DCA as Context (Environment) Sensitive System

Olubadeji Bukola
Department of computer science
Federal University of Technology, Akure

Adetunmbi. A.O
Department of computer science
Federal University of Technology, Akure

Alese B.K.
Department of computer science
Federal University of Technology, Akure

ABSTRACT
Existing immune-inspired techniques have not performed as well as expected when applied to the detection of intruders in computer systems. In nature, dendritic cells function as natural anomaly detection agents, instructing the immune system to respond if stress or damage is detected, it is also a crucial cell in the detection and combination of 'signals' which provide the immune system with a sense of context.

General Terms
Dendritic Cell Algorithm (DCA), Security

Keywords
Context, Artificial Immune System (AIS), Human Immune System

1. INTRODUCTION
The Dendritic Cell Algorithm (DCA) is an emerging algorithm within the field of artificial immune systems (AIS) [7]. It is a biologically inspired population based algorithm which is derived from behavioural models of natural dendritic cells (DCs) [8]. It is also underpinned by a recent paradigm in immunology termed the danger theory [1], which states that the human immune system is activated in response to the detection of 'danger signals'. As an algorithm, the DCA performs fusion of real valued input signal data and correlates this information with potentially anomalous 'antigen' data. The resulting correlation values are then classified to form an anomaly detection style of two-class classification. For the algorithm to function, input signal data is classified into one of three user- denied categories. The semantics which define the categories are based on the types of input used by natural DCs, which are currently termed PAMP signal, danger signal and safe signal.

An abstraction of the semantics of the natural signals is used to form a schema for the signal pre-categorization. This categorization is based on the following general principles:

• PAMPs: This is the Pathogenic associated molecular patterns are proteins expressed exclusively by bacteria, which can be detected by DCs and result in immune activation. The presence of PAMPS usually indicates an anomalous situation.

• Danger signals: These are the Signals produced as a result of unplanned necrotic cell death. On damage to a cell, the chaotic breakdown of internal components forms danger signals which accumulate in tissue. DCs are sensitive to changes in danger signal concentration. The presence of danger signals may or may not indicate an anomalous situation, however the probability of an anomaly is higher than under normal circumstances.

• Safe signals: These are the Signals produced via the process of normal cell death. Cells must die for regulatory reasons, and the tightly controlled process results in the release of various signals into the tissue. These `safe signals' result in immune suppression. The presence of safe signals almost certainly indicates that no anomalies are present.

Demonstration of properties of the DCA version 0.1

DCA Version 0.1, a prototype implementation using object-oriented python programming language.

This is to demonstrate the link between multiple signals processing over a population of artificial DCs and their ability to follow either the mature or semi-mature pathway. The pathway followed depends on the resultant value produced by combining input signals. If presented with a two-class data set, with items of each class ordered contiguously, proportion of cells directed down either pathway may change at the transition boundary between the two classes. The DCA is used to transform a representation of input data items, as antigen and signals, into MCAV values which can be assessed as an indicator of abnormality.

2. RELATED WORK
Researchers in computational fields have through a cross discipline of immunology and computer science developed algorithms by modelling computational abstracts from the immune system theories, processes and elements [3], and representing detection and recognition in geometrical shape space. These algorithms are constantly being innovated and have served as a reference point in applied AIS research to address computational issues in anomaly detections, computer security, optimisation and data mining, etc. These algorithms and models are as follows:

1. Negative Selection Algorithm (NSA) [4]; [15].
2. Artificial Immune Network Algorithm (AIN) [14].
3. Clonal Selection Algorithm (CLONALG) [3].
4. Danger Theory [5] and DCA [11].
5. Research work in artificial immune system architecture referred to as (ARTIS) in which monitoring of network services, traffic and user behaviour are observed to detect any deviation from normal behaviour patterns [12]; [13]. A further adaptation of ARTIS called LISYS examines the broadcasts source and destination of each TCP SYN packets to a detection node to check for anomalies. The latent time for detectors to confirm anomalies can be an issue here. Further work has been done employing LISYS with NSA in hybrid artificial immune system and Self Organising Map for network intrusion detection [19].

