Turkish Journal of Electrical Engineering & Computer Sciences

http://journals.tubitak.gov.tr/elektrik/

Research Article

Turk J Elec Eng & Comp Sci
(2020) 28: 3270 – 3284

doi:10.3906/elk-2002-75

# An intelligent diagnostic method based on optimizing B-cell pool clonal selection classification algorithm

**Chao LAN, Hongli ZHANG*, Xin SUN, Zhongyuan REN**
Department of Mechatronics Engineering and Automation, Faculty of Engineering, Shanghai University, Shanghai, P.R. China

**Received:** 13.02.2020 • **Accepted/Published Online:** 28.07.2020 • **Final Version:** 30.11.2020

**Abstract:** The trend of intellectualization and complication of mechanical equipment makes the demand for intelligent diagnostic methods more and more intense in industry. In view of the difficulty of obtaining mechanical fault samples and the requirement of clear and reliable diagnosis results, intelligent diagnosis methods need to adapt to the learning of small samples and have the interpretability of white box model. In this paper, inspired by biological immunity, an intelligent fault diagnosis method was proposed——optimizing b-cell pool clonal selection classification algorithm (OBPCSCA). The OBPCSCA provides a method to construct unique B-cell pools corresponding to specific antigen pools, and uses greedy strategy to generate memory B-cell pools. The experimental comparison with AIRS and AICSL on four UCI benchmark data sets shows that the OBPCSCA has a better balance between the number of memory cells and the accuracy of classification. In particular, compared with AIRS, the OBPCSCA can greatly reduce the number of memory B-cells on the premise of ensuring high classification accuracy. In comparison with the top general classifiers, the OBPCSCA has certain competitiveness in these four data sets. Finally, the algorithm was applied to the bearing data set of Case Western Reserve University for fault diagnosis, and the results showed effectiveness of the algorithm.

**Key words:** Intelligent diagnosis, optimizing B-cell pool, clonal selection, immune system

## 1. Introduction

Research on mechanical fault diagnosis has a long history, which can be traced back to 1960s in the United States [1]. In recent years, with the development of science and technology, mechanical equipment tends to be complex and intelligent, and the demand for intelligent fault diagnosis technology in the field of mechanical fault diagnosis is increasingly strong [2, 3]. The application of intelligent methods such as expert system [4], artificial neural network (ANN) [5, 6] and support vector machine (SVM) [7, 8] in the field of mechanical fault diagnosis is the best proof. These methods play a positive role in many important fields, such as signal processing [9], dynamics analysis [10], and reliability analysis [11, 12]. However, these intelligent methods are not completely compatible with the field of fault diagnosis. ANN needs a lot of samples to train, but it is very difficult to obtain fault samples in reality [4, 5]. Although SVM does not require as many training samples as ANN, the selection of its kernel function and its parameters both depend on experience. As for expert system, the difficulty of knowledge acquisition and the poor updating ability of knowledge base are its fatal defects [4].

Biological immunity is a natural system that protects a host organism against disease-causing elements threatening its normal functioning [13, 14]. It offers many interesting features that inspired the design of artificial

*Correspondence: zhang40941@126.com

immune systems (AIS) to solve several kinds of engineering problems [13], including abnormal detection and fault diagnosis. One of the earliest engineering applications of the AIS was the negative selection algorithms in 1994, proposed by Forrest [15, 16]. In the following decades, a large number of papers had been published on the improvement of negative selection algorithm, among which some influential ones were: the real-value negative selection algorithm proposed by Gonzalez [17]; variable radius detector (v-detector) proposed by Zhou and Dasgupta [18]. Stibor et al. proposed their own detector classification algorithm (positive selection algorithm) [19, 20]. In addition, many clustering algorithms had been inspired by artificial immunity. For example, Timmis proposed the artificial immune system with limited resources [21] and De Casto and Von Zuben [22] proposed artificial immune network model. Inspired by the biological immune system, these algorithms proposed some artificial immune system concepts such as artificial recognition ball, affinity degree, affinity threshold, and memory cell pool, and inspired immune classification algorithms such as artificial immune recognition system (AIRS) [23], cloning selection classification algorithm [24], and artificial immune classifier with swarm learning (AICSL)[25].

The algorithm proposed in this paper also belongs to classification algorithm. Compared with the existing immune classification algorithms, there are two outstanding innovations:

1) A method of constructing B-cell pool was designed. B-cells in immune algorithm are usually hyperspheres with the same radius. Our scheme is to construct hyperspheres with scale-adaptive radii. This B-cells with scale-adaptive radii can better express the distribution characteristics of data in the feature space;

2) A method of optimizing B-cell pool was designed. The application of traditional cloning selection is to clone each antigen to obtain B-cell population, and then delete the redundant B-cells. There are many disadvantages in this scheme, such as the large number of cloned B-cells and the elimination of high-quality B-cells when generating memory cells. Therefore, we abandoned the scheme and used greedy strategy to generate memory B-cells one by one. In fact, it is an incremental learning model.

The remaining sections of the paper are structured as follows. In Section 2, the optimizing B-cell pool clonal selection classification algorithm (OBPCSCA) will be introduced in detail. The experiments on four UCI benchmark data sets and application on the bearing data set of Case Western Reserve University[1] for fault diagnosis will be presented in Section 3. In Section 4, conclusion and future work are provided.

