PRIFYSGOL
ABERYSTWYTH
UNIVERSITY

Aberystwyth University

A Hybrid Fuzzy Maintained Classification Method Based on Dendritic Cells
Chelly Dagdia, Zaineb; Elouedi, Zied

A Hybrid Fuzzy Maintained Classification Method Based on Dendritic Cells

Zaineb Chelly Dagdia[1,2] · Zied Elouedi[2]

Abstract

The dendritic cell algorithm (DCA) is a classification algorithm based on the behavior of natural dendritic cells (DCs). In literature, DCA has given good classification results. However, DCA was known to be sensitive to the order of the instance classes. To solve this limitation, a fuzzy DCA version was developed stating that the cause of such sensitivity is related to the DCA crisp classification task (hypothesis 1). In this paper, we hypothesize that there is a second possible cause of such DCA sensitivity which is related to the possible existence of noisy instances presented in the DCA signal data set (hypothesis 2). Thus, we aim, first of all, to test the trueness of the latter hypothesis, and second, we aim to develop an overall hybrid DCA taking both hypotheses into consideration. Based on hypothesis 1, our new DCA focuses on smoothing the crisp classification task using fuzzy set theory. Based on hypothesis 2, a data set cleaning technique is used to guarantee the quality of the DCA signal data set. Results show that our proposed hybrid fuzzy maintained algorithm succeeds in obtaining results of interest.

Keywords Evolutionary computing · Fuzzy set theory · Maintenance · Classification

1 Introduction

Evolutionary computing forms a family of algorithms inspired by the theory of biological evolution that solve various problems (Cui and Xiao 2014; Kaur et al. 2015). One recent addendum to the evolutionary class of algorithms is the dendritic cell algorithm (DCA) (Greensmith and Aickelin 2005). As the name suggests, DCA is based on the behavior

---

✉ Zaineb Chelly Dagdia
    chelly.zaineb@gmail.com

    Zied Elouedi
    zied.elouedi@gmx.fr

[1] Department of Computer Science, Aberystwyth University, Aberystwyth, UK

[2] LARODEC, Institut Supérieur de Gestion de Tunis, Tunis, Tunisia

of natural dendritic cells (DCs). The DCA aims to perform a binary classification task by correlating a series of informative signals, termed either danger or safe, inducing a signal data set with a sequence of repeating abstract identifiers: termed "antigens." Based on the induced DCA signal data set, the binary algorithm classifies in a crisp manner each antigen as being either anomalous, mapped as the DC mature context, or normal, mapped as the DC semi-mature context. The DCA uses this metaphor to decide if items within a presented set are members of class 1 or class 2.

Despite being successfully applied to a diverse range of binary classification problems (Greensmith and Aickelin 2006, 2007), it was noticed that the DCA performance is only observed when the algorithm is applied to ordered training data sets, where all class 1 examples are followed by all class 2 examples. This leads the DCA to be sensitive to the input class data order (Greensmith and Aickelin 2008). This critical point was partially handled in our first work, named the fuzzy dendritic cell method (FDCM) (Chelly and Elouedi 2010) by hybridizing the DCA with fuzzy set theory (Zadeh 1965). A main limitation with FDCM was that its input parameters had to be automatically generated instead of being introduced by an ordinary user. Thus, in Chelly and Elouedi (2015), we have developed an overall automated fuzzy DCA version based on generating automatically the parameters of the system, using a fuzzy clustering technique, to avoid false and uncertain values given by the ordinary user. The developed algorithm was non-sensitive to the input class data order. In Chelly and Elouedi (2015), we have hypothesized that one of the causes of the DCA's mentioned sensitivity is the non-clear separation between the already mentioned DC mature context and the DCA semi-mature context (first hypothesis). By adopting the fuzzy set component and a fuzzy clustering technique, the sensitivity problem was solved and, hence, the first hypothesis was confirmed.

Yet, we believe that there can be a second possible cause leading the DCA to be sensitive to the input class data order. Actually, as the DCA classification task depends on the induced signal data set, we hypothesize that the quality of this signal base can influence the classification task of the DCA that we will refer to as its behavior (second hypothesis). In fact, the DCA signal data set may contain misleading objects such as noisy, incoherent or redundant instances. Such a non-maintained and not-cleaned base may influence the DCA behavior leading the algorithm to be sensitive to the input class data order. Therefore, in this paper, we propose first of all to test the trueness of our second hypothesis, then, we will propose an overall hybrid fuzzy maintained DCA based on the two mentioned possible hypotheses in order to develop a more robust binary classifier.

This paper unfolds as follows: In Section 2, we present more details about the motivation of this work and its aims and scope; and in Section 3 we introduce the dendritic cell algorithm. The cleaning data set policies are presented in Section 4. In Section 5, we describe our fuzzy maintained dendritic cell method in detail. The experimental methodology is presented in Section 6 and the discussion and results carried out on binary data sets from the U.C.I. repository are given in Section 7. Finally, Section 8 presents the application of our proposed fuzzy maintained DCA to intrusion detection.

2 Aims and Scope

As already stated, in this paper, we aim first of all to test the trueness of our second mentioned hypothesis. Once confirmed and as we have already confirmed the trueness of our first hypothesis in Chelly and Elouedi (2015), we aim to develop a fuzzy maintained DCA

solution which is based on both hypotheses. Our solution, named clustering, outliers and internal cases detection (COID)-fuzzy dendritic cell method (COID-FDCM), is presented as a two-leveled hybrid fuzzy DCA classifier. COID-FDCM can be seen as an extension of our work presented previously in Chelly and Elouedi (2015).

Based on our second hypothesis, cleaning the DCA signal data set as a primary step becomes essential. Hence, in the top level of COID-FDCM, we use a cleaning policy to guarantee the quality of the DCA signal data set. Actually, several data set cleaning policies have been proposed in literature (Dasu and Johnson 2003). Most of these methods can offer an acceptable data set size reduction and satisfy classification accuracy; however, some of them are expensive to run and neglect the importance of the noisy instances elimination. For our proposed method, in order to clean the signal data set, we choose to apply the method named COID (Smiti and Elouedi 2010). COID seems appropriate to DCA since it is characterized by its capability of removing noisy and redundant instances as well as its ability to improve the classification accuracy and to offer a reasonable execution time. In fact, the choice of this policy is based on a rigorous analysis and a comparative study of different data set cleaning policies which will be later presented in our experimental setup section.

Based on our first hypothesis, our new method should be able to smooth the crisp classification task performed by the DCA. This will be handled by the use of fuzzy set theory (Zadeh 1965) in the second level of COID-FDCM. In fact, fuzzy set theory—applied to solve several problems in various real word problems—permits with linguistic terms a simple and natural description of problems which have to be settled rather than in terms of relationships between definite numerical values. Furthermore, our new model adopts a fuzzy clustering approach in order to generate automatically the extents and midpoints of the membership functions which describe the variables of COID-FDCM fuzzy part. Hence, we can avoid negative influence on the results when an ordinary user introduces such parameters. In Chelly and Elouedi (2015), we have made a comparative study of several fuzzy clustering techniques and selected the Gustafson–Kessel (GK) algorithm (Gustafson and Kessel 1979) as it has shown promising results in comparison to the rest of the compared techniques. Thus, COID-FDCM adopts GK to generate automatically the parameters of the fuzzy part of the algorithm. Indeed, we aim to compare the classification performance of our COID-FDCM proposed solution with a set of state-of-the-art classifiers, mainly with the standard DCA, as it has been recently introduced among the evolutionary class of binary algorithms, neural networks (Zhang 2000), decision trees (Gu et al. 2010), and support vector machines (Cortes and Vapnik 1995). It is, also, important to note that DCA has been subject to several modifications since its commencement. Works such as Chelly and Elouedi (2012a, b) focused on various algorithmic aspects of the DCA trying to make it a better classifier. Furthermore, other works such as Amaral (2011a, b) proposed fuzzy DCA versions based on some fuzzy inference systems able to combine the DCA set of signals or to reduce the amount of time taken to sample antigens from the antigen vector. Most importantly, we have to clarify that all these DCA modified versions have been applied to ordered data sets only and have not tackled the problem of the DCA as it is sensitive to the input class data order. Therefore, a comparison to these modified crisp and fuzzy versions, which are also sensitive to the input class data order, is not possible and, thus, we will focus our comparison on the mentioned state-of-the-art classifiers. Finally, we aim to show the promising classification results obtained by our proposed COID-FDCM through a concrete application example which is the intrusion detection problem and more precisely the application to the KDD 99 data set.

3 Overview of the Dendritic Cell Algorithm

Before describing the functioning of the algorithm, we give a general overview of the biological principles used by the DCA.

