) \tag{2}$$

where, in turn

$$f_{\text{co}}(C) = \frac{\left(1 - |C|^{-1} \sum_{x \in C} d(x, \mu)\right) - co_{\min}}{co_{\max} - co_{\min}} \quad (3)$$

$$f_{\text{ca}}(C) = \frac{|C| - ca_{\min}}{ca_{\max} - ca_{\min}} \quad (4)$$

where $co_{\min}$, $co_{\max}$, $ca_{\min}$ and $ca_{\max}$ are the minimum and maximum compactness and cardinality values observed so far during the evolution, μ denotes the centroid of the cluster and $\lambda \in [0, 1]$ is a user-defined trade-off parameter. In order to retain only good quality clusters, for each agent, only the best cluster $C^\star$ is retained, namely the cluster in $\mathcal{P}$ that maximizes Eq. (2). Since we are dealing with a supervised problem, each pattern is mapped with a ground-truth class label, hence each cluster can also be assigned to one of the problem-related classes by considering the most frequent label amongst the patterns in the cluster itself. Another quality index can be defined by considering a supervised approach: let $\mathcal{S}_{\text{vl}}$ be a validation set, then it is possible to define the accuracy of cluster C, say $Acc(C)$ as the ratio between the validation data sharing the same label as C that fall within C and the number of all validation set patterns falling in C. The latter is only evaluated on $C^\star$ and if $Acc(C^\star)$ is greater than a user-defined threshold $Acc_{\min}$ and it is greater than an adaptive parameter Acc_{gl}, then $C^\star$ is added to the model $\mathcal{M}$, incrementally built generation-by-generation. $\mathcal{M}$ is the core of the classification model: after all agents have offloaded their good clusters in $\mathcal{M}$, then it is possible to consider such clusters as building blocks for the classification model. The Acc_{gl} parameter is the global accuracy of $\mathcal{M}$ computed over the entire $\mathcal{S}_{\text{vl}}$. Each agent is finally assigned with a fitness function f defined as

$$f = Acc_{gl} \cdot Acc(C^\star) + (1 - Acc_{gl}) \cdot f_{\text{cc}}(C^\star) \quad (5)$$

The rationale behind this formulation can be summarized as follows: at early generations we do not expect to find good clusters in $\mathcal{M}$, hence Acc_{gl} will be rather low and the fitness takes mainly into account the quality index $f_{\text{cc}}(C)$. For the sake of completeness, the procedure designed for the agent's fitness evaluation just discussed, is reported in Algorithm 2. After all agents have been evaluated, standard genetic operators (elitism, crossover, selection, mutation) take care of moving from one generation to the next. The evolution goes on until either N_G generations have been completed or an early-stop criterion is triggered: the latter counts the number of consecutive generations with an Acc_{gl} increment below a given threshold. If the count overcomes a maximum value the evolutionary procedure terminates. A general picture of the whole system evolution and model building can be found in Algorithm 3.

Algorithm 2 Agents Fitness Evaluation and Model Design

$\mathcal{P}_i$ - Partition of clusters $\{C_1, \ldots, C_{|\mathcal{P}|}\}$ i^{th} agent
$\mathcal{R}_i$ - data shard for i^{th} agent
$\langle C \rangle$ - cluster compactness
$|C|$ - cluster cardinality

Input: P, $\mathcal{R}_i$, $\mathcal{S}_{\text{vl}}$
Output: Agent Fitness f, Best Cluster $C^\star$

procedure AGENTSEVALUATION
  Compute Acc_{gl} for the model $\mathcal{M}$ on $\mathcal{S}_{\text{vl}}$
  for each agent a_i in P **do**
    $\mathcal{P}_i$ = **agentExecution**($\mathcal{R}_i$)
    for each C in $\mathcal{P}_i$ **do**
      $f_{\text{cc}}(C) = \lambda \cdot \langle C \rangle + (1 - \lambda) \cdot |C|$
    $C^\star = \underset{C \in \mathcal{P}_i}{\text{argmax}}\, f_{cc}$
    Compute cluster accuracy $Acc(C^\star)$ on $\mathcal{S}_{\text{vl}}$
    $f = Acc_{gl} \cdot Acc(C^\star) + (1 - Acc_{gl}) \cdot f_{\text{cc}}(C^\star)$

Algorithm 3 E-ABC^2 Training Phase

N - Number of agents
P_i - population of i^{th} generation
$\mathcal{R}_j$ - data shard for j^{th} agent