## 2. Optimizing B-cell pool clonal selection classification algorithm (OBPCSCA)

The principal of clonal selection is one of the most elegant in all of immunology, which uses a small number of B-cells in one class (there will only be about thirty B-cells in the blood that can produce an antibody which will bind to a given antigen) to defend against a large number of antigen invasions [26]. Inspired by this immune mechanism, the OBPCSCA was designed to minimize the number of B-cells while ensuring high classification accuracy. The whole OBPCSCA was comprised of optimizing B-cell pool clonal selection algorithm (OBPCSA) and classifier modules, which corresponded to the training and testing phases of the algorithm. The flow of the whole algorithm was shown in Figure 1. In Figure 1, the whole algorithm is visualized, and the training data contains three different labels, namely, “Orange”, “Green” and “Red”. In the training phase of the algorithm,

[1]CWRU (2018). Bearing Data Center [online]. Website https://csegroups.case.edu/bearingdatacenter/pages/download-data-file [accessed 10 August 2020].

the memory B-cell pool (MBP) is obtained through training, which will be used as the basis for the classification of the testing phase.

The OBPCSA module contains two main points:

1) A method was designed to construct B-cell pool, which is described in Section 2.1;

2) A method was designed to optimize B-cells from B-cell pool to form Memory B-cell Pool (MBP), which is described in Section 2.2.

The design of classifier was similar to AIRS [27], which adopted the idea of the $k$-nearest neighbor algorithm and is described in Section 2.3.

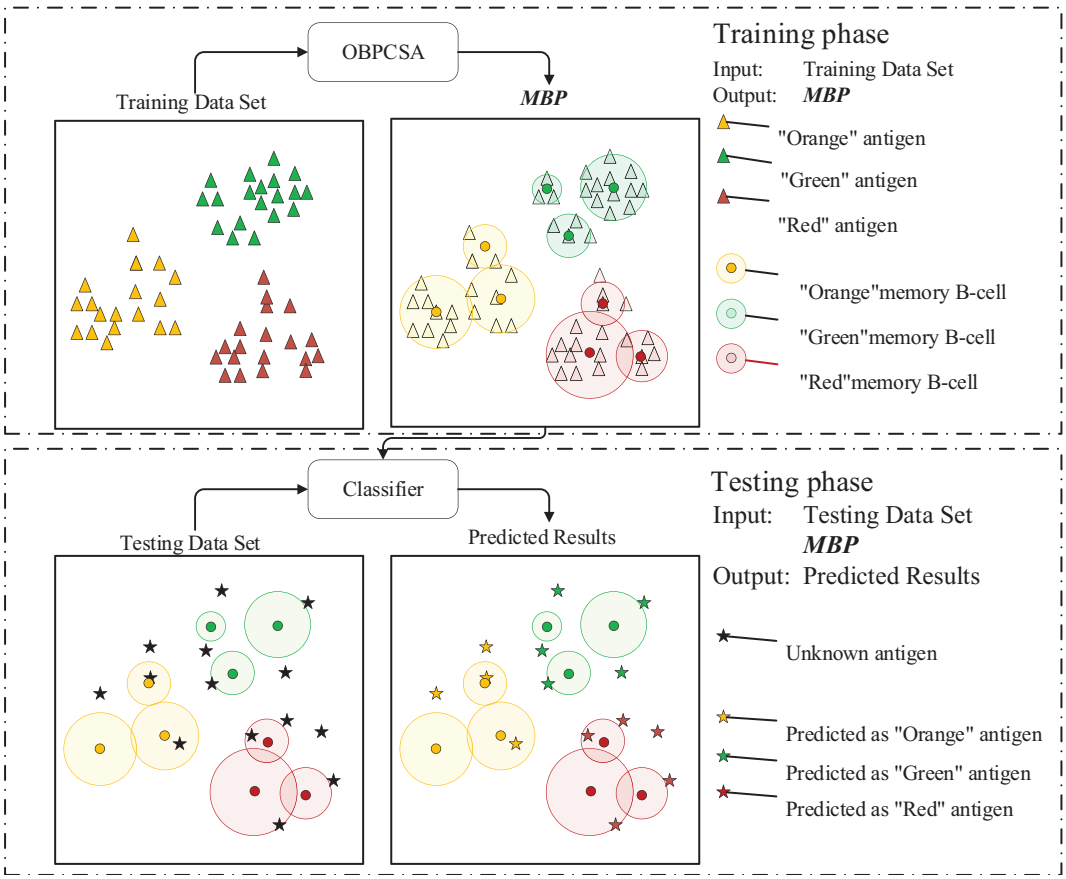


**Figure 1**. The whole flow of the optimizing B-cell pool clonal selection classification algorithm. The algorithm consists of OBPCSA module and classifier module. In training phase, the OBPCSA module obtains memory B-cell pool $\boldsymbol{MBP}$ through training training data set; in testing phase, the classifier module classifies testing data one by one by combining $\boldsymbol{MBP}$ obtained in training phase.

### 2.1. Constructing B-cell pool

The classifier of SVM is designed as a hyperplane of state space, and the core of SVM algorithm is to find an optimal hyperplane in the state space [28]. Similarly, the OBPCSCA uses hyperspheres to divide state space, and the core of the algorithm is to construct and optimize hyperspheres in the state space.