These first generations of AISs above were adaptive immunity inspired which was modelled on the principles of the classical immunological concept of discrimination between Self/Non-Self (SNS) but subsequent second generations of Artificial Immune System (AIS) are links between innate and adaptive

immunity. [2] was critical of the implementation of NSA's SNS, and concluded it was too simplistic to explain the whole complex human immune system representation to solve computational issues; and decided on an approach using immunological Danger Theory [16] which is considered appropriate to solve the computational complex abstraction from immune system. Following this model, AIS was implemented on an autonomous and distributed feedback and healing mechanism, triggered when a small amount of damage could be detected at an initial attacking stage. The system he named CFengine was DT inspired based on statistical methods of detecting anomalies. It is now established that the innate immune system also controls the adaptive immune system [18].

In the last decade new approaches to computer security anomalies detection has taken inspiration from Matzinger's danger theory [16] on an immunological concept which is a new notion to immunological understanding; a shift in paradigm from the widely held SNS paradigm on immunology. [1] published their novel paper, work on by Matzinger's danger theory, titled The Danger Theory and Its Application to Artificial Immune Systems called the DT. This paper drew reference from the human immune systems capability to respond to danger signals caused by necrotic cells (unnatural death of cells). There are many implementations of this DT by researchers in attempts to address issues relating to computer security but of which the DCA stands out in terms of functionality and results. The DCA [10] is a bio-inspired innate immunity computational algorithm modelled on both the innate and adaptive principles. The DT concept in intrusion detection is modelled like the Dendritic Cell (DC) of the neuron seeking out danger signals when there is a sudden increase in computer network traffic. Algorithms inspired by DT are the DCA [11] and Toll-like Receptor algorithm [17]. The DT was extended for computer network anomaly detection in [20]. [3] explored Botnet detection using DCA. DCA has had a high success rate in intrusion detection but not in responding to an attack.

3. PRE-PROCESSING PHASE

The DCA requires a data pre-processing phase in order to remove noise, redundancy in the dataset. The pre-processing phase of the DCA is of interest. The pre-processing phase of the DCA usually involves signal selection and categorisation, to generate the input signal stream of the algorithm. Signal selection is required to select the most interesting features from the original feature set. This is equivalent to the task of feature abstraction or selection in the area of machine learning, which is often accomplished by applying dimensionality reduction techniques.

The DCA has the ability of building a good classifier even on small training some dataset but work only on discretized data. Since, real life data is made of both or either continuous and discrete attributes valves then the need for discretization before training commence. Discretization can be defined as set of cuts over domains of attributes, representing an important pre-processing task for numeric data analysis.

The numerical (continuous) attributes in dataset are discretized based on Entropy, a supervised splitting technique exploring class distribution information in its calculation and determination of split-point. Entropy discretization technique leads to reduction of data size and makes use of class information, which may assist in improving classification accuracy. In discretizing a numerical attribute A, the value of A with the minimum entropy value is selected as split-point, and the resulting intervals are recursively partitions to arrive at a hierarchical discretization computer as follows [6].

Given D consisting of data tuples defined by a set attributes and a class label attribute

A split-point for A can partition the tuples in D into two subsets satisfying the conditions

A ≤ split point and A > split point respectively, thereby creating a binary discretization.

The expected information requirement for classifying a tuple in D based on partitioning by A is given by

$$Info_A(D) = \frac{|D_1|}{|D|} Entropy(D_1) + \frac{|D_2|}{|D|} Entropy(D_2) \quad 1.0$$

Where D_1 and D_2 correspond to the data tuples in D satisfying the conditions A ≤ split point and A > split point respectively. |D| is the number of tuples in D.

The entropy function for a given set is computed based on the class distribution of the tuples in the set. For example, given n classes, $C_1, C_2, ...C_n$, the entropy of D_1 is

$$Entropy(D_1) = \sum_{l=l}^{n} P_1 \log_2(p_1) \quad 2.0$$

Where P_1 is the probability of C_1 D_1, determined by dividing the number of tuples of C_1 in D_1 by $|D_1|$, the total number of tuples in D_1.

Hence, in selecting a split-point for attribute A, the chosen attribute is the one with attribute value that gives the minimum expected information required (i.e. min ($Info^A$ (D)). The process of determining a split-point is recursively applied to each partition obtained until the information requirement is less than a small threshold $\varepsilon(0)$.