Input: $\mathcal{S}_{\text{tr}}$, $\mathcal{S}_{\text{vl}}$
Output: $\mathcal{M}$

procedure TRAINAGENTS
  Set $\mathcal{M} = \emptyset$
  Initialize individuals in P_0 with random genetic code
  for $i = 0$ to N_G **do**
    Draw random shards $\{\mathcal{R}_1, \ldots, \mathcal{R}_N\}$ from $\mathcal{S}_{\text{tr}}$
    $C^\star$, f = **agentsEvaluation**($P_i, \{\mathcal{R}_1, \ldots \mathcal{R}_N\}, \mathcal{S}_{\text{vl}}$)
    Compute accuracy Acc_{gl} of model $\mathcal{M}$ on $\mathcal{S}_{\text{vl}}$
    for each agent a_j **do**
      if $Acc(C^\star) > Acc_{\min}$ && $Acc(C^\star) > Acc_{gl}$ **then**
        Add $C^\star$ to model $\mathcal{M}$
    if Stop Condition == True **then**
      break
    P_{i+1} = **Evolve**(P_i)

C. Testing Phase

At the end of the optimization procedure, the model $\mathcal{M} = \{C_1, \ldots, C_{|\mathcal{M}|}\}$ is the key player as it contains the best clusters found by the swarm during the evolution. It is worth recalling here that each cluster C is uniquely identified by the tuple $\langle \mu, \theta, \mathbf{w} \rangle$, namely its centroid, radius and subspace mask, respectively. The test set has the final goal of evaluating the performances of the final model returned by E-ABC^2. For each pattern $x \in \mathcal{S}_{\text{ts}}$:

1) the distance with respect to all clusters in $\mathcal{M}$ is evaluated. Each distance is obviously weighted according to each cluster's $\mathbf{w}$ metric parameters (subspace)
2) the resulting distances are checked against each cluster's θ: if less than (or equal to) θ, a match is scored
3) the labelling takes place according to the following rules:
   - in case of only one match, x inherits the label of its closest cluster
   - in case of multiple matches, x inherits the most frequent label across all clusters in which it felt

- in case of no matches, x is marked as *unclassified*.

D. Multimodal E-ABC^2

From Section II it is clear that each cluster is identified not only by its centroid μ and radius θ, but also by its binary mask $\mathbf{w}$ containing the parameters of the subspace in which said cluster has been found. Further, recall that the search of these parameters is driven by an evolutive metaheuristic. Standard genetic algorithms notably aim at optimizing a given fitness function in a unimodal manner, i.e., the population tends to converge to a single (sub-)optimal solution. However, since meaningful clusters might lie in different subspaces, a unimodal, plain, genetic algorithm might as well limit the exploration in this regard, returning one (or very few) subspaces. In the technical literature, these kind of problems are known as *multimodal*, as multiple optima exist [20], [21]. To this end, we removed the elite pool and introduced a *fitness sharing* based approach [22]–[25]: after the fitness of each agent has been evaluated, an additional step occurs to modify it. As if they where sharing a limited resource, close agents (in terms of search subspace) are penalized. The more agents share the same subspace, the more their fitness value is reduced. In this way, agents are encouraged to explore new subspaces that no other agents are investigating.
In particular, the considered fitness sharing tweaking on the i^{th} agent fitness function reads as [22]

$$f_i\prime = f_i / m_i \tag{6}$$

where f_i is its original fitness value and m_i is an estimate on the number of individuals sharing the same fitness as agent i. The latter formally reads as

$$m_i = \sum_{j=1}^{N} sh(d_{i,j}) \tag{7}$$

being N the population size and $d_{i,j}$ the distance between individuals i and j. Since we want to employ the fitness sharing mechanism to foster subspace exploration, $d_{i,j}$ is defined as the Jaccard dissimilarity between the $\mathbf{w}$ portions of their respective genetic codes. The sharing function $sh(\cdot)$ is defined as

$$sh(d_{i,j}) = \begin{cases} 1 - (d_{i,j}/\sigma)^{\omega} & \text{if } d_{i,j} < \sigma \\ 0 & \text{otherwise} \end{cases} \tag{8}$$

where σ is a threshold parameter and ω regulates the smoothness of the sharing function.