For the sake of description, hyperspheres in state space will be called B-cells in the OBPCSCA. As shown in Figure 2, any B-cell $\boldsymbol{B_i}$ can be described by $(\boldsymbol{x_i}, r_i)$, where spherical center is the $\boldsymbol{x_i}$, named antibody and radius is $r_i$. In this paper, the word "pool" expresses the concept of set. The B-cell pool constructed contains the following characteristics:

1) B-cell pool corresponds to antigen pool one by one. The B-cell pool constructed by this algorithm is specific, just like the B-cell pool of human immunity: a B-cell secretes only antibodies against specific antigens;

2) The radii of B-cells in the B-cell pool constructed by this algorithm are scale-adaptive. Each B-cell contains the information of cell's center and radius, using $\boldsymbol{B_i}$ to represent the $i$th B-cell, that is, $\boldsymbol{B_i} = (\boldsymbol{x_i}, \delta_i)$, where $\delta_i$ contains the radius information of the $i$th B-cell;

3) In theory, the number of B-cells in B-cell pool is infinity.

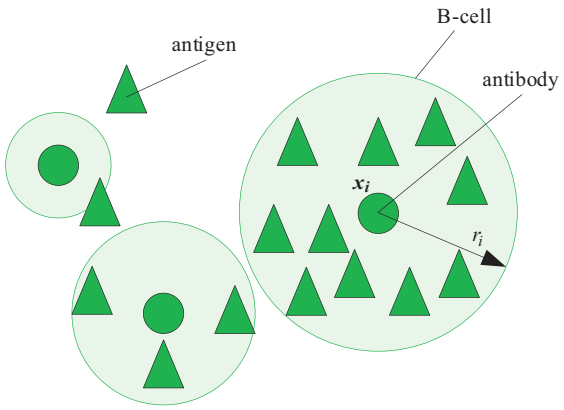


**Figure 2**. Some immune concepts in the optimizing B-cell pool clonal selection classification algorithm. Antigen and antibody are both points in the same state space, and B-cell is a hypersphere with antibody as its center. The radius of B-cell is related to its location in the state space: the larger the antigen density, the larger the radius of B-cell.

To this end, we have made the following two definitions:

**Definition 1. The antigenic closeness centrality** Given an antigen pool $\boldsymbol{Agp_k} = \{\boldsymbol{y_1}, \boldsymbol{y_2}, ..., \boldsymbol{y_N}\}$ with $N$ antigens in an m-dimensional state space, where the subscript $k$ is the label of all antigens in the antigen pool, the antigenic closeness centrality $\rho(\boldsymbol{x})$ at any point $\boldsymbol{x}$ of the state space is defined as Eq. (1):

$$\rho(\boldsymbol{x}) = e^{-\frac{d_{av}(\boldsymbol{x})}{\theta_k}}, \tag{1}$$

where

$$d_{av}(\boldsymbol{x}) = \frac{1}{N}\sum_{i=1}^{N} \|\boldsymbol{x} - \boldsymbol{y_i}\|. \tag{2}$$

The parameter $\theta_k$ is a constant associated with antigen pool $\boldsymbol{Agp_k}$.

**Definition 2: Affinity function** Given an antigen pool $\boldsymbol{Agp_k} = \{\boldsymbol{y_1}, \boldsymbol{y_2}, ..., \boldsymbol{y_N}\}$ with $N$ antigens in an m-dimensional state space, where the subscript $k$ is the label of all antigens in the antigen pool, the algorithm

generates randomly B-cells with its antibodies at any point $\boldsymbol{x_j}$ and the affinity between the B-cell $\boldsymbol{B_j}(\boldsymbol{B_j} = (\boldsymbol{x_j}, \delta_j))$ and any antigen $\boldsymbol{y_i}(\boldsymbol{y_i} \in \boldsymbol{Agp_k})$ is defined as Eq. (3):

$$f_A(\boldsymbol{B_j}, \boldsymbol{y_i}) = e^{-\left(\frac{\left\|\boldsymbol{y_i}-\boldsymbol{x_j}\right\|}{\delta_j}\right)^2}, \tag{3}$$

where

$$\delta_j = \rho(\boldsymbol{x_j}). \tag{4}$$

Therefore, if the affinity threshold of antigen-antibody matching in antigen pool $\boldsymbol{Agp_k}$ is set to $Ta_k$, the B-cell Pool $\boldsymbol{BP_k}$ can be described as follows: $\forall\ \boldsymbol{B_j} = (\boldsymbol{x_j}, \delta_j)\ \in \boldsymbol{BP_k}$, $\exists\ \boldsymbol{y_i} \in \boldsymbol{Agp_k}\ \boldsymbol{s.t.}\ f_A(\boldsymbol{B_j}, \boldsymbol{y_i}) \geq Ta_k$.

According to the critical condition Affinity($\boldsymbol{B_j}, \boldsymbol{y_i}$)=$Ta_k$, the hyperspherical radius of the $\boldsymbol{B_j}$ is deduced as Eq. (5):

$$r_j = \delta_j \sqrt{-\ln(Ta_k)}. \tag{5}$$

It is noteworthy that if the affinity between an antigen $\boldsymbol{y_i}$ and a B-cell $\boldsymbol{B_j}$ exceeds the affinity threshold $Ta_k$, in the state space, $\boldsymbol{y_i}$ is inside the hypersphere corresponding to $\boldsymbol{B_j}$. In addition, under the condition that $Ta_k$ is determined, the $r_j$ of the B-cell $\boldsymbol{B_j}$ is affected by $\delta_j$, that is, the larger $\delta_j$ is, the larger B-cell is.