4. DENDRITIC CELL ALGORITHM

The UCI Wisconsin Breast cancer data set is used to validate the DCA and is a well understood two-class data set. The UCI data consists of 700 items, classified by their corresponding real valued attributes. A further attribute is given showing membership of the data items to either class one or class two. The data ID is used to form antigen, with a pre-processed subset of attributes used to form the signals.

Table 1.0: Showing the UCI Dataset

DataID	CT	CS	CH	AD	EP	BN	CO	NN	MM	ClassID
1	10	8	8	2	3	4	8	7	8	1
2	3	3	5	2	3	10	7	1	1	1
3	10	5	7	3	3	7	3	3	8	1
4	10	4	6	1	2	10	5	3	1	1
5	8	10	10	10	8	10	10	7	3	1
6	6	1	3	1	4	5	5	10	1	1
7	8	8	8	1	2	10	6	10	1	2

DataID	CT	CS	CH	AD	EP	BN	CO	NN	MM	ClassID
8	5	5	7	8	6	10	7	4	1	2
9	8	4	10	5	4	4	7	10	1	2
10	8	3	4	9	3	10	3	3	1	2

Table 2.0: Showing the statistics of the UCI Wisconsin Breast Cancer Data set, where Clump Thickness = CT, Cell Size = CS, Cell Shape = CH, Adhesion = AD, Epithelial Cell Size = EP, Bare Nuclei = BN, Chromatin = CO, Normal

Nucleoli = NN, Mitoses = MM.

Statistic	CT	CS	CH	AD	EP	BN	CO	NN	MM
Mean	4.42	3.12	3.2	2.81	3.22	3.5	3.42	2.86	1.59
Median	4	1	1	1	2	1	3	1	1
Standard Dev.	2.82	3.04	2.97	2.86	2.21	3.63	2.41	3.06	1.71

The dimensionality of the data, n, is reduced from n=9 to n=6, this is achieved through examination of the nine attributes across all 700 data items, with the standard deviation of each attribute is calculated as shown in Table 2.0, using the attributes with the highest standard deviations to generate the signals.

Table 3.0: The selected attributes for calculating Danger and safe signals

AttributeText	AttributeValue	MedianValue	Column Index
BN	3.63	1	6
NN	3.06	1	8
CS	3.04	1	2
CH	2.97	1	3
AD	2.86	1	4
CT	2.82	4	1

The clump thickness attribute has the lowest standard deviation in this attribute subset and is used to derive the PAMP and safe signal, making it the "most certain" signal. The five other attributes of normal nucleoli, Adhesion, cell shape, bare nuclei and cell size are used to calculate the danger signal values. Each data item is mapped as an antigen, with the value of the antigen equal to the data ID of the item. Below are the methods used to derive the resultant signal values.

4.1 Signal selection method and data pre-processing

The general signal selection rules and some basic statistics are used for attribute selection and signal derivation. The general guidelines are presented in the list below:

PAMPs: The presence of PAMPs usually indicates an anomalous situation.

Danger signals: The presence of danger signals may or may not indicate an anomalous situation, however the probability of an anomaly is higher than under normal circumstances.

Safe signals: The presence of safe signals almost certainly indicates that no anomalies are present.

All attributes in this data set are not equal: some have greater information to be gleaned from than others. As stated in the general signal selection rules, both the PAMP and safe signal are positive indicators of an anomalous and normal signal. To achieve this with this machine learning data set, one attribute is used to form both PAMP and safe signal.

Using one attribute for these two signals requires a threshold level to be set: values greater than this can be classed as a safe signal, while values below this level would be used as a PAMP signal. In this experiment, the clump thickness attribute is used. Clump thickness has the lowest standard deviation out of the selected attribute set.

The low standard deviation of this attribute indicates its suitability for this signal. The exact procedure for calculating safe and PAMP signals is given below:

1. Selection a suitable attribute - clump thickness is chosen.

2. Calculate the median of all the selected attribute's values across both classes of data. In the case of clump thickness, the median value is 4.