3.1 Biological Dendritic Cells

Dendritic cells are types of antigen-presenting cells. They can sense various signals that may be present in the tissue through receptors expressed on the cells surface. These receptors are sensitive to the concentration of signals received from their neighborhood. There are three main types of signals which are the pathogenic associated molecular patterns (PAMPs), the danger signals (DS) and the safe signals (SS). The relative proportions and potency of the different signals received lead either to a partial maturation state or to a full maturation state changing the behavior of the immature DCs (iDCs). iDC are the initial maturation state of a DC. Under the reception of safe signals, iDCs migrate to the semi-mature state (smDCs) and they cause antigen tolerance. iDCs migrate to the mature state (mDCs) if they are more exposed to danger signals and to PAMPs than to safe signals. They present the collected antigens in a dangerous context (Lutz and Schuler 2002). Swapping from one state to another is dependent upon the receipt of different signals throughout the iDC state:

- **PAMPs:** PAMPs are proteins expressed exclusively by bacteria, which can be detected by DCs and result in immune activation. The presence of PAMPs indicates definitely an anomalous situation.
- **Safe signals:** Signals produced via the process of normal cell death, namely, apoptosis. They are indicators of normality which means that the antigen collected by the DC was found in a normal context. Hence, tolerance is generated to that antigen.
- **Danger signals:** DS are indicators of abnormality but with a lower value of confidence than the PAMP signals.

3.2 Formalization of the Dendritic Cell Algorithm

By Algorithm 1, we present a generic form of the DCA based on DC biology. DCA has two types of input data which are signals from all categories and antigens. Signals are represented as vectors of real-valued numbers while antigens are the data item IDs. The DCA, as a binary classifier, has to classify each antigen either as normal (referred to as a semi-mature cell context) or as anomalous (referred to as a mature cell context). Thus, the algorithm output is the antigen context which is represented by the binary value 0 to mean that the antigen is likely to be normal, or 1 meaning that the antigen is likely to be anomalous. Formally and through the *initialization phase*, DCA performs data pre-processing where feature selection and signal categorization are achieved. DCA selects the most important features/attributes from the input training data set and assigns each selected attribute to its specific signal category (either as SS, as DS, or as PAMP). In other words, each attribute is mapped as a signal category based on the immunological definitions stated in Section 3.1. For instance, an attribute reflecting the number of error messages generated per second by a failed network connection can be categorized as a PAMP signal. This is because this feature, like a PAMP signal, is seen as a signature of abnormal behavior, e.g., an error message. For the DCA data pre-processing phase, either experts may be involved for feature selection and signal categorization or dimensionality reduction and statistical inference techniques may

be used. Formally, to perform data pre-processing, the standard DCA applies the principal component analysis (PCA). This task is achieved via the initialize-DC() function.

Algorithm 1 The Dendritic Cell Algorithm

 1: input : signals from all categories and antigen; (SS, PAMP, DS and antigens)
 2: output: antigen context values; (0/1)
 3: **For each** DC **Do**   /* Pre-processing & Initialization phase */
 4:    initialize-DC();
 5: **end**
 6: **While** CSM < mt **do**   /* Detection phase */
 7:    get-antigen();
 8:    get-signals();
 9:    calculate-inter();
10:    update-cumul();
11: **end while**
12: **if** smDC > mDC **then**   /*Context Assessment */
13:    cell-context = 0;
14: **else**
15:    cell-context = 1;
16: **end if**
17: **For each** antigen **do**   /* Classification phase */
18:    **if** cell-context == 1 **then**
19:       Nb-mature ++;
20:    **end if**
21: **end**
22: **For each** antigen **do**
23: MCAV = Nb-mature / Nb-antigen ;
24: **end**

After that, and throughout the *detection phase*, DCA has to generate a signal database by combining the input signals with the antigens. This combination is achieved via the use of both get-antigens() and get-signals() functions. The induced signal database rows represent the antigens to classify and the attributes represent SS, PAMP, and DS. The attribute values, for each antigen, are calculated based on specific equations that are detailed in Greensmith et al. (2010). Formally, this combination is achieved through using a population of artificial DCs. Using multiple DCs means that multiple antigens are sampled multiple times. Based on the induced signal database, the algorithm processes its input signals to get three cumulative output signal values known as the costimulatory molecule signal value (CSM), the semi-mature signal value ($smDC$), and the mature signal value (mDC). This task is performed through the use of the calculate-inter() function where these cumulative output signal values are calculated using a signal processing equation detailed in Greensmith et al. (2010) and a set of weights. These three DC output signals perform two roles which are, first, to determine if an antigen type is anomalous and, second, to limit the time spent sampling data. In fact, each DC in the population is assigned a migration threshold value (mt) upon its creation. So, if the value of CSM exceeds mt then the DC stops sampling antigens and signals, else the algorithm continues sampling and, also, keeps calculating and updating the values of CSM, $smDC$, and mDC via the update-cumul() function.

Now, through the *context assessment phase*, the DC forms a cell context that is used to perform antigen classification. In fact, the cumulative output signals of both $smDC$ and mDC are assessed and the one that has a greater/higher output signal is the one that becomes the cell context, either 1 or 0. This information is ultimately used in the generation of an anomaly coefficient which will be dealt with in the final step: the *classification phase*. More

precisely, the derived value for the cell context is used to derive the nature of the response by measuring the number of DCs that are fully mature. This generated number is represented by the mature context antigen value ($MCAV$). The $MCAV$ is calculated by dividing the number of times an antigen appears in the mature context, *Nb-mature*, by the total number of presentation of that antigen, *Nb-antigen*. Once the $MCAV$ is calculated for each antigen, the algorithm can perform its classification task. This is done by comparing the $MCAV$ of each antigen to an anomalous threshold. More precisely, those antigens whose $MCAVs$ are greater than the anomalous threshold are classified into the anomalous category while the others are classified into the normal one.

To have further details about the DCA, its strengths and limitations, how it can be applied to a sample training data set, its development pathway, and its application domains, we kindly invite the reader to refer to Chelly and Elouedi (2015, 2016).

4 Overview of the Data Set Cleaning Policies

The quality of the training data set is very important to generate accurate results and to have the possibility to learn models from the presented data. To achieve this, the misleading instances—especially noisy and redundant instances which affect negatively the quality of the results—have to be handled. To address this situation, data cleaning is a feasible way. The goal of the data set cleaning methods is to select the most representative information from the used training data set. This information can increase the model capabilities and generalization properties. In fact, many works dealing with data sets cleaning have been proposed in literature (Dasu and Johnson 2003). Most of them are based on updating a training data set by adding or deleting instances to optimize and reduce the initial base (Dasu and Johnson 2003). The data set cleaning policies include different operations such as redundant or inconsistent instances may be deleted, groups of instances may be merged to eliminate redundancy and improve reasoning power, instances may be re-described to repair incoherencies, signs of corruptions in the training data set have to be checked, and any abnormalities in the training data set which might signal a problem have to be controlled. A training data set which is not cleaned can give inaccurate data, and with these, users will make uninformed decisions. The quality of the cleaned new training data sets will be tested using a classification algorithm, such as k-NN. Among the data set cleaning policies proposed in literature, we can cite the following:

One type of cleaning methods is the selective reduction approach. This type starts with an empty set, selects a subset of instances from the original training data set, and adds it into the new data set, initially empty. We can mention among the most efficient selective reduction data methods shown in Dasu and Johnson (2003) the condensed nearest neighbor rule (CNN) proposed in Chou et al. (2006). CNN is a redundancy technique that incrementally builds an edited training data set from scratch. Formally, instances which are misclassified by the algorithm, k-NN for example and where $k > 1$, are removed from the original training data set while the remaining instances which are correctly classified are added to a new data set, initially empty. However, CNN suffers from some shortcomings as it is sensitive to noise. CNN can view the noisy instances as important exceptions and give an unsatisfying result. Another selective reduction method is the reduced nearest neighbor (RNN) rule (Gates 1972) which starts using the whole training set, S, as the initial reduced set, T. In other words, it starts with $S = T$ and removes each instance from S if such a removal does not cause any other instances in T to be misclassified by the instances remaining in S. The process is repeated until no further reduction can be achieved. Nevertheless, the iterative

process is very time-consuming if the original training set is large and it is computationally more expensive than the CNN method. The edited nearest neighbor rules (ENN) (Wilson 1972) is, also, a selective reduction method which removes all the misclassified instances from the training set. ENN keeps all the internal instances but deletes the border instances as well as the noisy instances, unlike the CNN algorithm. These methods are among the most efficient methods due to their balanced behavior in diversity and convergency.

Another type of data set cleaning methods is the deletion reduction method which addressed the problem of optimization. In fact, from a given training data set, these strategies are able to suppress "useless" instances from the initial training data set and bring the base to a specific number of instances. Hence, this type of reduction methods works on the same initial training data set without the use or the construction of a new data set where the correctly classified instances will be added. Some researchers advocate a random deletion or addition policy (Markovich and Scott 1988). In this policy, a random instance is removed from the training data set once the base size exceeds a threshold that should be defined a priori by the user. The random deletion method presents a limitation as, sometimes, the key instances may be deleted. Substantial performance improvements of this policy, in terms of the quality of the cleaned training data set, were presented in Markovitch and Scott (1993) by incorporating filtering processes. In Racine and Yang (1996, 1997), other deletion methods based on eliminating any kind of redundancy and inconsistency from the training data set were described. To clean the training data set, additional methods reducing the size of the training data set by focusing on its density were presented in Smyth and McKenna (1998). We can, also, mention the COID deletion method proposed in Smiti and Elouedi (2010). It uses the clustering technique to partition the initial training data set into multiple clusters in order to clean each cluster apart. It is believed that it is easier to clean each cluster apart instead of using the whole initial training data set. Then, COID applies outliers and internal instances detection methods, for each cluster, to reduce the size. This method aims at selecting instances which influence the quality of the data set. For each cluster, the instances of type outliers and the instances which are the nearest to the center of each cluster are kept and the rest of the instances are removed.