The B-cell pool $\boldsymbol{BP_k}$ is the specific B-cell pool corresponding to the antigen pool $\boldsymbol{Agp_k}$, if the constants $\theta_k$ and $Ta_k$ associated with the antigen pool $\boldsymbol{Agp_k}$ are known. The principle of $\theta_k$ optimization was as follows: the optimal $\dot{\theta_k}$ maximizes the value[max($\rho(\boldsymbol{y})$)-min($\rho(\boldsymbol{y})$)] , where $\boldsymbol{y} \in \boldsymbol{Agp_k}$. The reason is to make the difference between the $\delta$ of B-cells in central and that of edge B-cells obvious.

According to Eq. (1) and Eq. (2), this optimization problem can be described as Eq. (6):

$$\begin{cases} \boldsymbol{Max}\ e^{-\frac{d_{min}(\boldsymbol{x})}{\theta_k}} - e^{-\frac{d_{max}(\boldsymbol{x})}{\theta_k}}; \\ \boldsymbol{s.t.}\ d_{av}(\boldsymbol{x}) \in [d_{min}, d_{max}],\ \boldsymbol{y} \in \boldsymbol{Agp_k}. \end{cases} \tag{6}$$

From Eq. (6), we can have Eq. (7):

$$\dot{\theta_k} = \frac{d_{max} - d_{min}}{\ln\left(\frac{d_{max}}{d_{min}}\right)} \tag{7}$$

Therefore, the optimal $\dot{\theta_k}$ is determined according to Eq. (7). (Note: $d_{min} = \boldsymbol{Min}[d_{av}(\boldsymbol{y}),\ \boldsymbol{y} \in \boldsymbol{Agp_k}]$, and $d_{max} = \boldsymbol{Max}[d_{av}(\boldsymbol{y}),\ \boldsymbol{y} \in \boldsymbol{Agp_k}]$.)

The parameter $Ta_k$ reflects the inherent property of antigen pool $\boldsymbol{Agp_k}$ and its real value cannot be obtained because of incomplete antigens in $\boldsymbol{Agp_k}$ in theory. The solution in the OBPCSCA is to accept the idea of Watkins [27]: the affinity threshold is the average affinity value of the eigenvector of all training data items. The affinity threshold is calculated as described in Eq. (8):

$$Ta_k = \frac{\sum_{i=1}^{N}\sum_{j=i+1}^{N} f_A(\boldsymbol{y_i}, \boldsymbol{y_j})}{\frac{N(N-1)}{2}},\ y_i, y_j \in \boldsymbol{Agp_k} \tag{8}$$

where

$$f_A(\boldsymbol{y_i}, \boldsymbol{y_j}) = e^{-\left(\frac{\left\|\boldsymbol{y_i}-\boldsymbol{y_j}\right\|}{\delta_i}\right)^2}$$

The number of B-cells in B-cell Pool $\boldsymbol{BP_k}$ corresponding to $\boldsymbol{Agp_k}$ is infinity in theory, and mapping between antigen pool $\boldsymbol{Agp_k}$ and B-cell pool $\boldsymbol{BP_k}$ is described in Figure 3.

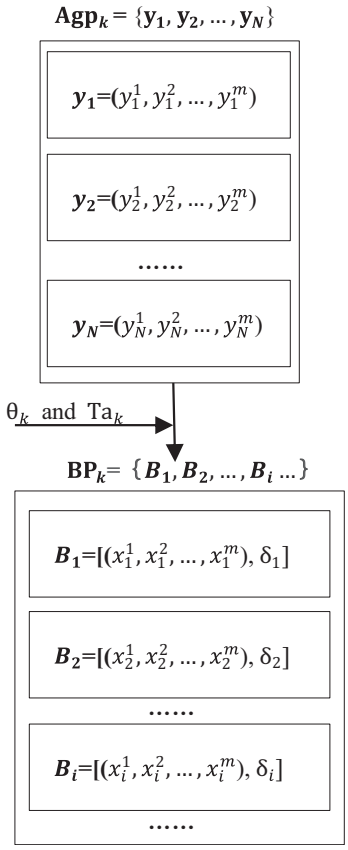


**Figure 3**. Mapping between antigen pool $\boldsymbol{Agp_k}$ and B-cell pool $\boldsymbol{BP_k}$. The two parameters $\theta_k$ and $Ta_k$ are found by training $\boldsymbol{Agp_k}$.

### 2.2. Optimizing B-cell pool with clonal selection

Biological immune system is able to remember every source of infection (antigen) and when the same infection occurs again, the immune system reacts more quickly and processes it more efficiently [29]. What supports the second response of the immune system is the immune memory mechanism. Inspired by this immune mechanism, the OBPCSCA uses the OBPCSA to achieve immune memory.