3. For each attribute value determine if it is a PAMP or safe signal, as shown in [5]

Algorithm 1.0: Process for calculating PAMP and safe signals

if *value* > *median* then

value is a safe signal;

safe signal value = |mean - attribute value|;

PAMP signal value = 0;

else

value is a PAMP signal;

PAMP signal value = |mean - attribute value|;

safe signal value = 0;

end

The danger signal is also generated using the same process, the danger signal is 'less than certain to be anomalous'. This can be interpreted as a combination of several attributes, resulting in a value that may be used as anomalous. To obtain valves for danger signal, the mean value for each attribute set is required from the normal class alone (just class 1, not class 1 and class 2). Five attributes and their sets of values are used to derive the danger signal values. The process is as follows:

1. Mean values are calculated across the values of class 1 for each attribute chosen, not including class 2 as with the PAMP and safe signals. The five attributes selected for this thesis are:

Table 4.0: Statistics used to calculate danger signal

Statistic	CT	CS	CH	AD	EP	BN	CO	NN	MM
Mean	7.22	6.56	6.56	5.57	5.32	7.62	5.95	5.88	2.6
Median	8	6	6.5	5	5	10	7	6	1
Standard Dev.	2.41	2.71	2.56	3.21	2.44	3.13	2.24	3.37	2.56

• Cell Size, mean = 6.56;

• Cell Shape, mean = 6.56;

• Bare Nuclei, mean = 7.62;

• Normal Nucleoli, mean = 5.88;

Adhesion, mean = 5.57;

2. Take each attribute value in turn and calculate the absolute distance between the attribute values and the means shown above

   Attribute set – Means = absolute distance 3.0

Table 5.0: Shows the calculated absolute distance for danger signal

DataID	1	2	3	4	5
0	BN	NN	CS	CH	AD
1	3.62	1.12	1.44	1.44	3.57
2	2.38	4.88	3.56	1.56	3.57
3	0.62	2.88	1.56	0.44	2.57
4	2.38	2.88	2.56	0.56	4.57
5	2.38	1.12	3.44	3.44	4.43
6	2.62	4.12	5.56	3.56	4.57
7	2.38	4.12	1.44	1.44	4.57
8	2.38	1.88	1.56	0.44	2.43
9	3.62	4.12	2.56	3.44	0.57
10	2.38	2.88	3.56	2.56	3.43

3. The five calculated distance values are used in a further calculation to form the single value for the danger signal, DS. This value is the mean value

Table 6.0: The feature vectors

DataID	PAMP	Safe	Danger
1	0	6	2.24
2	1	0	3.19
3	0	6	1.61
4	0	6	2.59
5	0	4	2.96
6	0	2	4.09
7	0	4	2.79
8	0	1	1.74
9	0	4	2.86
10	0	4	2.96

The absolute distances calculated in Equation 3.0, with the derivation shown in Equation 4.0:

$$DS = \frac{\sum absolute\ \ distances}{number\ \ of\ \ attributes} \qquad 4.0$$

Following the signals generated which are a set of feature vectors shown in Table 4.6, with sample numbers added. If the value of PAMP is greater than zero, the value for the safe signal is set to zero.

One further calculation must be performed as part of pre-processing: the derivation of an anomaly threshold. This is for the analysis of the resultant MCAV coefficient produced per data ID. In this research work the distribution between the two classes is used to bias this value. The calculation displayed in Equation 4.7 demonstrates this process. In this equation, is the number of anomalous data items, tn is the total number of data items and at is the derived anomaly threshold. The sample values shown in the equation are correct for the

Wisconsin Breast Cancer data set.

$$at = \frac{an}{tn} \qquad 5.0$$

$$\frac{460}{700} = 0.657$$

Total Number of anomaly data items = 460

Total Number of all data items = 700

Anomaly Threshold = 0.657

The antigen used is the ID number of the data item. This item is not classified according to the label's value but by the associated signal values. On completion of the derivation of signals and conjugation with the associated antigen, the data is presented to the system.

The purpose is to validate an implementation of a DC based algorithm, to prove it is a feasible algorithm to construct and that it can perform some useful function.