E. A Parallel Implementation

All of the parallelizations have been implemented by means of the OpenMP API. OpenMP provides a thread pool architecture whose size (number of available threads) can be set at the application startup. The use of a thread pool reduces the multi-threading management overhead because it avoids to create and destroy a thread each time it is used. In brief, each agent independently receives a random data set shard and outputs a set of clusters. These tasks can be easily parallelized with a naïve parfor-like loop. The master task spawns as many tasks as there are agents and forwards each one, with the agent genetic code and the data set shard, to an available thread in thread pool: threads perform in parallel agents' search for clusters. When a thread completes its task receives a new agent to perform its cluster search, until the task queue is over. The control now returns to the master process that updates the minimum and maximum bounds for compactness and cardinality ($co_{\min}$, $co_{\max}$, $ca_{\min}$ and $ca_{\max}$) and the new values are returned to the threads (agents) that, now, can evaluate their respective clusters (compactness and cardinality – see Eqs. (3)–(5)). Therefore, they are also able to elect the best cluster and evaluate its classification capabilities over the validation set. The analysis of the validation set is also included in a parfor-like loop so that the classification of each pattern in validation set is performed in parallel. However, since the validation set might be huge as well (and forwarding the entire validation set to all threads can be a serious bottleneck) it undergoes a subsampling (like the training set) and the shard is forwarded to the agent that, to this end, is now able to evaluate of its best cluster accuracy.

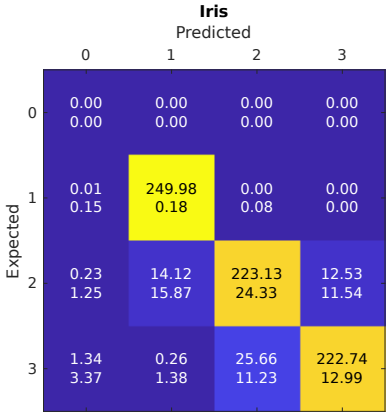
III. Experimental Results

A. Dataset Description

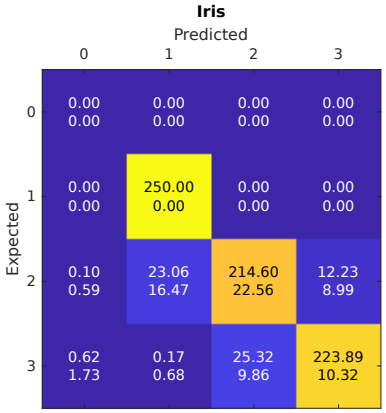
For this test we used two data sets: a modified version of well-known Iris and an additional synthetic data set conceived to stress E-ABC^2 local metric learning capability. Iris data set has been slightly manipulated to be adapted to the algorithm properties and to highlight its abilities to work on suitable random samples drawn from the dataset. The original Iris data set is composed of 3 classes, each one containing 50 patterns. We artificially increased the data set cardinality up to 3000 patterns, 1000 for each class:

1) the entire data set has preliminarily been normalized in $[0,1]$ feature by feature
2) from each original pattern p^{old}, 19 additional patterns are created. The i^{th} component of the new pattern p^{new} is computed as
$$p_i^{\text{new}} = p_i^{\text{old}} + \delta \tag{9}$$
where $\delta \in [-0.05, 0.05]$ is a displacement randomly extracted with uniform distribution
3) the obtained patterns are normalized again to ensure they lie in the unitary hypercube
4) 10 additional randomly drawn features are appended to each pattern.

The initial normalization is needed to avoid a dataset deformation introduced by patterns extraction. The role of the additional features is to increase the complexity of the feature extraction task. They are randomly drawn with a uniform distribution in $[0,1]$ so that the additional features do not hold any informative content and they just act as noisy components for the model synthesis procedure. The synthetic dataset scheme is summarized in Table I. It is composed of 8 clusters lying in different subspaces, each one assigned to one out of two classes identified by "X" and "O" symbols. All patterns lie in the unitary hyper-cube and they are uniformly

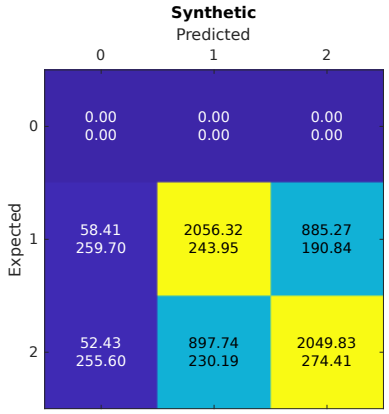


(a) Unimodal

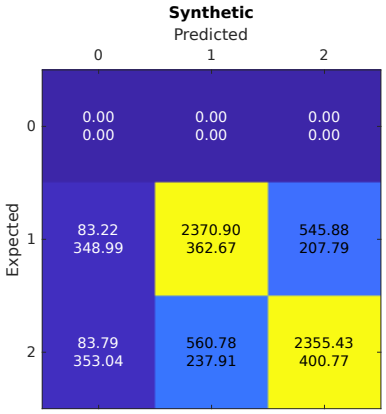


(b) Multimodal

Fig. 1. Confusion matrices for *Iris* dataset. On each box average values (top) are reported along with standard deviation (bottom).