We believe that cleaning the training data set by removing instances may not necessarily lead to a degradation of the results. In fact, we have observed experimentally that a little number of instances can have performances comparable to those of the whole sample, and sometimes higher. To explain such an observation, some noise or repetitions in data could be deleted by removing instances. Based on a comparison of different data set cleaning policies, we have noticed that the COID method is an interesting technique among the other methods in terms of its capability of removing noisy and redundant instances as well as its ability to improve the classification accuracy and to offer a reasonable execution time. Thus, our proposed COID-fuzzy dendritic cell method will be based on the COID method for the signal data set cleaning process.

5 COID-FDCM: the Fuzzy Maintained Dendritic Cell Method

In this section, we describe our fuzzy hybrid model named COID-fuzzy dendritic cell method (COID-FDCM). A global view of the COID-FDCM architecture is presented in Fig. 1.

More precisely, COID-FDCM which is based on the two previously mentioned hypotheses has the same DCA steps except for the DCA detection phase and the DCA context assessment phase. The standard DCA detection phase is modified to be a COID-FDCM
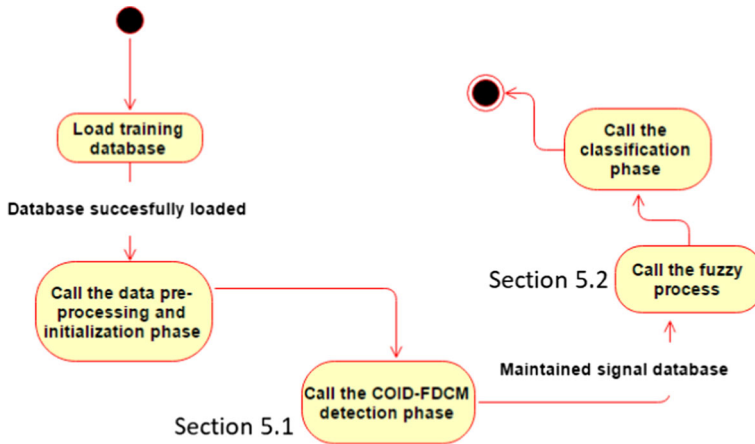
Fig. 1 The COID-FDCM architecture

detection phase where the COID technique is added to clean the signal data set. The DCA context assessment phase is replaced by a fuzzy procedure composed of five main steps:

1. Defining the inputs and the output of the system;
2. Defining the linguistic variables;
3. Constructing the membership functions;
4. Creating the rule base;
5. Assessing the fuzzy context.

We will mainly discuss the COID-FDCM detection phase and the fuzzy process as the rest of the algorithm steps, including the initialization phase and the classification phase, are performed the same as DCA and as described previously in Section 3.2.

5.1 COID-FDCM Detection Phase

5.1.1 COID-FDCM Detection Phase Basic Concepts

Once data pre-processing is achieved and through the detection phase, COID-FDCM has to generate the signal data set and to clean it. As previously stated, the signal data set may contain misleading objects especially noisy and redundant instances, in the sense that it affects negatively the quality of the algorithm classification results. To guarantee the algorithm performance, the process of cleaning the signal data set becomes essential. Thus, we apply the COID method as a first step in our new COID-FDCM algorithm. In fact, applying COID to the signal data set aims at eliminating its noisy and redundant instances. As a result, a new smaller signal base is constructed to represent the original one. To achieve this, COID performs two main steps—seen as two substeps of the COID-FDCM detection phase—which are:

- Partitioning the signal base into small groups.
- For each cluster, deleting antigens which are not outliers or internal cases.

As stated above, COID defines two important types of instances which should not be deleted from the original signal data set. These types of objects are described in Fig. 2.
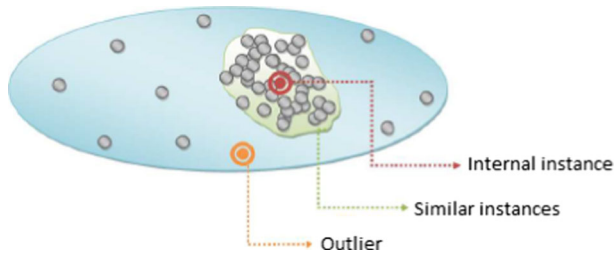
Fig. 2 Instance categories

- *Outlier* is an isolated instance. No other instance can replace an outlier or be similar to it. By eliminating outliers, the competence of the system will be negatively affected since there will not be any other instances that can solve the deleted outliers.
- *Internal instance* is one instance from a group of similar instances. Each instance from this group provides similar coverage to the other instances of the same group. Deleting any member of this group has no effect on the system's performance since the remaining instances offer the same value. However, deleting the entire group is tantamount to deleting an outlier instance as competence is reduced. Thus, we have to keep one instance from each group of similar instances.

Based on this idea, the COID method reduces the original signal data set by keeping only these two types of instances and eliminating the rest. For that, it uses a clustering method to create groups of similar instances. Then, for each cluster, it applies specific methods to detect outliers and others to select the internal instances which are defined as the nearest instances to the center of each cluster. For the signal base cleaning, COID processes as follows:

- For the clustering method, COID adopts the density-based spatial clustering of applications with noise (DBSCAN) method (Ester et al. 1996) since it offers minimal requirements of domain knowledge to determine the input parameters. Moreover, it can detect the noisy instances. Hence, the improvement of classification accuracy could be achieved. COID adopts DBSCAN to partition the initial input signal data set into multiple clusters to handle each cluster apart.
- For the selection of the internal instances, from each created cluster, COID calculates the Euclidean distance between the cluster's center, defined via the mean of the cluster instances, and each instance from the same cluster. Based on the calculated smallest distance, COID selects from each cluster a single instance which has the smallest distance with the center of its cluster. Hence, from each cluster, one internal instance is kept and the rest are deleted.
- For the outliers detection, COID applies two outliers detection methods to announce the univariate outliers and the multivariate outliers. For univariate outliers detection, COID uses the interquartile range (IQR) (Bussian and Härdle 1984) as it is a robust statistical method, e.g., beyond the 25th and 75th percentile values of the variable are considered outliers. For multivariate outliers detection, the Mahalanobis distance is used. Mahalanobis distance takes into account the covariance among the variables in calculating distances. Specifically, the Mahalanobis distances are calculated for all observations using the trimmed mean and the trimmed covariance matrix, which are (more) robust to the effect of extreme observations. With these measures, the problems of scale and

correlation are no longer an issue (Filzmoser et al. 2005). COID keeps these two types of outliers, from each cluster, as they are seen as specific instances where no other instances from the same cluster can replace them or be similar to them.

Consequently, the result of this phase is a new reduced and cleaned signal data set lacking noisy and redundant instances.

5.1.2 The Signal Base Maintenance Process

In this section, we provide a detailed description of how to maintain the signal data set. The relative workflow is, thus, illustrated in Fig. 3. First of all, throughout the COID-FDCM detection phase, the algorithm has to generate groups of similar antigens (Fig. 3b). This step is handled by the use of the DBSCAN clustering method. Once the DBSCAN detects noisy objects and partitions the signal base, COID-FDCM removes the detected noisy instances (Fig. 3c). The result of this process is a set of independent small signal databases (Fig. 3d).
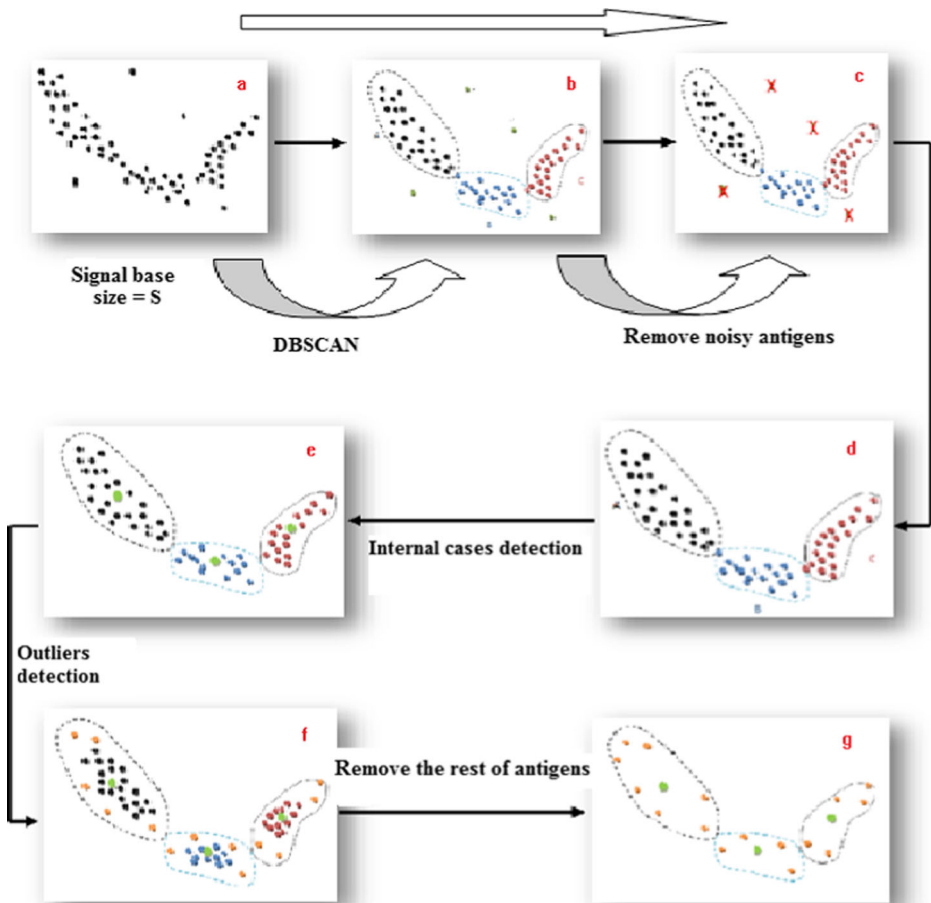