Taking the antigen pool $\boldsymbol{Agp_k} = \{\boldsymbol{y_1}, \boldsymbol{y_2}, \ldots, \boldsymbol{y_N}\}$ for example, the purpose of the OBPCSA is to obtain the memory B-cell pool $\boldsymbol{MBP_k} = \{\boldsymbol{M_1}, \boldsymbol{M_2}, \ldots, \boldsymbol{M_n}\}$, where $n$ is the number of memory B-cells in $\boldsymbol{MBP_k}$. This optimization problem can be described by Eq. (9):

$$\begin{cases} \boldsymbol{Min}\ n; \\ \boldsymbol{s.t.}\ \boldsymbol{MBP_k} \subset \boldsymbol{BP_k}. \end{cases} \tag{9}$$

According to Eq. (9), the optimization is to find a minimum set of hyperspheres to satisfy that any antigen of $\boldsymbol{Agp_k}$ is inside at least one hypersphere in the state space. Greedy strategy was used to optimize memory B-cells. Through recursion, memory B-cells were generated one by one to form a memory B-cell pool.

To this end, we introduced the clonal selection mechanism and modified the traditional clonal selection operation appropriately. The clonal selection of the OBPCSA is divided into three parts:

1) Cloning proliferation of B-cells, whose purpose is to produce a cloned B-cell pool to be optimized;

2) Optimal selection of memory B-cell, whose purpose is to select a memory B-cell from the cloned B-cell pool obtained;

3) Recursion, whose purpose is to form a memory B-cell pool.

### 2.2.1. Cloning proliferation of B-cells

Cloning proliferation in OBPCSA involves two steps: antigen presentation and B-cells cloning mutation proliferation.

***Step 1 : Antigen presentation.*** Just as immune helper cells such as macrophages in the biological immune system process antigens, the purpose of antigen presentation is to expose the characteristics of antigens. Antigen presentation stage in the OBPCSA will select an antigen $\boldsymbol{y_*}$ from the antigen pool $\boldsymbol{Agp_k} = \{\boldsymbol{y_1}, \boldsymbol{y_2}, ..., \boldsymbol{y_N}\}$ based on $\rho(\boldsymbol{y}_*) = \boldsymbol{Max}\ [\rho(\boldsymbol{y}), \boldsymbol{y} \in \boldsymbol{Agp_k}]$.

***Step 2 : $\boldsymbol{B-cells\ cloning\ mutation\ proliferation.}$*** Each B-cell has two parts: antibody (AT), which is the center of the B-cell, and delta ($\delta$), which expresses the radius of the B-cell. For convenience of expression, the cloned B-cell pool ($\boldsymbol{BP_{clone}}$) was described in Eq. (10) to Eq. (12):

$$\boldsymbol{BP_{clone}} = (\boldsymbol{AT_{clone}}, \boldsymbol{\delta_{clone}}). \tag{10}$$

where

$$\begin{aligned} \boldsymbol{AT_{clone}} &= (\boldsymbol{x_1}, \boldsymbol{x_2}, , \boldsymbol{x_{N_c}})^T \\ &= \begin{bmatrix} x_1^1 & x_1^2 & \cdots & x_1^d \\ x_2^1 & x_2^2 & \cdots & x_2^d \\ \vdots & \vdots & \ddots & \vdots \\ x_{N_c}^1 & x_{N_c}^2 & \cdots & x_{N_c}^d \end{bmatrix}, \end{aligned} \tag{11}$$

$$\boldsymbol{\delta_{clone}} = (\boldsymbol{\rho(x_1)}, \boldsymbol{\rho(x_2)}, , \boldsymbol{\rho(x_{N_c})})^T. \tag{12}$$

B-cells cloning mutation proliferation produces $N_c$ cloned B-cells named $\boldsymbol{cB}$ to form a clone B-cell pool $\boldsymbol{BP_{clone}}$ , which obeys Eq. (13) and Eq. (14):

$$\boldsymbol{AT_{clone}} = \begin{bmatrix} y_*^1 & y_*^2 & \cdots & y_*^d \\ y_*^1 & y_*^2 & \cdots & y_*^d \\ \vdots & \vdots & \ddots & \vdots \\ y_*^1 & y_*^2 & \cdots & y_*^d \end{bmatrix} + \begin{bmatrix} \lambda_1^1 & \lambda_1^2 & \cdots & \lambda_1^d \\ \lambda_2^1 & \lambda_2^2 & \cdots & \lambda_2^d \\ \vdots & \vdots & \ddots & \vdots \\ \lambda_{N_c}^1 & \lambda_{N_c}^2 & \cdots & \lambda_{N_c}^d \end{bmatrix} \times \mu, \tag{13}$$

where

$$\lambda_1^1 - \lambda_{N_c}^d \in (-1, 1). \tag{14}$$

Here any element $\lambda(\lambda = \lambda_1^1, \lambda_1^2, ..., \lambda_{N_c}^d)$ is a random number named mutation rate; the parameter $\mu$ is an antigen-dependent constant named step size. If the antigen pool is normalized, a value of 0.01 to 0.1 is recommended; the parameter $N_c$ is the number of the child clones of the $\boldsymbol{B_*}$(where $\boldsymbol{B_*} = (\boldsymbol{y_*}, \delta_*)$), a value of 500 to 1000 is recommended. These two parameters involve the step size and the scale of clonal variation. The smaller the step size and the larger the scale of clonal variation, the more ideal the memory B-cells are to be found, but the greater the consumption of computing resources. It was found that for a small training set, the change of parameters in the range of recommended values had little effect on the training results.