(a) Unimodal



(b) Multimodal

Fig. 2. Confusion matrices for *Synthetic 24k* dataset. On each box average values (top) are reported along with standard deviation (bottom).

TABLE I
SCHEMATIC REPRESENTATION OF THE CLUSTERS' STRUCTURES FOR SYNTHETIC DATASET

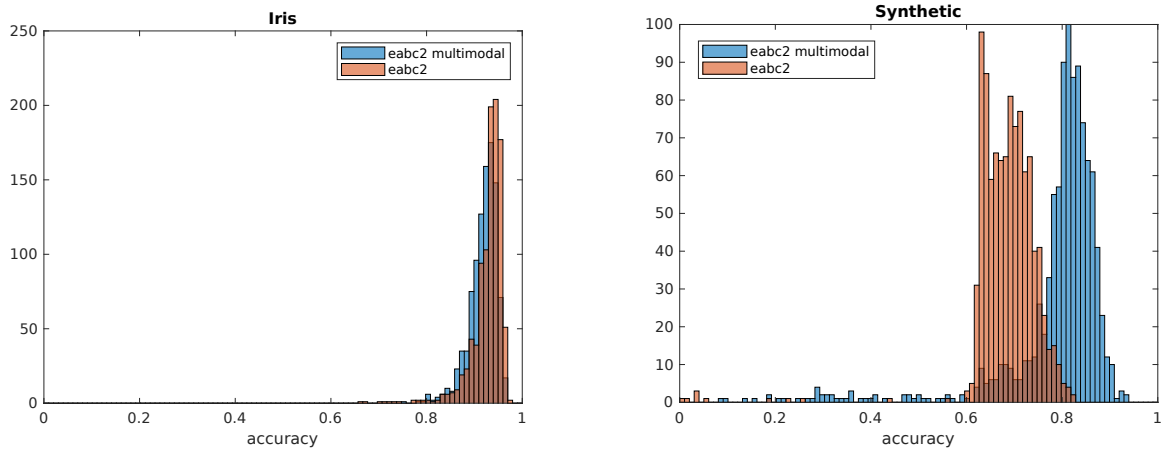
	1	2	3	4	5	6	7	8	9	10	11	12	13	14-20
C_1	X	X	X	∼	∼	∼	∼	∼	∼	∼	∼	∼	∼	∼
C_2	∼	∼	∼	X	X	∼	∼	∼	∼	∼	∼	∼	∼	∼
C_3	∼	∼	∼	∼	∼	O	O	O	∼	∼	O	O	O	∼
C_4	O	O	∼	∼	∼	∼	∼	∼	O	O	∼	∼	∼	∼
C_5	∼	∼	O	O	∼	∼	∼	∼	∼	∼	∼	∼	∼	∼
C_6	∼	∼	∼	∼	∼	X	X	X	∼	∼	∼	∼	∼	∼
C_7	∼	∼	∼	∼	O	O	O	O	O	∼	∼	∼	∼	∼
C_8	∼	∼	∼	∼	∼	∼	∼	∼	X	X	X	X	X	∼

distributed in an hyper-sphere with fixed radius. For each cluster, the corresponding hyper-sphere is defined in the feature subspace marked as "X" or "O". Features marked with ∼ are not informative for that cluster: patterns in that cluster have $[0,1]$ uniformly drawn values for that feature. Features 14-20, collapsed in a single column, are not informative for any cluster. Each cluster contains 3000 patterns. Exactly 4 clusters are assigned to "X" (in the following referred as class 1) and 4 assigned to "O" (referred as class 2), yielding perfectly balanced classes. The whole data set is composed of 24000 patterns (referred as *Synthetic 24k*).

For both data sets, *Iris* and *Synthetic 24k*, we have built in advance 100 splits in training, validation and test set by means of a stratified sampling. Recall that $\mathcal{S}_{\text{tr}}$ is used by the algorithm to look for the clusters, $\mathcal{S}_{\text{vl}}$ is needed for testing the discovered clusters and drive the overall swarm optimization and S_{ts} is used to check the performances of the final classifier when the model synthesis is over.

B. Basic and Multimodal E-ABC^2 Performance Comparison

In this subsection we are going to analyze extensive tests performed to compare the two E-ABC^2 versions: the vanilla



(a) (b)

Fig. 3. Accuracy distribution on *Iris* and *Synthetic 24k* datasets in unimodal and multimodal configuration.

(unimodal) version against the multimodal one to show how this approach affects the performances. As the multimodal implementation is concerned, we empirically found $\omega = 1$ and $\sigma = 0.4$ to be suitable values for the scaling function (Eq. (8)). Other systems parameters include $Acc_{\max} = 0.99$, $Acc_{\min} = 0.6$, $\lambda = 0.5$ and $\alpha = 1$.