Algorithm: Pseudocode of the processing performed by DCA Version 0.1[5].

input: antigen and signals feature vectors

output: antigen plus context values

create DC population of size 100;

initialise DCs;

for *each feature vectors* do

randomly select 10 DCs from the population;

for *the 10 selected DCs* do

get antigen;

store antigen;

get signals;

calculate interim output signals;

update cumulative output signals;

if *CSM output signal > migration threshold* then

DC removed from population;

DCs context is assigned;

all DCs collected antigen and context is output for analysis;

DC removed from population;

new DC added to population;

else

DC returned to population for further sampling;

end

end

end

collate the 10 context per antigen ID;

generate MCAV per antigen type;

For each incoming data Do

Calculate the number of mature DC and semi-mature DC;

If *nb semi-mature DC* > *nb* mature *DC* Then

Antigen = normal;

MCAV = 0

else

Antigen = abnormal;

MCAV = 1;

end

end

Three DCs are used termed DC1, DC2 and DC3 for the purpose of identification. Each DC is assigned an identical migration threshold value t_m, to a value of 10. The input signal values are artificially constructed so that each DC only collects one set of signals and antigen, with each DC exposed to a different set of signals.

Table 7.0: Shows the weight used for processing signals

W_{ijp}	j = 0	j = 1	j = 2
p = 0	2	1	2
p = 1	0	0	1
p = 2	2	1	-1.5

Selected DC for PAMP, Danger and Safe signals

Cycles = [DC0, DC1, DC2]

$PAMP\ S_0 \quad Danger\ signal\ S_1 \quad Safe\ signal\ S_2$

$Csm0_0 = \{s_{0,0} = 0\ ;\ s_{0,1} = 2.24\ ;\ s_{0,2} = 6;\}$

$Semi0_1 = \{\ s_{0,0} = 1;\ s_{0,1} = 3.19;\ \ s_{0,2} = 0;\}$

$Mat\ 0_2 = \{s_{0,0} = 0;\ s_{0,1} = 1.61;\ s_{0,2} = 6;\}$

The equation for calculating output signals is as follows:

$$O_p = (s_{0,0} * w_{0,0,p}) + (s_{0,1} * w_{0,1,p}) + (s_{0,2} * w_{0,2,p})\ \forall_p$$

The DC version 0.1 is performed in the following order;

1. The antigen vector is updated:

$A\{\ Ag1; Ag1; Ag1; Ag1; Ag1; Ag2; Ag2; Ag2; Ag2; Ag3; Ag3; Ag3;\}$

Cycle $l = 0$:

DC samples antigen, so DC1 $a(m) = \{Ag1; Ag2; Ag2; Ag2\}$

DC samples input signals so DC1 $s(m) = \{\ 0; 2.24; 6\}$,

DC calculates output signals so DC1 $o_p(m)$:

$$(csm)o_0 = (0 * 2) + (2.24 * 1) + (6 * 2) = 14.24$$

$$(\text{semi})o_1 = (0 * 0) + (2.24 * 0) + (\ 6 * 1) = 6.0$$

$$(\text{mature})o_2 = (0 * 2) + (2.24 * 1) + (6\ * -1.5) = -6.76$$

For DC1, t(m) = 10, therefore this DC has now exceeded its migration threshold as the value for o_0 is greater than t(m). Also, $o_2 < o_1$ and therefore DC1 is assigned a *cell context value of 0*, indicating that its collected antigen may be normal.

2. The antigen vector now consists of:

$A = \{\ Ag1; Ag1; Ag1; Ag1; Ag2; Ag3; Ag3; Ag3\}$

Cycle $l = 1$:

DC samples antigen, so DC2 $a(m) = \{Ag1; Ag1; Ag1; Ag1; Ag2\}$

DC samples input signals so DC2 $s(m) = \{\ 1; 3.19; 0\}$,

DC calculates output signals so DC2 $o_p(m)$:

$$(csm)o_0 = (1 * 2) + (3.19 * 1) + (0 * 2) = 5.19$$

$$(\text{semi})o_1 = (1 * 0) + (3.19 * 0) + (\ 0 * 1) = 0.0$$

$$(\text{mature})o_2 = (1 * 2) + (3.19 * 1) + (0\ * -1.5) = 5.19$$

For DC2, t(m) = 10, therefore this DC has not exceeded its migration threshold as the value for o_0 is not greater than t(m). Even though there area mixture of signals and the highest signal value comes from the danger signal value, $o_2 > o_1$ and therefore DC3 is assigned a *cell context value of 1*. This is due to the negative weight of the safe signal, which has a suppressive effect on the other two categories of signal.