### 2.2.2. Optimal selection of memory B-cell

In order to find a memory B-cell from the cloned B-cell pool $\boldsymbol{BP_{clone}}$, a function named $f_o(\boldsymbol{cB},\boldsymbol{y})$ was defined as Eq. (15):

$$f_o(\boldsymbol{cB},\boldsymbol{y}) = \begin{cases} 1\ , & \text{if } f_A(\boldsymbol{cB},\boldsymbol{y}) \geq Ta_k; \\ 0\ , & \text{otherwise.} \end{cases} \tag{15}$$

where

$$\boldsymbol{cB} \in \boldsymbol{BP_{clone}},\ \boldsymbol{y} \in \boldsymbol{Agp_k}.$$

If $f_o(\boldsymbol{cB},\boldsymbol{y})$=1 holds, it means that antigen $\boldsymbol{y}$ is inside the cloned B-cell $\boldsymbol{cB}$. The optimal memory B-cell $\boldsymbol{M_s}$ obeys Eq. (16):

$$\begin{aligned} N_{M_s} &= \sum_{i=1}^{N} f_o(\boldsymbol{M_s},\boldsymbol{y_i}) \\ &= \boldsymbol{Max}\big[\sum_{i=1}^{N} f_o(\boldsymbol{cB},\boldsymbol{y_i}),\ \boldsymbol{cB} \in \boldsymbol{BP_{clone}}\big]. \end{aligned} \tag{16}$$

The memory B-cell $\boldsymbol{M_s}$ satisfying Eq. (16) could form a nonempty set, and our scheme was to select one randomly from it. At the same time, an antigen pool $\boldsymbol{Agp_k - left}$ would be generated, which was described by Eq. (17):

$$\boldsymbol{Agp_k - left} = \{\boldsymbol{y} \| f_A(\boldsymbol{M_s},\boldsymbol{y}) < \boldsymbol{Ta_k}, \boldsymbol{y} \in \boldsymbol{Agp_k}\}. \tag{17}$$

### 2.2.3. Forming a memory B-cell pool

This is a recursive process. Section 2.2.2 provides the method of producing a memory B-cell. By continuously calling the method of producing memory B-cell in Section 2.2.2, we can obtain memory B-cells one by one, and then form a memory B-cell pool. The process was as follows:

**step 1. Initialization :** $\boldsymbol{Agp_k - left} = \boldsymbol{Agp_k}, N_{left} = N; \boldsymbol{MBP_k} = \boldsymbol{\varnothing}, n = 0.$

**step 2. Antigen presentation :** Select an antigen $\boldsymbol{y_*}$ from the antigen pool $\boldsymbol{Agp_k - left}$, where $\boldsymbol{Agp_k - left} = \{\boldsymbol{y_1}, \boldsymbol{y_2}, \cdots, \boldsymbol{y_{N_{left}}}\}$, and $\boldsymbol{y_*}\ \boldsymbol{s.t.\rho(y_*)} = \boldsymbol{Max}[\boldsymbol{\rho(y)}, \boldsymbol{y} \in \boldsymbol{Agp_k - left}]$.

**step 3. B − cells cloning mutation proliferation :** $\boldsymbol{BP_{clone}} = (\boldsymbol{AT_{clone}}, \boldsymbol{\delta_{clone}})$.

**step 4. Find a memory B − cell $\mathbf{M_s}$ from $\mathbf{BP_{clone}}$ :** The optimal memory B-cell $\boldsymbol{M_s}$ obeys Eq. (16).

**step 5. Update the memory B − cell pool :** $\boldsymbol{MBP_k} = \{\boldsymbol{MBP_k}, \boldsymbol{M_s}\}$, $n = n + 1$.

**step 6. Update the antigen pool left :** $\boldsymbol{Agp_k - left} = \{\boldsymbol{y} \| f_A(\boldsymbol{M_s},\boldsymbol{y}) < \boldsymbol{Ta_k}, \boldsymbol{y} \in \boldsymbol{Agp_k}\}, N_{left} = N_{left} - N_{M_s}$.

**step 7. Termination condition :** if $N_{left} \leq 1$, output $\boldsymbol{MBP_k}$ and stop; else, return to **step 2**.

In step 7, an isolated antigen did not participate in the formation of memory B-cells because it might not really belong to the antigen pool $\boldsymbol{Agp_k}$ (It could be a noise).

### 2.3. Design of classifier

The memory B-cells in memory B-cell pool obtained from the OBPCSA are available for use for classification. The classification is performed in a $k$-nearest neighbor approach [27], which is like AIRS. In this paper, the $k$ of $k$-nearest neighbor in OBPCSCA is one.

It was described as follows: if there is an antigen $\boldsymbol{y}$ to be classified, the predicted result is denoted as $classify(\boldsymbol{y})$, where $\boldsymbol{M} \in (\boldsymbol{MBP_1} \cup \boldsymbol{MBP_2} \cup \ldots \cup \boldsymbol{MBP_c})$ and the subscripts of memory B-cell pools are the names of classes. The predicted result $classify(\boldsymbol{y}) = p$ ($p = 1, 2, \ldots, c$) obeys Eq. (18):

$$\begin{cases} \boldsymbol{M_*} \in \boldsymbol{MBP_p}; \\ \boldsymbol{M_*} = argmax\ f_A(\boldsymbol{M}, \boldsymbol{y}). \end{cases} \tag{18}$$

According to Eq. (3) and Eq. (4), because the affinity calculation contains the information of antigen distribution ($\theta_k$), the distance of the nearest neighbor here is replaced by affinity, with different weights of different classes.