Results are presented in terms of aggregated confusion matrices, averaged across 10 runs of the algorithm, where each run has been performed on 100 training-validation-test different splits. Confusion matrices also account for a dummy class '0', that corresponds to *unclassified* patterns. Fig. 1a and 2a, respectively, show the results obtained with unimodal E-ABC^2 on *Iris* and *Synthetic 24k*. Similarly, Fig. 1b and 2b show analogous results for multimodal E-ABC^2. Finally Fig. 3a and 3b, respectively, show the distribution of the overall accuracies obtained across runs of both vanilla and multimodal E-ABC^2 algorithms on both *Iris* and *Synthetic 24k*.

Fig. 3a shows there is no significant difference between basic and multimodal approaches applied to *Iris*. It is due to the internal structure of this dataset. The informative features are the same for all of the *Iris* classes. The benefit of using a multimodal search is evident in Fig. 3b. When local metric learning is actually required: the basic approach accuracy degrades if compared to multimodal. In order to better investigate this aspect, we define the stratified cluster-aware accuracy (SCAA) for the i^{th} cluster, say C_i, as

$$SCAA(C_i) = \frac{\sum_{x \in C_i} h(x)}{|C_i|} \tag{10}$$

where $h(x) = 1$ if pattern x has been correctly classified (i.e., the decision cluster model outputs the same class as C_i) and $h = 0$ otherwise. Tables II and III show the SCAA values on the *Synthetic 24k* dataset for basic and multimodal variants, respectively. Clearly, adopting the multimodal variant results in a more accurate cluster classification suggesting that, when vanilla E-ABC^2 finds a good subspace, it starts spreading and saturates the population (preventing exploration of new potentially useful subspaces), whereas the multimodal variant leads to a better subspace exploration.

TABLE II
SCAA INDICES FOR BASIC E-ABC^2 ON THE TEST SET. RESULTS ARE SHOWN SEPARATELY FOR EACH OF THE 8 CLUSTERS IN SYNTHETIC 24K.

	C_1	C_2	C_3	C_4	C_5	C_6	C_7	C_8
0	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
1	1.00	0.04	0.05	0.19	0.04	0.05	0.07	0.05
2	1.00	0.05	0.57	1.00	0.58	0.06	0.56	0.06
3	1.00	0.43	0.54	1.00	0.57	0.42	0.54	0.42
4	1.00	0.43	0.55	1.00	0.57	0.42	0.54	0.44
5	1.00	0.42	0.56	1.00	0.59	0.40	0.55	0.40
6	1.00	0.41	0.56	1.00	0.61	0.40	0.56	0.41
7	1.00	0.41	0.56	1.00	0.62	0.40	0.57	0.41
8	1.00	0.41	0.57	1.00	0.63	0.39	0.57	0.40
9	1.00	0.40	0.57	1.00	0.66	0.39	0.58	0.39
10	1.00	0.41	0.57	1.00	0.66	0.39	0.58	0.39
11	1.00	0.40	0.58	1.00	0.66	0.38	0.58	0.39
12	1.00	0.41	0.58	1.00	0.67	0.39	0.58	0.39
13	1.00	0.41	0.58	1.00	0.67	0.38	0.58	0.39

C. Multimodal E-ABC^2 Scalability

In order to address the scalability of the parallel E-ABC^2 implementation (see Section II-E) the following well-known index has been considered [26]–[29]:

Speedup: measures the ability of the parallel and distributed algorithm to reduce the running time as more workers are considered. The dataset size is kept constant and the number of workers increases from 1 to m. Hence, the speedup with m computational units reads as

$$speedup(m) = \frac{\text{running time on 1 worker}}{\text{running time on } m \text{ workers}} \tag{11}$$

TABLE III
SCAA INDICES FOR MULTIMODAL E-ABC^2 ON THE TEST SET. RESULTS ARE SHOWN SEPARATELY FOR EACH OF THE 8 CLUSTERS IN SYNTHETIC 24K.