Fig. 3 Overview of the COID-FDCM maintenance process

The second step of the COID-FDCM detection phase is to select antigens which are close to the center of each cluster (Fig. 3e). After the detection of the internal antigens, COID-FDCM applies outliers detection methods to announce outliers (Fig. 3f). Finally, all unselected antigens, which are not outliers or internal cases, are deleted from the signal data set. Based on these steps, we obtain a reduced signal data set (Fig. 3g).

5.2 Fuzzy Process

5.2.1 Fuzzy System Inputs–Output Variables

Once the signal data set is cleaned using the COID policy, COID-FDCM processes the signal values to get the semi-mature and the mature values, defined previously as $smDC$ and mDC (Section 3.2). This process is performed the same as the standard DCA (Greensmith and Aickelin 2005). In order to describe each of these two contexts, linguistic variables are used. Two inputs and one output are defined. The semi-mature context, $smDC$, is denoted by C_s. The mature context, mDC, is denoted by C_m. C_s and C_m represent the input variables. The output variable of our algorithm reflects the final state "maturity" of a DC and is denoted by S_{mat}. Formally, the variables are defined as follows:

$$C_s = \{\mu_{C_s}(c_{s_j})/c_{s_j} \in X_{C_s}\} \tag{1}$$

$$C_m = \{\mu_{C_m}(c_{m_j})/c_{m_j} \in X_{C_m}\} \tag{2}$$

$$S_{mat} = \{S_{mat}(s_{mat_j})/s_{mat_j} \in X_{S_{mat}}\} \tag{3}$$

where c_{s_j}, c_{m_j}, and s_{mat_j} are, respectively, the elements of the discrete universe of discourse X_{C_s}, X_{C_m}, and $X_{S_{mat}}$. μ_{C_s}, μ_{C_m}, and $\mu_{S_{mat}}$ are, respectively, the corresponding membership functions.

5.2.2 Linguistic Variables

The term set, $T(S_{mat})$, interpreting S_{mat}, is defined as:

$$T(S_{mat}) = \{Semi-mature, Mature\} \tag{4}$$

Each term in $T(S_{mat})$ is characterized by a fuzzy subset in a universe of discourse $X_{S_{mat}}$. *Semi-mature* might be interpreted as an object (data instance) collected under safe circumstances, reflecting a normal behavior, and *Mature* as an object collected under dangerous circumstances, reflecting an anomalous behavior. Similarly, the input variables C_s and C_m are interpreted as linguistic variables with:

$$T(Q) = \{Low, Medium, High\} \tag{5}$$

where Q = C_s and C_m respectively.

5.2.3 Membership Function Construction

In order to specify the range of each linguistic variable, we have run the algorithm and we have recorded both semi-mature and mature values which reflect the ($smDC$) and (mDC) outputs generated by the DCA. Then, we picked up the minimum and maximum values of each of the two generated values to fix the borders of the range which are determined as follows:

$$min(range(S_{mat})) = min(min(range[C_m]), min(range[C_s])) \tag{6}$$

$$max(range(S_{mat})) = max(max(range[C_m]), max(range[C_s])) \tag{7}$$

Now, it seems important to fix the extents and midpoints of each membership function. We aim at generating automatically these parameters to avoid any negative influence on the results if they will be given by any ordinary user. In Chelly and Elouedi (2015), we have made a comparative study of several fuzzy clustering techniques and selected the Gustafson-Kessel algorithm (GK) as it has shown promise when hybridized with the fuzzy DCA version. Thus, we will use GK to generate automatically the parameters of the COID-FDCM fuzzy part.

To the recorded list of (mDC) and ($smDC$) values, we apply GK. It helps to build a fuzzy inference system by creating membership functions to represent the fuzzy qualities of each cluster. The output of this COID-FDCM third phase is a list of cluster centers and several membership grades generated by GK. Thus, the extents and midpoints of the membership functions which describe the system's variables are automatically determined.

5.2.4 The Rule Base Construction

A knowledge base, comprising rules, is built to support the fuzzy inference. The different rules of the fuzzy system are extracted from the information, listed below, reflecting the effect of each input signal on the state of a DC.

- **Safe signals:** an increase in value is a probable indicator of normality. High values of SSs can cancel out the effects of both PAMPs and DS.
- **Danger signals:** an increase in value is a probable indicator of damage, but there is less certainty than with a PAMP signal.
- **PAMPs:** an increase in value is a definite indicator of anomaly.

From the list above, we can generate the set of rules presented in Table 1 where all the mentioned signals are taken into account implicitly in the fuzzy system. Let us consider Rule(2) as an example: if the C_m input is set to the "Low" membership function and the second input C_s is set to the "Medium" membership function, then the "Semi-mature" context of the output S_{mat} is assigned. This could be explained by the effect of the high values of SSs (which lead to the semi-mature context) that cancel out the effects of both PAMPs and DSs (which both lead to the mature context). The same reasoning is affected to the rest of the rules.

Table 1 The fuzzy rule base

Rule number	Description
Rule(1):	If (C_m is Low) and (C_s is Low) then (S_{mat} is Mature)
Rule(2):	If (C_m is Low) and (C_s is Medium) then (S_{mat} is Semi-mature)
Rule(3):	If (C_m is Low) and (C_s is High) then (S_{mat} is Semi-mature)
Rule(4):	If (C_m is Medium) and (C_s is Low) then (S_{mat} is Mature)
Rule(5):	If (C_m is Medium) and (C_s is Medium) then (S_{mat} is Semi-mature)
Rule(6):	If (C_m is Medium) and (C_s is High) then (S_{mat} is Semi-mature)
Rule(7):	If (C_m is High) and (C_s is Low) then (S_{mat} is Mature)
Rule(8):	If (C_m is High) and (C_s is Medium) then (S_{mat} is Mature)
Rule(9):	If (C_m is High) and (C_s is High) then (S_{mat} is Mature)

5.2.5 The Fuzzy Context Assessment

Our COID-FDCM is based, first, on the *max–min* composition method which is also known as the "Mamdani" method and, second, on the *centroid* method as a defuzzification mechanism. Once the inputs are fuzzified and the output (centroid value) is generated, the cell context has to be fixed by comparing the generated output value (issued from the fuzzification step) to the middle of the S_{mat} range, e.g., the min and max range of S_{mat}. In fact, if the centroid value generated is greater than the middle of the output range, then the final context of the data instance is "Mature" (set to 1), indicating that the collected antigen may be anomalous, otherwise the antigen collected is likely to be normal (set to 0).

Once the context is fixed for all data item IDs (antigens), the classification phase has to be performed. Same as DCA, the derived values for the cell contexts (1 or 0) are used to derive the nature of the response by measuring the number of DCs that are fully mature and are represented by the MCAVs. To perform classification, a threshold which is generated automatically from the data must be applied to the MCAVs. In this case and as we are focusing on binary classification problems, the distribution of data between class 1 (anomalous data items) and class 2 (normal data items) is used and reflects the threshold rate to define. The threshold (at) can be defined as $at = an/tn$, where an is the number of anomalous data items and tn is the total number of data items.

6 Experimental Methodology

6.1 Experimental Hypotheses

As previously stated, the DCA suffers from being sensitive to the input class data order. In this section, we try to investigate the reasons of such a limitation. The list below presents the hypotheses of the causes of the DCA sensitivity to the input class data order.

- **H1:** The DCA sensitivity is related to the crisp separation between the normal context (referred as the semi-mature context) and anomaly context (referred as the mature context).
- **H2:** The DCA sensitivity is related to the quality of the signal data set which contains misleading objects such as noisy or redundant instances.

We have divided our experimental analysis and results section into four main parts.