3. The antigen vector now consists of:

$A = \{Ag3; Ag3; Ag3\}$

Cycle $l = 2$:

DC samples antigen, so DC3 $a(m) = \{Ag3; Ag3; Ag3\}$

DC samples input signals so DC3 $s(m) = \{\ 0; 1.61; 6\}$,

DC calculates output signals so DC3 $o_p(m)$:

$$(csm)o_0 = (0 * 2) + (1.61 * 1) + (6 * 2) = 13.61$$

$$(\text{semi})o_1 = (0 * 0) + (1.61 * 0) + (6 * 1) = 6.0$$

$$(\text{mature})o_2 = (0 * 2) + (1.61 * 1) + (6\ * -1.5) = -7.39$$

For DC3, t(m) = 10, therefore this DC has exceeded its migration threshold as the value for o_0 is greater than t(m). $o_2 < o_1$ and therefore DC3 is assigned a *cell context value of 0* indicating that its collected antigen is likely to be normal.

Mean Context Antigen Generation

$$MCAV = \frac{No.of\ mature\ \ presentation}{No.of\ presentation} \qquad 5.0$$

Table 8.0: Shows the result of Calculated MCAV

Antigen type	No. of presentation	No. of mature presentation	MCAV
Ag1	5	2	0.4
Ag2	4	2	0.5
Ag3	3	1	0.33

To perform anomaly detection, a threshold must be applied to the generated MCAVs through the calculated anomaly threshold which is 0.5. Therefore, Ag 2 is classed as anomalous, since the threshold will either be grater or equal while Ag1 and Ag3 classified as normal because they are lower than 0.5.

4.2 Experiment

Three simple experiments are used to demonstrate the capability of the DCA to differentiate between two distinct contexts using the UCI and NSL KDD data set to justify the DCA capability. To achieve this, three different data orders are used. Experiment one uses a one step data order. Here, the data is ordered continuously i.e. all class one items are processed followed by all class two items. In experiment two, the data is partitioned into three sections, resulting in a two-step data order. The data comprising class one is split into two sections and the class two data is embedded between the classes one partitions. This partitioning is represented in

Figure 1.0. Experiment three consists of data randomised between class one and class two.

For each experiment the results presented are based on mean MCAV values generated across 20 runs. Antigen is sampled ten times and there are a total of 700 antigens in the data set. This gives 7,000 antigen presentations per run, yielding 140,000 presentations in total. The classification threshold is set to a MCAV of 0.66 as previously stated. Items below the threshold are classified as class one and above as class two. The resulting classified antigens are compared to the labels given in the original data set. The rate of errors is

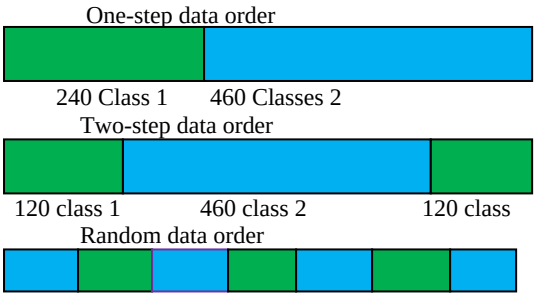

Fig.1.0: The Data Order

Experiment on Anomaly Detection

The performance measures are calculated from:

- True Positives (TP), the number of malicious executables correctly classified as malicious
- True Negatives (TN), the number of benign program correctly classified as benign
- False Positive (FP), the number of benign program falsely classified as malicious
- True Negatives (FN), the number of malicious program falsely classified as benign
- The measures are based on the formulae

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

$$Detection\ rate = \frac{TP}{TP+FN}$$

Table 9.0: Classification obtained from data orders

Experi	TP	TN	FP	FN	Class 1	Class 2
1 step	460	236	0	4	240	460
2 step	453	239	6	1	240	460
Random	285	177	63	174	240	460

For the one-step experiment, the rate of correct classifications is exemplary, yielding four errors out of a total of 700 WBC data items. This yields a classification rate of 99%.