	C_1	C_2	C_3	C_4	C_5	C_6	C_7	C_8
0	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
1	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
2	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
3	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
4	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
5	0.01	0.00	0.00	0.00	0.00	0.00	0.00	0.00
6	0.03	0.00	0.00	0.00	0.00	0.00	0.00	0.00
7	0.06	0.02	0.00	0.00	0.00	0.12	0.00	0.02
8	0.06	0.02	0.00	0.00	0.00	0.12	0.00	0.02
9	0.06	0.02	0.00	0.00	0.00	0.12	0.00	0.02
10	0.06	0.02	0.00	0.00	0.00	0.12	0.00	0.02
...								
74	0.89	0.83	1.00	0.71	0.68	0.66	0.95	1.00
75	0.89	0.86	1.00	0.71	0.68	0.67	0.95	1.00
76	0.89	0.86	1.00	0.71	0.68	0.67	0.95	1.00
77	0.89	0.86	1.00	0.71	0.68	0.67	0.95	1.00

Since in Section III-B we showed that the multimodal variant outperforms the basic (unimodal) counterpart, we concentrate the scalability results on the former only. Tests have been performed on a workstation equipped with two 6-cores Intel® Xeon® E5-2620v3 CPUs @2.40GHz, 64GB of RAM, running Linux Ubuntu 18.04. The software has been implemented in C++ using the OpenMP library for parallelization.

Figure 4 shows the speedup factor for multimodal E-ABC^2. The expected speedup behaviour is perfecly linear as, intuitively, an m-times larger computing power shall take m times less time to process a given dataset (dashed line). In order to stress the algorithm, we increasingly changed the size of the dataset shard $\mathcal{R}$: one usually tries several datasets of increasing size in order to stress how the speedup not only reacts to the number of cores, but also to different datasets [26]–[29], yet it is worth recalling that the algorithm input is indeed a set of data chunks rather than the entire dataset. To this end, we first generated a 800000-patterns synthetic dataset following the same schema as for *Synthetic 24k* and then we changed not only the number of cores (from 1 to 12, up to the number of physical cores on the available hardware), but also the dataset shard size (from 400 patterns per agent to 6400 patterns per agent). Results clearly show that as the problem gets harder (i.e., more data to be processed by each agent), then the speedup tends to the expected linear behaviour: a sign that E-ABC^2 tends to be quite robust when it comes to handling large datasets. Due to the stochastic behaviour of the overall algorithm, a meticulous care has been paid to letting (for each dataset) the algorithm to converge to the same solution (regardless of the number of cores) by a proper tweaking of the pseudo-random number generator.

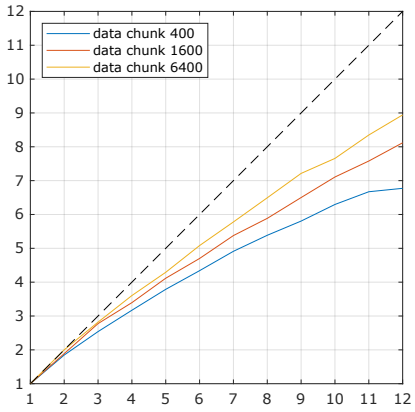


Fig. 4. Speedup performances

IV. CONCLUSION

In this paper, we proposed two improvements in E-ABC^2, a multi-agent-based classification system on the top of decision clusters. Since its original design, E-ABC^2 has been conceived to be a lightweight algorithm for finding regularities in big datasets: we started with a first version that performed unsupervised learning (i.e., clustering) and then we synthesized a classification model on the top of discovered clusters in order to let E-ABC^2 be able to solve also supervised problems. Regardless of the nature of the problem at hand, our approach is based on a swarm of multiple agents, each of which only sees a small and randomly chosen portion of the dataset and aims at finding regularities in this data shard. The swarm of agents, properly driven by an optimization metaheurisic, has the final goal of returning meaningful clusters that well characterize the decision regions of classification processes, giving the hard task of allowing different clusters to lie in different subspaces. This poses the additional challenge of jointly looking for well-formed and meaningful clusters and, at the same time, the subspace in which they lie. However, if one lets a plain genetic algorithm to drive the optimization, then it will likely converge to a single solution, as typical in elitist genetic algorithms. In order to overcome this problem, we replaced the standard genetic algorithm with a multimodal variant, able to converge simultaneously to different (sub)optimal solutions. Our tests carried out on both synthetic and synthetically augmented datasets show that the multimodal approach leads to a more precise identification of the underlying clusters. A second contribution, as instead, has a merely computational rationale: the fact that agents receive a (small) dataset portion and output a set of clusters independently one another make their evaluation particularly suited for parallelization. To this end, we adopted the OpenMP framework in order to parallelize agents' execution and the obtained results are satisfactory, yielding a speedup factor that approaches the desired linear behaviour as more and more data are analyzed. The effectiveness of the parallelization observed in our studies with demonstrative synthetic datasets, together with the natively distributed nature of the system, suggest E-

ABC^2 as a promising framework for facing Big Data, where partitioning the available data on different processing nodes is often mandatory due the large memory/space footprint. In this regard, future efforts will be devoted to extend and test the proposed system towards structured domains such as graphs [30], [31], sequences [32], audio and video data [33], text documents [34], which are common in real-world Big Data applications.