1. First of all, we will show that the selected COID method that is hybridized with our proposed COID-FDCM version is an interesting technique among other data set cleaning techniques proposed in literature.
2. Secondly, we will test the trueness of H2 as H1 was confirmed in Chelly and Elouedi (2015) and as it led to the development of the $FCDCM_{GK}$ algorithm. Let us recall that $FCDCM_{GK}$ (Chelly and Elouedi 2015) is the fuzzy DCA version based only on H1 and which uses the GK algorithm. In this part, we will compare the classification results of DCA when it is applied to the original signal data set with the results obtained from the DCA when it uses the cleaned reduced signal data set, case we dub COID-DCA. COID-DCA is a modified DCA version based only on H2. We will also analyze the behavior of our new COID-DCA to show that the algorithm does depend neither on such transitions nor on ordered data sets contrary to the DCA. Thus, we can conclude that H2 is true and that by cleaning the signal base using COID, the problem related

to the sensitivity to the input class data order is solved. To achieve this, three different data orders are used. Experiment 1 uses a one-step data order. Here, data are ordered continuously, i.e., all class 1 data items are processed followed by all class 2 data items. In experiment 2, data are partitioned into three sections resulting in a two-step data order. The data comprising class 1 is split into two sections and the class 2 data is embedded between the class 1 partitions. Experiment 3 consists of data randomized between class 1 and class 2.

3. Third and once H2 is confirmed, we will show the performance of our proposed COID-FDCM which takes into account both H1 and H2. We will compare the results obtained from $FCDCM_{GK}$ (Chelly and Elouedi 2015) and COID-DCA with the COID-FDCM generated results. This is to show that if we take both hypotheses into consideration we can generate a more robust binary classifier.

4. Finally, we will show that our COID-FDCM model outperforms not only the standard DCA but also other well-known state-of-the-art classifiers.

6.2 Parameter Description and Evaluating Criteria

We are focusing on binary classification problems, and to test the performance of the algorithms, different experiments are performed using two-class data sets from the UCI machine learning repository (Asuncion and Newman 2007) which is a collection of benchmark data sets for regression and classification tasks. The used training data sets are described in Table 2.

For data pre-processing and for DCA, $FCDCM_{GK}$, COID-DCA, and COID-FDCM, the standard deviation of each attribute, for each training data set, is calculated. More precisely, for the feature selection sub-step, the attributes with the highest standard deviations are used as this allows for an exploration of the properties of the algorithm through having more variable data. For the signal categorization sub-step, the attribute having the lowest standard deviation in the selected set of attributes is used to derive the PAMP and safe signals, making it the "most certain" signal. The rest of the attribute set is used to calculate the DS values. The signal value derivation process is based on specific equations as shown in Greensmith (2007). Each data item is mapped as an antigen with the value of the antigen equals to the data ID of the item. All featured parameters are derived from empirical immunological data.

	Database	Ref	No. of instances	No. of attributes
Table 2 Description of the training data sets	Sonor	SN	208	61
	Molecular-Bio	Bio	106	59
	Cylinder Bands	CylB	540	40
	Chess	Ch	3196	37
	Ionosphere	IONO	351	35
	Sick	Sck	3772	30
	Horse Colic	HC	368	23
	German-Credit	GC	1000	21
	Labor Relations	LR	57	16
	Red-White-Win	RWW	6497	13

A population of 100 cells is used and 10 DCs sample the antigen vector each cycle. The migration threshold of an individual DC is set to 10 to ensure this DC to survive over multiple iterations. To perform anomaly detection, a threshold which is automatically generated from the data is applied to the MCAVs. The MCAV threshold is derived from the proportion of anomalous data instances of the whole data set. Items below the threshold are classified as class 1 and above as class 2. The resulting classified antigens are compared to the labels given in the original training data sets. For each experiment, the results presented are based on mean MCAV values generated across 10 runs. For DBSCAN, the used measure is the Euclidian distance and the suggested values are $MinPts = 4$ and $Eps = 0.2$. We choose these DBSCAN parameters as in Ester et al. (1996) it was shown that setting the $MinPts$ to 4 is a "good" choice while in Yarifard and Yaghmaee (2008) it was shown that when Eps is between 0.05 and 0.20, better results can be produced. To measure the performance of the algorithms, we will use the following criteria:

1. The rate of the size reduction (S): denotes the percentage of the number of instances which are kept among the whole set of instances presented initially in the signal base.
2. Sensitivity, specificity, accuracy: These criteria are defined as:

$$Sensitivity = \frac{TP}{TP + FN} \tag{8}$$

$$Specificity = \frac{TN}{TN + FP} \tag{9}$$

$$Accuracy = \frac{TP + TN}{TP + TN + FN + FP} \tag{10}$$

where TP, FP, TN, and FN refer respectively to true positive, false positive, true negative, and false negative.
3. Execution Time (t): This quantifies the amount of time taken by an algorithm to run. The execution time is measured in seconds (s).

To evaluate the performance of the data set cleaning techniques, including COID, CNN, RNN, IB2, and IB3, the accuracy is based on the k-nearest neighbor algorithm (k-NN) based on the use of the Euclidean distance and where $k = 1$. 1-NN is used to test all the mentioned data set cleaning policies. All experiments are run on a Sony Vaio G4 2.67 Ghz machine.

7 Experimental Results and Discussion

7.1 Results and Analysis of the Data Set Cleaning Policies

To measure the performance of the COID method and to discover its characteristics, we compare it to other well-known data set cleaning techniques. The selected algorithms are among the most efficient ones shown in Dasu and Johnson (2003). The used algorithms are the condensed nearest neighbor (CNN) algorithm (Chou et al. 2006), the reduced nearest neighbor (RNN) technique (Gates 1972), and the instance-based learning (IBL) schemes (IB2 and IB3) (Aha et al. 1991). All these algorithms are run on the previous training data sets presented in Table 2. The results of this comparison are presented in Table 3.

From Table 3, we can remark that the reduction rate (S%) obtained using the COID maintenance method is notably better than the one provided by the other maintenance policies in most data sets. For instance, for the Ch data set, COID keeps about 52.79% of the data instances and that is a huge difference compared to the initial Ch database with 100%

Table 3 Comparing data set cleaning schemes

Database	Evaluation criteria	DCA	COID	CNN	RNN	IB2	IB3
	S (%)	100	98.00	78.00	65.00	67.00	88.00
SN	Accuracy (%)	77.88	76.92	62.50	68.75	69.80	73.00
	Execution time (s)	1.36	1.25	1.14	1.30	1.31	1.29
	S (%)	100	90.00	71.10	71.70	89.00	89.00
Bio	Accuracy (%)	41.50	40.56	37.95	36.32	38.00	38.00
	Execution time (s)	0.27	0.24	0.22	0.25	0.26	0.26
	S (%)	100	86.52	96.70	96.60	98.00	93.98
CylB	Accuracy (%)	92.38	93.51	86.52	76.50	72.95	72.95
	Execution time (s)	11.31	10.12	11.04	11.13	11.09	10.66
	S (%)	100	52.79	62.79	67.46	77.00	85.07
Ch	Accuracy (%)	93.86	94.61	91.52	90.04	89.23	89.59
	Execution time (s)	183.96	104.88	168.71	166.08	170.24	166.21
	S (%)	100	56.00	64.39	62.08	62.12	61.00
IONO	Accuracy (%)	94.58	95.44	87.40	89.19	87.39	88.26
	Execution time (s)	7.79	4.06	5.7	5.0	5.3	4.8
	S (%)	100	68.28	79.26	77.50	75.40	71.20
Sck	Accuracy (%)	94.11	94.83	81.29	81.66	89.00	83.21
	Execution time (s)	256.98	162.08	246.34	229.12	208.62	198.77
	S (%)	100	67.20	68.04	79.62	67.00	76.00
HC	Accuracy (%)	83.96	87.77	84.97	83.74	86.66	84.54
	Execution time (s)	10.04	7.14	7.66	8.89	7.08	8.03
	S (%)	100	90.40	98.00	96.00	94.40	91.20
GC	Accuracy (%)	87.00	88.10	73.68	63.16	73.26	68.42
	Execution time (s)	61.32	43.15	45.24	45.23	48.97	45.07
	S (%)	100	87.00	67.00	77.04	73.70	72.96
LR	Accuracy (%)	84.21	82.45	61.02	60.24	79.78	78.09
	Execution time (s)	0.50	0.33	0.25	0.19	0.12	0.18
	S (%)	100	76.92	77.38	77.80	89.00	86.79
RWW	Accuracy (%)	97.87	97.90	97.40	96.26	97.29	97.45
	Execution time (s)	513.97	326.75	394.66	409.52	460.03	448.21

antigens. Comparing the COID reduction rate to the rest of the maintenance policies rates, on the same database, the rates are 62.79%, 67.46%, 77.00%, and 85.07% for CNN, RNN, IB2, and IB3, respectively.