## 3. Case studies

The performance of the algorithm was evaluated using two case studies. The classification performance of the algorithm was first tested on four benchmark data sets that are available from a machine learning repository[2]. In the second case, the algorithm was applied to the ball bearing fault diagnosis as a real world problem with the data sets from Case Western Reserve University (CWRU). The results of the case studies are given in the following sections.

### 3.1. Case study 1: comparison with other methods on benchmark data sets

In this part, the classification performance of the OBPCSCA was tested on four UCI benchmark data sets. In order to verify the comprehensive performance of OBPSCSA in classification accuracy and number of memory cells, the experimental results were compared with other two immune classification algorithms—AIRS [27] and AICSL [25]. Two immune systems are inspired by the immune network model and consist of artificial immune cells. We compared two important features of these algorithms: classification accuracy and number of memory cells. In addition, the performance of the OBPCSCA was also compared with the well-known classification techniques such as support vector machines, neural networks, fuzzy neural network, and C4.5.

#### 3.1.1. Data sets and experimental design

The descriptions of the data sets used are summarized in Table 1. Specifically, for the Iris data set, the four attributes are sepal length, sepal width, petal length, and petal width. One of the classes is linearly separable from the other two which are not linearly separable from each other. For Pima Indian Diabetes data set, the classification task is to determine if the patient tested positive for diabetes or not , according to these eight attributes. For Ionosphere data set, the classification task is to determine "good" and "bad" radar returns from the atmosphere, where "good" returns are those that indicate structure in the ionosphere and "bad" ones do not. For Sonar data set, the classification task is to determine whether a sonar signal bounced back from a metal or rock object.

Each simulation experiment consists of three stages: data processing stage, training stage, and testing stage. In order to better reflect the performance of the algorithm, we first did dimensionless data processing, specifically using min–max normalization. The normalized data would be divided into training set and testing set for training and testing, respectively. Considering the consistency of the control test conditions, the $k$-fold cross validation was run for each data set to compare the performance of our method to other classifiers that

[2]UCI (2007). UCI Machine Learning Repository [online]. Website https://archive.ics.uci.edu/ml/index.php [accessed 10 August 2020].

are reported in the literature. Figure 4 shows how the data sets were partitioned and how the classification performances were obtained. As shown in Figure 4, each data set was partitioned into $k$ portions, thereby generating $k$ different sets of data, each containing one portion as the testing set and other portions as the training set. The result of each run is the average of $k$-fold classification accuracy. More specifically, for Iris data set, a 5-fold cross validation scheme was employed with each result representing an average of three runs. For Pima Indian Diabetes data set and Sonar data set, the 10-fold cross validation scheme and 13-fold cross validation scheme were employed, respectively. For Ionosphere data set, 200 instances which are carefully split almost 50% positive and 50% negative are used for training with the remaining 151 as test instances, consisting of 125 "good" and only 26 "bad" instances. Except Iris data set, all results are an average of ten runs.

**Table 1**. Datasets used for experiments.

| Data set | Samples ($n$) | Attributes ($n$) | Classes($n$) | Class distribution |
|---|---|---|---|---|
| Iris | 150 | 4 | 3 | 50/50/50 |
| Pima Indian Diabetes | 768 | 8 | 2 | 500/268 |
| Ionosphere | 351 | 34 | 2 | 225/126 |
| Sonar | 208 | 60 | 2 | 97/111 |

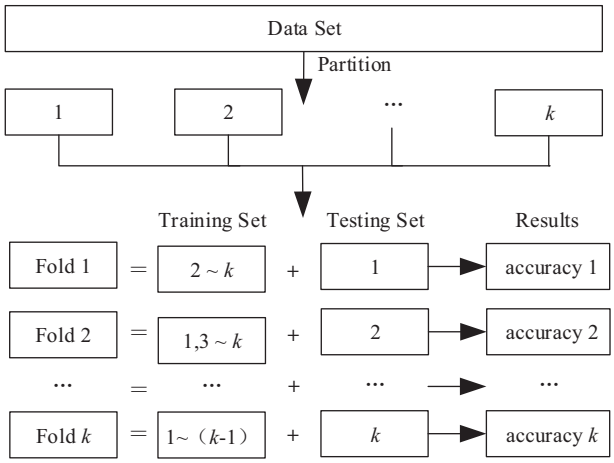


**Figure 4**. Partitioning of data set ($k$-fold cross validation).

### 3.1.2. Experimental results and analysis

As shown in Table 2, the performance of the OBPCSCA is compared to that of AIRS and AICSL, which shows that the OBPCSCA has a better balance between the number of memory cells and the accuracy of classification. In particular, compared with AIRS, the OBPCSCA can greatly reduce the number of memory B-cells on the premise of ensuring high classification accuracy.

Table 3 shows the location of our proposed methods in the well-known classification techniques in detail. In Table 3, the results of other algorithms were obtained from [30, 31] and this website of *Datasets used for classification: comparison of results*[3]. Just as shown in Table 3, the OBPCSCA ranks in the top 10 in terms

[3]Duch W (2010). Datasets used for classification: comparison of results [online]. Website http://fizyka.umk.pl/kis-old/projects/datasets.html [accessed 10 August 2020].

of classification accuracy on all four data sets, and ranks the second with classification accuracy of 90.46% on the Sonar data set. From the ranking of classification accuracy, the OBPCSCA is a very competitive classifier.