The measures are based on the formulae for 1 step

$$Accuracy = \frac{460 + 236}{460 + 236 + 4 + 0} = \frac{696}{700} = 0.99\%$$

$$Detection\ rate = \frac{460}{460 + 0} = \frac{460}{460} = 1\%$$

$$False\ alarm = \frac{4}{4 + 236} = \frac{4}{240} = 0.2\%$$

The measures are based on the formulae for 2 steps

$$Accuracy = \frac{453 + 239}{453 + 239 + 6 + 1} = \frac{692}{700} = 0.98\%$$

$$Detection\ rate = \frac{453}{453 + 1} = \frac{453}{459} = 0.99\%$$

$$False\ alarm = \frac{6}{6 + 239} = \frac{6}{245} = 0.02\%$$

For the two-step data seven errors out of 700 are recorded, which despite being slightly higher is still a low rate of error. This yields a classification rate of 98%.

The measures are based on the formulae for random order

$$Accuracy = \frac{286 + 177}{286 + 177 + 63 + 174} = \frac{463}{700} = 0.66\%$$

$$Detection\ rate = \frac{286}{286 + 174} = \frac{286}{460} = 0.62\%$$

$$False\ alarm = \frac{63}{63 + 177} = \frac{63}{237} = 0.27\%$$

For the random data two hundred and thirty seven errors out of 700 are recorded, which extremely high, with a classification rate of 66%.

$$False\ alarm = \frac{FP}{FP + TN}$$

4.3 Nsl kdddata

Before pre-processing

0,tcp,private,REJ,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,121,19,0.00,0.00,1.00,1.00,0.16,0.06,0.00,255,19,0.07,0.07,0.00,0.00,0.00,0.00,1.00,1.00,normal,21

0,tcp,private,S0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,166,9,1.00,1.00,0.00,0.00,0.05,0.06,0.00,255,9,0.04,0.05,0.00,0.00,1.00,1.00,0.00,0.00,neptune,21

Generated cut_point during pre-processing

cut_point0[60]={0.5,21,26,113.5,127,148.5,210.5,213,215,225.5,241,260,261.5,11305,12437,13517,13553,14873,15068,15214,15361,15397,15508,15705,18062,21010,21320,21786,23586,23910,24014,24024,25018,25044,25084,25220,25390,25672,25678,28006,28024,29964,112624,112824,116344,118664,120944,128912,176368,210080,210096,210160,211920,212528,215168,216816,217232,218688,220864,232848,};

The pre-processed NSL-KDD Dataset

1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1.

1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1

1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,2.

1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,2.

The NSL KDD intrusion detection dataset is also used to compare and verify the effectiveness and efficiency of the dendritic cell algorithm using 700 data out of the whole dataset for the validation.

Table 10.0: Classification obtained from comparing three different data orders

Experi	TP	TN	FP	FN	Class 1	Class 2
1 step	458	240	0	4	240	460
2 step	456	238	2	4	240	460
Random	400	110	130	60	240	460

$$Accuracy = \frac{458 + 240}{458 + 240 + 2 + 0} = \frac{698}{700} = 0.99\%$$

$$Detection\ rate = \frac{458}{458 + 2} = \frac{458}{460} = 0.99\%$$

$$False\ alarm = \frac{0}{0 + 240} = \frac{0}{240} = 0\%$$

For the one-step experiment, the rate of correct classifications is exemplary, yielding four errors out of a total of 700 NSL KDD data items. This yields a classification rate of 99%.

The measures are based on the formulae for 2 step NSL KDD data order

$$Accuracy = \frac{456 + 238}{456 + 238 + 2 + 4} = \frac{694}{700} = 0.99\%$$

$$Detection\ rate = \frac{456}{456 + 4} = \frac{456}{460} = 0.99\%$$

$$False\ alarm = \frac{2}{2 + 238} = \frac{2}{240} = 0.00\%$$

For the two-step data six errors out of 700 are recorded, which despite being slightly higher is still a low rate of error. This yields a classification rate of 99%.

The measures are based on the formulae for random order NSL KDD data order

$$Accuracy = \frac{350 + 180}{350 + 180 + 60 + 110} = \frac{530}{700} = 0.76\%$$

$$Detection\ rate = \frac{350}{350 + 110} = \frac{350}{460} = 0.76\%$$

$$False\ alarm = \frac{130}{130 + 110} = \frac{130}{240} = 0.54\%$$

for the random data one hundred and ninety errors out of 700 are recorded, which extremely high, with a classification rate of 76%.

Result discussion: The algorithm is evaluated by applying it to two universal classification dataset and assessing its performance according to three evaluation metrics: detection rate, false detection rate, and accuracy. The results show that the dendritic cell algorithm performs best on ordered data than unordered one, which means that the DCA is context (environment) sensitive.