We can, also, notice that in some data sets the reduction rate of COID is less than the rates of the other techniques. While focusing on these data sets, we can notice that there are few antigens, and consequently, we can conclude that if we apply a maintenance technique to a small size signal base then the classification accuracy of the algorithm will be negatively affected and that is seen from the results displayed in Table 3. For example, when applying the COID technique to the LR database (57 antigens), the accuracy of the COID-DCA algorithm is set to 82.45%. However, when applying DCA using the whole signal base, the

accuracy of DCA is set to 84.21%. Same remark is noticed when applying the rest of the maintenance policies to the DCA signal data set where we will obtain less classification results in comparison to the initial whole set of the signal database.

Conversely, for the rest of the databases, we notice that the classification accuracy of COID-DCA is even better than the DCA initial results when using 100% of the signal base. For instance, when applying the COID-DCA to the HC base, the classification accuracy of the algorithm is set to 87.77% which is more important than the classification accuracy of the DCA which is set to 83.96%. Now, regarding the execution time of the hybrid algorithms, we can notice that in most data sets COID takes less time to process in comparison to the rest of the DCA hybrid maintenance policies.

From the obtained results, we have shown that the COID method is an interesting maintenance technique among others proposed in literature. We focused on its efficiency in terms of shrinking the size of databases by removing their "useless" instances, its reasonable and acceptable execution time, and its ability to generate satisfactory classification results among the other studied techniques. These important characteristics are the base for choosing COID as an appropriate technique to use in our proposed method, COID-FDCM, in order to maintain the DCA signal database.

7.2 Results and Discussion About H2: Comparing DCA and COID-DCA

Let us remind that previous examinations with DCA, in Greensmith and Aickelin (2005), show that the misclassifications occur exclusively at the transition boundaries. Hence, DCA makes more mistakes when the context changes multiple times in a quick succession (2-Step and Random experiments) which is not the case when data are ordered between the classes (1-Step experiment). We hypothesize that this shortcoming is related to the quality of the DCA signal data set. Thus, we apply the COID method to clean it.

From Table 4, we can notice that by maintaining the DCA signal database using the COID technique, the percentage of classification accuracy is nearly stable between the three data orders, i.e., 1-Step, 2-Step, and Random experiments. Thus, the limitation of the algorithm which consists of being sensitive to the input class data order is overcome. From this observation, we can affirm that the standard DCA drawback is related to the quality of the DCA signal database, and by maintaining it using the COID method, the problem is solved. Consequently, our second hypothesis (H2) is confirmed.

More precisely, from Table 4, it is clearly noticed that the classification accuracy of the COID-DCA algorithm is nearly stable through the three realized experiments with comparison to the accuracies of the standard DCA results which are decreasing from the ordered contexts (1-Step experiment) to the disordered contexts (2-Step and Random experiments). This phenomenon could be explained by the fact that the original signal database contains noisy and redundant instances which affect negatively the performance of the algorithm. Thus, by using an appropriate maintenance policy, the COID method, we can guarantee the quality of the database leading to a stable DCA classifier.

For instance, when applying the COID-DCA method to the SN database, the accuracy of the algorithm is around 76.92% and 76.44%. Nevertheless, when applying the DCA to the same database, the accuracy of the algorithm decreases from 77.88 to 69.71%. This high value of the DCA accuracy (77.88%) in case of an ordered database is explained by the appropriate use of this algorithm only in an ordered case. From the 2-Step experiment to the Random one, the DCA accuracy decreases from 74.51 to 69.71%. This behavior shows that the DCA is sensitive to the input class data order which confirms the results obtained from literature. However, this problem is solved when using the COID-DCA since we notice a

Table 4 Experimental measures: accuracy (%)

Database	DCA			COID-DCA		
	1-Step	2-Step	Random	1-Step	2-Step	Random
SN	77.88	74.51	69.71	76.44	76.92	76.92
Bio	41.50	40.56	39.62	40.56	40.56	40.56
CylB	92.38	90.55	87.03	93.51	93.51	93.51
Ch	93.86	93.05	90.73	94.58	94.61	94.61
IONO	94.58	78.63	66.09	95.44	95.15	95.44
Sck	94.11	92.78	91.99	94.83	94.83	94.83
HC	83.96	81.52	80.16	87.77	87.77	87.77
GC	87.00	86.80	86.50	88.10	87.90	87.10
LR	84.21	82.45	78.94	82.45	82.45	82.45
RWW	97.87	96.96	95.42	97.90	97.90	97.90

stability in the algorithm classification results through the different experiments realized. Another example can be the RWW database where the accuracy of the algorithm is stable and set to 97.90%.

To sum up, we could approve that the second cause of the DCA sensitivity to the input class data order is related to the quality of its signal database. We have developed COID-DCA which is seen as a stable DCA classifier generating stable classification results through the variation of the input class data orders. From the obtained results, we can confirm the trueness of H2.

7.3 Results and Discussion About COID-FDCM

Based on the previous COID-DCA results and the results obtained from our first work $FCDCM_{GK}$ (Chelly and Elouedi 2015), we have confirmed both H1 and H2. Thus, we have developed the COID-FDCM algorithm which is based on both hypotheses. In this section, we try to show that if we take into consideration both H1 and H2, the performance of our proposed COID-FDCM algorithm can be improved in comparison to $FCDCM_{GK}$ and COID-DCA which are based on only one hypothesis, H1 and H2 respectively. We approach this by the development of an overall fuzzy maintained algorithm, COID-FDCM, which applies a cleaning data set policy, COID, to guarantee the quality of the DCA signal data set, smoothes the crisp separation between the DC contexts using fuzzy set theory, and generates automatically the parameters of the system using the GK clustering technique. Since we have shown that the previously developed $FCDCM_{GK}$ and COID-DCA algorithms are no more sensitive to the input class data order, in this section, we have run the algorithms to unordered data sets to test their classification performance in comparison with COID-FDCM. We will compare the three algorithms in terms of specificity, sensitivity, accuracy, and execution time. From Tables 5 and 6, and in most databases, it is clearly noticed that our COID-FDCM has given good results in terms of the mentioned comparison criteria while outperforming both $FCDCM_{GK}$ and COID-DCA. Thus, we can conclude that it is more appropriate and reasonable to take into consideration both hypotheses rather than only one.

Based on the results presented in Tables 5 and 6, we can notice that COID-FDCM generates better classification results in comparison to COID-DCA and $FCDCM_{GK}$. For

Table 5 Experimental measures: sensitivity and specificity

Database	Sensitivity (%)			Specificity (%)		
	$FCDCM_{GK}$	COID-DCA	COID-FDCM	$FCDCM_{GK}$	COID-DCA	COID-FDCM
SN	85.58	79.27	83.78	79.38	74.22	78.35
Bio	50.94	36.84	47.36	49.05	44.89	48.97
CylB	92.50	93.42	94.73	94.55	93.58	94.87
Ch	94.66	94.36	95.20	94.95	94.89	95.28
IONO	97.77	96.82	98.41	95.23	94.66	96.88
Sck	96.53	96.10	98.26	95.08	94.74	95.98
HC	92.59	91.66	93.98	85.52	82.23	86.18
GC	84.67	83.94	92.33	90.63	89.66	91.73
LR	70.00	80.00	65.00	78.37	83.78	75.67
RWW	98.18	98.04	98.50	98.74	97.56	98.99

instance, by applying our COID-FDCM to the HC database, the accuracy of our algorithm is set to 90.76%. However, when applying the COID-DCA to the same database, the accuracy of the algorithm is set to 87.77%. From these results, we can conclude that COID-FDCM produces better classification results than COID-DCA which is only based on H2. Indeed, by applying $FCDCM_{GK}$ to the same database, the accuracy of the algorithm is set to 89.67%. Again, we can remark that COID-FDCM generates better results than those of the $FCDCM_{GK}$ which is only based on H1. Same remark is noticed for the specificity and the sensitivity criteria. These results confirm that our COID-FDCM, which is based on H1 and H2, outperforms the results generated by both COID-DCA and $FCDCM_{GK}$ which are based on only one hypothesis, either H1 or H2.

From Table 6, we can also notice that our proposed COID-FDCM algorithm is characterized by its lightweight in terms of running time. In fact, COID-FDCM is characterized by its short time of process in comparison to $FCDCM_{GK}$, but it needs more time to process than COID-DCA. More precisely, the fact of reducing the size of the original DCA signal

Table 6 Comparison in terms of execution time (s) and accuracy

Database	Time (s)			Accuracy (%)		
	$FCDCM_{GK}$	COID-DCA	COID-FDCM	$FCDCM_{GK}$	COID-DCA	COID-FDCM
SN	4.01	1.25	3.11	82.69	76.92	81.25
Bio	3.06	0.24	2.51	50.00	40.56	48.11
CylB	13.02	10.12	11.49	93.75	93.51	94.81
Ch	188.65	104.88	120.72	94.80	94.61	95.24
IONO	14.91	4.06	8.54	96.86	95.44	97.43
Sck	277.23	162.08	231.19	95.17	94.83	96.12
HC	12.98	7.14	8.69	89.67	87.77	90.76
GC	63.88	43.15	49.31	89.00	88.10	91.90
LR	0.63	0.33	0.37	75.43	82.45	71.92
RWW	521.32	326.75	441.08	98.32	97.90	98.63

database decreases the execution time of our COID-FDCM comparing it to the $FCDCM_{GK}$ one as this latter algorithm is applied to the entire non-maintained database. Nevertheless, COID-FDCM needs more time to process than COID-DCA since it uses two extra components which are the fuzzy set concept and the GK clustering algorithm. For instance, applying the algorithms to the IONO database the amount of time taken by COID-FDCM to process is set to 8.54 s which is less than the time taken by $FCDCM_{GK}$ which is set to 14.91 s. Nevertheless, COID-FDCM processes a bit longer than COID-DCA which takes 4.06 s to process.