REFERENCES

[1] L. Panait and S. Luke, "Cooperative multi-agent learning: The state of the art," *Autonomous Agents and Multi-Agent Systems*, vol. 11, no. 3, pp. 387–434, Nov 2005.

[2] M. Alamgir and U. Von Luxburg, "Multi-agent random walks for local clustering on graphs," in *2010 IEEE International Conference on Data Mining*. IEEE, 2010, pp. 18–27.

[3] S. Chaimontree, K. Atkinson, and F. Coenen, "Clustering in a multi-agent data mining environment," in *Agents and Data Mining Interaction: 6th International Workshop on Agents and Data Mining Interaction, ADMI 2010, Toronto, ON, Canada, May 11, 2010, Revised Selected Papers*, L. Cao, A. L. C. Bazzan, V. Gorodetsky, P. A. Mitkas, G. Weiss, and P. S. Yu, Eds. Berlin, Heidelberg: Springer, 2010, pp. 103–114.

[4] ——, "A multi-agent based approach to clustering: Harnessing the power of agents," in *Agents and Data Mining Interaction: 7th International Workshop on Agents and Data Mining Interation, ADMI 2011, Taipei, Taiwan, May 2-6, 2011, Revised Selected Papers*, L. Cao, A. L. C. Bazzan, A. L. Symeonidis, V. I. Gorodetsky, G. Weiss, and P. S. Yu, Eds. Berlin, Heidelberg: Springer, 2012, pp. 16–29.

[5] E. Ogston, B. Overeinder, M. van Steen, and F. Brazier, "A method for decentralized clustering in large multi-agent systems," in *Proceedings of the Second International Joint Conference on Autonomous Agents and Multiagent Systems*, ser. AAMAS '03. New York, NY, USA: ACM, 2003, pp. 789–796.

[6] X. Pan and H. Chen, "Multi-agent evolutionary clustering algorithm based on manifold distance," in *2012 Eighth International Conference on Computational Intelligence and Security*, Nov 2012, pp. 123–127.

[7] J.-E. Park and K.-W. Oh, "Multi-agent systems for intelligent clustering," *World Academy of Science, Engineering and Technology, International Journal of Computer, Electrical, Automation, Control and Information Engineering*, vol. 1, no. 11, pp. 3650–3655, 2007.

[8] F. M. Bianchi, E. Maiorino, L. Livi, A. Rizzi, and A. Sadeghian, "An agent-based algorithm exploiting multiple local dissimilarities for clusters mining and knowledge discovery," *Soft Computing*, pp. 1–23, 2015.

[9] F. M. Bianchi, A. Rizzi, A. Sadeghian, and C. Moiso, "Identifying user habits through data mining on call data records," *Engineering Applications of Artificial Intelligence*, vol. 54, no. Supplement C, pp. 49 – 61, 2016.

[10] A. Martino, M. Giampieri, M. Luzi, and A. Rizzi, "Data mining by evolving agents for clusters discovery and metric learning," in *Neural Advances in Processing Nonlinear Dynamic Signals*, A. Esposito, M. Faundez-Zanuy, F. C. Morabito, and E. Pasero, Eds. Cham: Springer International Publishing, 2019, pp. 23–35.

[11] M. Giampieri and A. Rizzi, "An evolutionary agents based system for data mining and local metric learning," in *2018 IEEE International Conference on Industrial Technology (ICIT)*, 2018, pp. 1461–1466.

[12] M. Giampieri, E. De Santis, A. Rizzi, and F. M. Frattale Mascioli, "A supervised classification system based on evolutive multi-agent clustering for smart grids faults prediction," in *2018 International Joint Conference on Neural Networks (IJCNN)*, July 2018, pp. 1–8.

[13] D. E. Goldberg, *Genetic Algorithms in Search, Optimization and Machine Learning*, 1st ed. Boston, MA, USA: Addison-Wesley Longman Publishing Co., Inc., 1989.

[14] F. Wang and J. Sun, "Survey on distance metric learning and dimensionality reduction in data mining," *Data Mining and Knowledge Discovery*, vol. 29, no. 2, pp. 534–564, 2015.

[15] M. Zaharia, M. Chowdhury, M. J. Franklin, S. Shenker, and I. Stoica, "Spark: Cluster computing with working sets." *HotCloud*, vol. 10, no. 10-10, p. 95, 2010.

[16] J. Dean and S. Ghemawat, "Mapreduce: simplified data processing on large clusters," *Communications of the ACM*, vol. 51, no. 1, pp. 107–113, 2008.

[17] T. Ravindra Babu, M. Narasimha Murty, and S. V. Subrahmanya, "Big data abstraction through multiagent systems," in *Compression Schemes for Mining Large Datasets: A Machine Learning Perspective*. London: Springer, 2013, pp. 173–183.