5. CONCLUSION

The DCA was applied to various detection problems. The algorithm was validated using a standard machine learning data set. In this experiment context switching between two classes of data was detected by the DCA. An evaluation of the algorithm showed success when applied to the detection of intrusions on the network. Comparisons of the performances of the DCA using kdd'99 intrusion detection and breast cancer evaluation dataset were drawn and implemented using python programming language.

6. RECOMMENDATION

While the DCA has performed well on the problems presented, to fully assess the effectiveness of the DCA, it must be thoroughly benchmarked against the criticism from [1] in that the DCA has a propensity to have a high false positive rate on unordered data because not all data generated on the network will follow the same regular pattern at all time, how to benchmark the DCA will be discuss in details in the next publication.

7. REFERENCES

[1] Aickelin and Cayzer 2002. The danger theory and its application to artificial immune systems. In Proc.of the1st International Conference on ArtificialImmune Systems (ICARIS). University of Kent at Canterbury Printing Unit.

[2] Burgess, M. 1998. Computer immunology. In Proc. of the Systems Administration Conference (LISA-98).

[3] de Castro and Von Zuben 2002. Learning and Optimization Using the Clonal Selection Principle, *IEEE* Transactions on Evolutionary Computation, Special Issue on Artificial Immune Systems.

[4] Forrest, .S., Perelson, .A., Allen, .L. and Cherukuri. 1994. Self-nonself discrimination in a computer. In Proc. of the IEEE Symposium on Security and Privacy, 1994, 202–209. IEEE Computer Society.

[5] Greensmith .J. 2007. The Dendritic CellAlgorithm. Ph.D thesis, School of Computer Science, University of Nottingham.

[6] Jiawei and Micheline, K. 2006. Data Mining:Concepts and techniques, second edition, Elsevier inc.Spector,

[7] Leandro N. de Castro and Timmis, J. 2002. Artificial Immune Systems: A New Computational Intelligent Approach. Springer-Verlag.

[8] Lutz, .M .B. and Schuler, G. 2002. Immature, semi-mature and fully mature dendritic cells: which signals induce tolerance or immunity? TRENDS in Immunology, 23(9): pages 445-449.

[9] Matzinger. P. 2002. The Danger Model: A Renewed Sense of Self. Science, 296(5566): pages 301{305.

[10] Hofmeyr, .S and Forrest, .S. 1999. Immunity by design. In Proc. of the Genetic and evolutionary computation Conference (GECCO), pages 1289–1296.

[11] Hofmeyr, S. A, Somayaji, .A and Forrest, .S. 1999, Intrusion detection using sequences of system calls, J. Computer. Security. Vol.6, pp. 151–180.

[12] Jerne. N. K. 1974. Towards a network theory of the immune system. Ann. Immunol. (Inst. Pasteur), 125C, 373-389.

[13] Ji, Z. and Dasgupta, D. 2007. Revisiting negative selection algorithms, Evolutionary Computation, vol.15, no. 2, pages .223-251.

[14] Matzinger, .P. 1998. An innate sense of danger. Semin. Immunol.10, pages399–415

[15] Powers, .S and He, .J. 2008. A hybrid artificial immune system and Self Organising Map for network intrusion

detection, Information Sciences, 2008, vol.178, no. 15, pp.3024-3042

[16] Schenten, .D, and Medzhitov, R. 2011. "The control of adaptive immune responses by the innate immune system", Adv Immunol. vol.109, pages 87-124.

[17] Stibor, .T, Mohr, .P, Timmis, J and Eckert, .C. 2005. Is negative selection appropriate for anomaly detection? In Proc. of Genetic and Evolutionary Computation Conference (GECCO), pages 321–328.

[18] Timmis, J. 2007. Artificial immune systems: today and tomorrow. Natural Computing, 2007, 6(1): pages1–18.

[19] Twycross, J. (2007): "Integrated innate and adaptive artificial immune systems applied to process anomaly detection", PhD thesis, School of Computer Science, The University of Nottingham.2007.

[20] Yeom, K. W. 2007. Immune-inspired algorithm for anomaly detection, Stud. Comput. Intell. (SCI). 57, pages 129–154.