From these results, we can say that COID-FDCM can be seen as an interesting binary classifier capable of generating satisfactory classification results. We will, also, show that COID-FDCM outperforms other well-known classifiers in terms of classification accuracy. This will be discussed in what follows.

7.4 Comparison with State-of-the-Art Recent Methods

In this section, we will compare the classification results of our proposed COID-FDCM with well-known classifiers which are the support vector machine (LibSVM), multilayer perceptron (MLP), the decision tree (C4.5), $FCDCM_{GK}$ (Chelly and Elouedi 2015), COID-DCA, FDCM (Chelly and Elouedi 2010), RC-DCA (Chelly and Elouedi 2012a), and RST-DCA (Chelly and Elouedi 2012b). FDCM is the standard fuzzy DCA version which is applied to non-ordered data sets. RC-DCA is the crisp DCA version based on rough set theory for data pre-processing and is based on the use of different signals for signal categorization. RST-DCA is, also, a crisp DCA version based on rough set theory for data pre-processing but based on the idea of using the same feature to represent both SS and DS and a combination of features to represent the PAMP signal.

All comparisons are made in terms of the mean average of accuracies on the used training data sets, presented previously in Table 2. The parameters of LibSVM, MLP, and C4.5 are set to the most adequate parameters to these algorithms using the Weka-3-8-1 software and a 10 cross-validation approach is applied. We will, also, add the classical DCA to the comparison in order to show its performance when it is applied to ordered training data sets and when it is applied to non-ordered data sets. To do so, we have ordered the label classes of the used training data sets in order to get all class 1 data items followed by all class 2 data items, and then we have applied the DCA. We have named this case DCA_O. Secondly, we have applied DCA to non-ordered data sets, case named DCA_{NO}. On these non-ordered data sets, we have applied the rest of the classifiers: except for RST-DCA and RC-DCA as they should be applied to ordered class data sets. This comparison is presented in Fig. 4.

Figure 4 shows that the standard DCA when applied to ordered training data sets, DCA_O, outperforms SVM, ANN, and DT in terms of overall classification accuracy. This is an interesting point which is, unfortunately, not seen when the algorithm is applied to non-ordered data sets, DCA_{NO}. This confirms the results shown in literature about the DCA sensitivity to the input class data order. However, if we look to the DCA classifier when applied to ordered data sets, we confirm that it is capable of producing high and satisfactory classification results in comparison to the state-of-the-art mentioned classifiers. Secondly, we can notice that RC-DCA outperforms both RST-DCA and DCA_O confirming the results obtained in Chelly and Elouedi (2013). This is because, first, RC-DCA is based on rough set theory of data pre-processing unlike DCA_O which is based on PCA and, second, because it is based on the concept of using different features for different signals in the signal categorization step, unlike both DCA_O and RST-DCA. Indeed, from Fig. 4, we can notice that both COID-DCA and FDCM have similar performance confirming the effect of fuzzy set
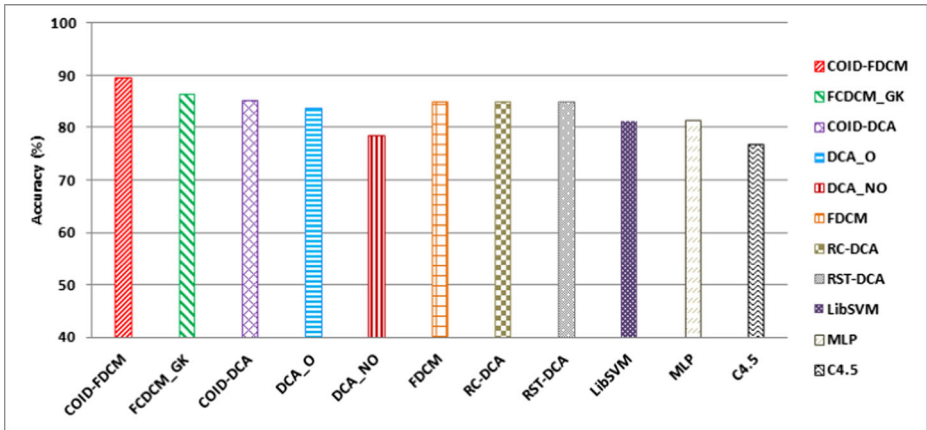
Fig. 4 Comparison of classifiers' mean average accuracies on the used binary data sets

theory and the database cleaning technique on the classification results of the algorithm. Yet, $FCDCM_{GK}$ outperforms these two algorithms as it includes the GK algorithm for parametrization as well as fuzzy set theory. Most importantly, from Fig. 4, we can notice that our developed COID-FDCM fuzzy maintained DCA version outperforms the mentioned classifiers including the classical DCA when applied to ordered bases, DCA_O, RC-DCA, RST-DCA, FDCM, COID-DCA, SVM, ANN, and DT in terms of overall accuracy.

Consequently, we have obtained a novel immune-inspired fuzzy maintained model making the standard DCA a better classifier by generating pertinent and more reliable classification results.

8 Application of the Proposed Solution to Intrusion Detection

As an immune-inspired algorithm, the dendritic cell algorithm produces promising performance in several application domains and specifically in the field of anomaly detection. Just like the DCA, our proposed COID-FDCM binary classifier can be applied to any of these applications. In this section, we illustrate the application of our newly proposed COID-FDCM algorithm to a real-world intrusion detection domain, the KDD Cup 1999 data set (Asuncion and Newman 2007). As COID-FDCM is seen as an extension of our previous work $FCDCM_{GK}$ (Chelly and Elouedi 2015), we will use the same data set as an application domain for comparison as well as the same experimental details in Chelly and Elouedi (2015).

In the realized experiment, the size of the DC population is set to 100 and it remains constant as the system runs. The migration threshold of an individual DC is a random value between 100 and 300, and this is to ensure that this DC survives over multiple iterations. The MCAV of an antigen type is calculated based on the labels of the original data set: normal is equivalent to context value 0 and anomalous is equivalent to context value 1. The MCAV threshold is derived from the proportion of anomalous data instances of the data set. The classification results of our COID-FDCM proposed algorithm, which is applied to the non-ordered KDD 99 data set, are compared with the results obtained from the standard DCA version applied to the ordered data set and to $FCDCM_{GK}$ (Chelly and Elouedi 2015). We
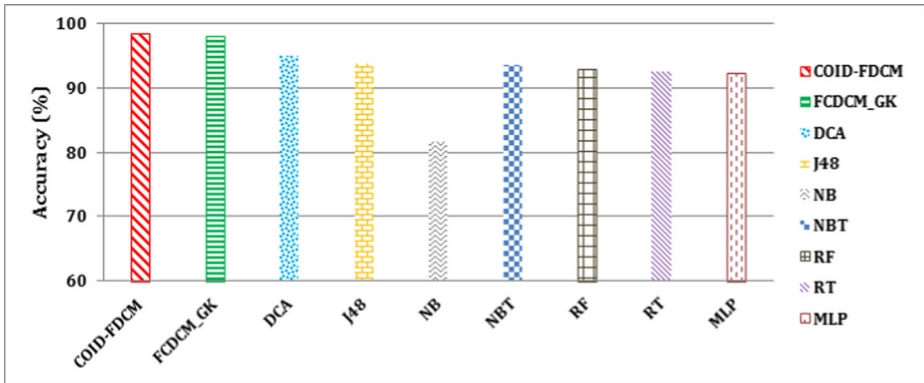
Fig. 5 The performance of the algorithms on the KDD 99 data set

have also compared the performance of COID-FDCM to seven widely used machine learning techniques namely J48 decision tree learning (J48) (Quinlan 1993), naive Bayes (NB) (John and Langley 1995), NBTree (NBT) (Kohavi 1996), random forest (RF) (Breiman 2001), random tree (RT) (Aldous 1991), multilayer perceptron (MLP) (Ruck et al. 1990), and support vector machine (SVM) (Chang and Lin 2001). For the experiments, we applied Weka's default values as the input parameters of these methods. For each single experiment, 10 runs are performed and the final result is the average of the 10 runs. The comparison of these algorithms, which is in terms of classification accuracy, is presented in Fig. 5.

From Fig. 5, we can notice that the classification performance of the standard DCA is comparable to J48 decision trees. We can also remark that DCA outperforms the rest of the classifiers in terms of classification accuracy. We, also, highlight the fact that the "best" classification performance is rendered by our proposed COID-FDCM. It is also important to mention that the classification accuracy given by COID-FDCM is slightly better than the one given by $FDCM_{GK}$. These COID-FDCM promising results are explained by the appropriate application of our COID-FDCM and $FCDCM_{GK}$ algorithms in the intrusion detection field. Indeed, the interesting characteristics of COID-FDCM are supported by the algorithm capability to clean the data first, and second to classify the items appropriately.