[18] A. Di Noia, A. Martino, P. Montanari, and A. Rizzi, "Supervised machine learning techniques and genetic optimization for occupational diseases risk prediction," *Soft Computing*, 2019.

[19] A. Rizzi, G. Del Vescovo, L. Livi, and F. M. Frattale Mascioli, "A new granular computing approach for sequences representation and classification," in *The 2012 International Joint Conference on Neural Networks (IJCNN)*, 2012, pp. 1–8.

[20] K. Wong, "Evolutionary multimodal optimization: A short survey," *CoRR*, vol. abs/1508.00457, 2015.

[21] J.-P. Li, M. E. Balazs, G. T. Parks, and P. J. Clarkson, "A species conserving genetic algorithm for multimodal function optimization," *Evolutionary Computation*, vol. 10, no. 3, pp. 207–234, Sep. 2002.

[22] B. Sareni and L. Krahenbuhl, "Fitness sharing and niching methods revisited," *IEEE Transactions on Evolutionary Computation*, vol. 2, no. 3, pp. 97–106, 1998.

[23] P. S. Oliveto, D. Sudholt, and C. Zarges, "On the benefits and risks of using fitness sharing for multimodal optimisation," *Theoretical Computer Science*, vol. 773, pp. 53–70, Jun. 2019.

[24] X. Li, M. G. Epitropakis, K. Deb, and A. Engelbrecht, "Seeking multiple solutions: An updated survey on niching methods and their applications," *IEEE Transactions on Evolutionary Computation*, vol. 21, no. 4, pp. 518–538, Aug 2017.

[25] P. Darwen and Xin Yao, "A dilemma for fitness sharing with a scaling function," in *Proceedings of 1995 IEEE International Conference on Evolutionary Computation*, vol. 1, Nov 1995, pp. 166–.

[26] A. Martino, A. Rizzi, and F. M. Frattale Mascioli, "Distance matrix pre-caching and distributed computation of internal validation indices in k-medoids clustering," in *2018 International Joint Conference on Neural Networks (IJCNN)*, July 2018, pp. 1–8.

[27] ——, "Efficient approaches for solving the large-scale k-medoids problem: Towards structured data," in *Computational Intelligence: 9th International Joint Conference, IJCCI 2017 Funchal-Madeira, Portugal, November 1-3, 2017 Revised Selected Papers*, C. Sabourin, J. J. Merelo, K. Madani, and K. Warwick, Eds. Cham: Springer International Publishing, 2019, pp. 199–219.

[28] ——, "Efficient approaches for solving the large-scale k-medoids problem," in *Proceedings of the 9th International Joint Conference on Computational Intelligence - Volume 1: IJCCI,*, INSTICC. SciTePress, 2017, pp. 338–347.

[29] X. Xu, J. Jäger, and H.-P. Kriegel, "A fast parallel clustering algorithm for large spatial databases," *Data Mining and Knowledge Discovery*, vol. 3, no. 3, pp. 263–290, 1999.

[30] L. Baldini, A. Martino, and A. Rizzi, "Stochastic information granules extraction for graph embedding and classification," in *Proceedings of the 11th International Joint Conference on Computational Intelligence - Volume 1: NCTA, (IJCCI 2019)*, INSTICC. SciTePress, 2019, pp. 391–402.

[31] ——, "Towards a class-aware information granulation for graph embedding and classification," in *Computational Intelligence: 11th International Joint Conference, IJCCI 2019 Vienna, Austria, September 17-19, 2019 Revised Selected Papers*, To appear in.

[32] E. Maiorino, F. Possemato, V. Modugno, and A. Rizzi, "Noise sensitivity of an information granules filtering procedure by genetic optimization for inexact sequential pattern mining," in *Computational Intelligence*, J. J. Merelo, A. Rosa, J. M. Cadenas, A. Dourado, K. Madani, and J. Filipe, Eds. Cham: Springer International Publishing, 2016, pp. 131–150.

[33] Xingquan Zhu, Xindong Wu, A. K. Elmagarmid, Zhe Feng, and Lide Wu, "Video data mining: semantic indexing and event detection from the association perspective," *IEEE Transactions on Knowledge and Data Engineering*, vol. 17, no. 5, pp. 665–677, 2005.

[34] V. N. Gudivada, D. Rao, and V. V. Raghavan, "Chapter 9 - big data driven natural language processing research and applications," in *Big Data Analytics*, ser. Handbook of Statistics, V. Govindaraju, V. V. Raghavan, and C. Rao, Eds. Elsevier, 2015, vol. 33, pp. 203 – 238.