9 Conclusion and Future Works

In this paper, we have presented an investigation on the causes of the standard DCA limitation as it is sensitive to the input class data order. We proposed two possible hypotheses for this investigation and both H1 and H2 were confirmed. Based on these hypotheses, we have developed a new fuzzy hybrid evolutionary algorithm named COID-fuzzy dendritic cell method (COID-FDCM). COID-FDCM has been tested on machine learning data sets. Experimental results have demonstrated the effectiveness of our proposed approach by producing more accurate and better classification results in comparison to the DCA versions as well as to recent and well-known state-of-the-art classifiers.

A number of future directions can be proposed for the COID-FDCM. Future works will include the application of the type-2 fuzzy set theory to smooth more the algorithm crisp separation between the two contexts. We can, also, be interested in the introduction of variable weights for the COID policy and to study its effectiveness in our new model.

References

Aha, D., Kibler, G., Albert, M.K. (1991). Instance-based learning algorithms. *IEEE Transactions on Fuzzy Systems*, *1*, 37–66.

Aldous, D. (1991). The continuum random tree. In *The annals of probability, pp. 11728*.

Amaral, M. (2011a). Fault detection in analog circuits using a fuzzy dendritic cell algorithm. In *Proceedings of the 10th international conference on artificial immune systems* (pp. 18–21). Springer.

Amaral, M. (2011b). Finding danger using fuzzy dendritic cells. In *Proceedings workshop on hybrid intelligent models and applications* (pp. 21–27). IEEE.

Asuncion, A., & Newman, D. (2007). UCI machine learning repository. http://www.ics.uci.edu/mlearn/.

Breiman, L. (2001). Random forests. *Machine learning*, *45*(1), 5–32.

Bussian, B., & Härdle, M. (1984). Robust smoothing applied to white noise and single outlier contaminated raman spectra. *Applied Spectroscopy*, *383*, 309–313.

Chang, C., & Lin, C. (2001). Libsvm: a library for support vector machines. In *Software available at* http://www.csie.ntu.edu.tw/cjlin/libsvm.

Chelly, Z., & Elouedi, Z. (2010). FDCM: a fuzzy dendritic cell method. In *Proceedings of the 9th international conference of artificial immune systems* (pp. 102–115). Springer.

Chelly, Z., & Elouedi, Z. (2012a). RC-DCA a new feature selection and signal categorization technique for the dendritic cell algorithm based on rough set theory. In *Proceedings of the 11th international conference on artificial immune systems* (pp. 152–165). Springer.

Chelly, Z., & Elouedi, Z. (2012b). RST-DCA a dendritic cell algorithm based on rough set theory. In *Proceedings of the 19th international conference on neural information processing* (pp. 480–487). Springer.

Chelly, Z., & Elouedi, Z. (2013). QR-DCA A new rough data pre-processing approach for the dendritic cell algorithm. In *Proceedings of the 11th international conference on adaptive and natural computing algorithms* (pp. 140–150).

Chelly, Z., & Elouedi, Z. (2015). Hybridization schemes of the fuzzy dendritic cell immune binary classifier based on different fuzzy clustering techniques. *New Generation Computing*, *33*(1), 1–31.

Chelly, Z., & Elouedi, Z. (2016). A survey of the dendritic cell algorithm. *Knowledge and Information Systems*, *48*(3), 505–535.

Chou, C., Kuo, B., Chang, F. (2006). The generalized condensed nearest neighbor rule as a data reduction method. In *Proceedings of the 18th international conference on pattern recognition* (pp. 556–559). IEEE.

Cortes, C., & Vapnik, V. (1995). Support-vector networks. *Machine Learning*, pp. 273–297.

Cui, Z., & Xiao, R. (2014). Bio-inspired computation: theory and applications. *Journal of Multiple-Valued Logic and Soft Computing*, *22*, 217–221.

Dasu, T., & Johnson, T. (2003). *Exploratory data mining and data cleaning* (Vol. 479). New York: Wiley.

Ester, M., Kriegel, H., Sander, J., Xu, X. (1996). A density-based algorithm for discovering clusters in large spatial databases with noise. In *Proceedings of 2nd international conference on knowledge discovery* (pp. 226–231).

Filzmoser, P., Garrett, R., Reimann, C. (2005). Multivariate outlier detection in exploration geochemistry. *Computers and Geosciences*, *31*, 579–587.

Gates, W. (1972). The reduced nearest neighbor rule. *IEEE Transactions on Information Theory*, *18*, 431–433.

Greensmith, J. (2007). The dendritic cell algorithm. PhD thesis, University of Nottingham.

Greensmith, J., & Aickelin, U. (2005). Introducing dendritic cells as a novel immune-inspired algorithm for anomaly detection. In *Proceedings of the 4th international conference on artificial immune systems* (pp. 153–167). Springer.

Greensmith, J., & Aickelin, U. (2006). Articulation and clarification of the dendritic cell algorithm. In *Proceedings of the 5th international conference on artificial immune systems* (pp. 404–417). Springer.

Greensmith, J., & Aickelin, U. (2007). The application of a dendritic cell algorithm to a robotic classifier. In *Proceedings of the 6th international conference on artificial immune systems* (pp. 204–215). Springer.

Greensmith, J., & Aickelin, U. (2008). The deterministic dendritic cell algorithm. In *Proceedings of the 7th international conference on artificial immune systems* (pp. 291–302). Springer.

Greensmith, J., Aickelin, U., Tedesco, G. (2010). Information fusion for anomaly detection with the dendritic cell algorithm. *Information Fusion*, *11*, 21–34.

Gu, F., Greensmith, J., Aickelin, U. (2010). Further exploration of the dendritic cell algorithm Antigen multiplier and time windows. In *Proceedings of the 5th international conference on artificial immune systems* (pp. 142–153). Springer.

Gustafson, D., & Kessel, W. (1979). Fuzzy clustering with a fuzzy covariance matrix. In *Proceedings of the IEEE conference on decision and control* (pp. 761–766). IEEE.

John, G., & Langley, P. (1995). Estimating continuous distributions in bayesian classifiers. In *Proceedings of the 11th conference on uncertainty in artificial intelligence* (pp. 33817345).

Kaur, S., Kumar, P., Singh, A. (2015). Nature inspired approaches for identification of optimized fuzzy model: a comparative study. *Journal of Multiple-Valued Logic and Soft Computing*, *25*, 555–587.

Kohavi, R. (1996). Scaling up the accuracy of naive-bayes classifiers: a decision-tree hybrid. In *Proceedings of the 2nd international conference on knowledge discovery and data mining*.

Lutz, M., & Schuler, G. (2002). Immature, semi-mature and fully mature dendritic cells: which signals induce tolerance or immunity? *Trends in Immunology*, *23*, 991–1045.

Markovich, S., & Scott, P. (1988). The role of forgetting in learning. In *Proceedings of the 5th international conference on machine learning* (pp. 459–465). Springer.

Markovitch, S., & Scott, P. (1993). Information filtering: selection mechanisms in learning systems. *Machine Learning*, *10*, 113–151.

Quinlan, J. (1993). C4.5: programs for machine learning. In *Morgan Kaufmann*.

Racine, K., & Yang, Q. (1996). On the consistency management of large case bases: the case for validation. In *Proceedings of the AAAI workshop on knowledge base validation* (pp. 84–90). AAAI.

Racine, K., & Yang, Q. (1997). Maintaining unstructured case bases. In *Proceedings of the 2nd international conference on case-based reasoning* (pp. 553–564). Springer.

Ruck, D.W., Rogers, S.K., Kabrisky, M., Oxley, M., Suter, B. (1990). The multilayer perceptron as an approximation to a bayes optimaldiscriminant function. In *IEEE transactions on neural networks* (pp. 29617298).

Smyth, B., & McKenna, E. (1998). Modelling the competence of case-bases. In *Advances in case-based reasoning: proceedings of the 4th European workshop on case-based reasoning* (pp. 208–220). Springer.

Smiti, A., & Elouedi, Z. (2010). COID: a maintaining case method based on clustering, outliers and internal detection. *Book chapter in Software Engineering, Artificial Intelligence, Networking and Parallel/Distributed Computing - Studies in Computational Intelligence, Springer*, *295*, 39–52.

Wilson, D. (1972). Asymptotic properties of nearest neighbor rules using edited data. *IEEE Transactions on Systems Man and Cybernetics*, *2*, 408–421.

Yarifard, A., & Yaghmaee, M. (2008). The monitoring system based on traffic classification. *World Applied Sciences Journal*, *5*, 150–160.

Zadeh, L. (1965). Fuzzy sets. *Information and Control*, *8*, 338–353.

Zhang, G. (2000). Neural networks for classification: a survey. *IEEE Transactions on Systems, Man, and Cybernetics*, *30*, 451–462.