**Table 2**. Performance comparisons with AIRS and AICSL.

| Data set | Instances | AIRS | | AICSL | | OBPCSCA | |
|---|---|---|---|---|---|---|---|
| | | Accuracy(%) | cells | Accuracy(%) | cells | Accuracy(%) | cells |
| Iris | 120 | 96.70 | 30.9 | 98.14 | 24 | 97.11 | 24.7 |
| Pima Indian Diabetes | 691 | 74.20 | 273.4 | 74.99 | 20 | 76.15 | 223.3 |
| Ionosphere | 200 | 95.60 | 96.3 | 89.05 | 50 | 95.30 | 82.3 |
| Sonar | 192 | 84.90 | 177.7 | 87.50 | 60 | 90.46 | 90 |

**Table 3**. Comparisons of OBPCSCA and other classifiers results on benchmark data sets. "Acc" denotes the classification accuracy.

| Rank | Iris | | Pima Indian Diabetes | | Ionosphere | | Sonar | |
|---|---|---|---|---|---|---|---|---|
| | Method | Acc(%) | Method | Acc(%) | Method | Acc(%) | Method | Acc(%) |
| 1 | Grobian (rough) | 100.00 | Logdisc | 77.70 | 3-NN +simplex | 98.70 | TAP MFT Bayesian | 92.30 |
| 2 | MOGICA | 98.30 | IncNet | 77.60 | VSS 2 epochs | 96.70 | OBPCSCA | 90.46 |
| 3 | SSV | 98.00 | DIPOL92 | 77.60 | MLP+BP | 96.00 | Nave MFT Bayesian | 90.40 |
| 4 | C-MLP2LN | 98.00 | Linear Disc. Anala | 77.50 | OBPCSCA | 95.30 | Best 2-layer MLP+BP, 12 hidden | 90.40 |
| 5 | PVM 2 rules | 98.00 | SMART | 76.80 | C4.5 | 94.90 | MOGICA | 87.50 |
| 6 | OBPCSCA | 97.11 | ASI | 76.60 | RIAC | 94.60 | MLP+BP, 12 hidden | 84.70 |
| 7 | PVM 1 rule | 96.70 | Fischer Disc. Anala | 76.50 | MOGICA | 94.30 | MLP+BP, 24 hidden | 84.50 |
| 8 | FuNe-I | 96.70 | MLP+BP | 76.40 | SVM | 93.20 | 1-NN, Manhattan | 84.20 |
| 9 | NEFCLASS | 96.70 | OBPCSCA | 76.15 | Nonlinear perceptron | 93.00 | FSM | 83.60 |
| 10 | CART | 96.00 | LVQ | 75.80 | FSM +rotation | 92.80 | MLP+BP, 6 hidden | 83.50 |

### 3.2. Case study 2: application of the OBPCSCA in bearing fault diagnosis

In this section, in order to verify the feasibility of the OBPCSCA as an intelligent diagnosis method, we chose the ball bearing data set of Case Western Reserve University (CWRU) as the object of diagnosis, whose experimental setup was shown in Figure 5. The test bench mainly contains a motor, a torque transducer, and a dynamometer. The test bearing is used to support the motor shaft, 0.007" faults are introduced to bearing inner race, outer race and ball via electrodischarge machining.

For comparison with other intelligent methods, in the case, there are four vibration waveforms: 1 normal working condition and 3 malfunctioning working conditions with the bearing type, fault size, motor speed, and

motor load as shown in Table 4. Data was collected at 12,000 samples/second and at 48,000 samples/second for drive end bearing experiments, and in this section, we chose the former. As shown in Table 4, the bearing data used in the experiment include four categories: normal, inner race fault, outer race fault, and ball fault. There were three groups data of fault location in outer race (outer raceway faults located at 3 o′clock, at 6 o′clock and at 12 o′clock) and in this section, data of fault located at 6 o′clock was chosen. Under the same experimental conditions, the experimental results were compared with those in reference [30].

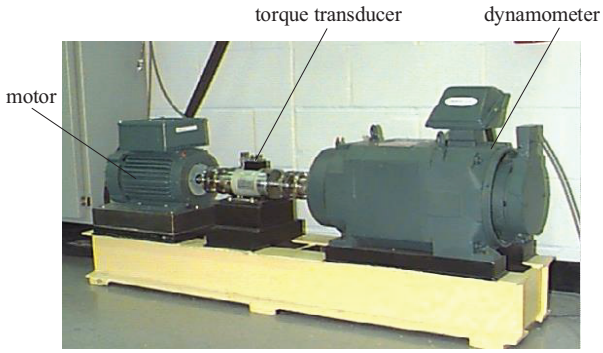


**Figure 5**. Experimental setup of CWRU test.

**Table 4**. The bearing type and parameters of ball bearing fault used in experiment.

| Bearing manufactuer | Bearing type | Fault ocation | Fault diameter d/(in) | Fault depth l/(in) | Motor speed rpm/(r/min) | Bearing location | Motor load ML/Hp |
|---|---|---|---|---|---|---|---|
| SKF | 6205 | Normal | 0.00 | 0.00 | 1730 | Drive end | 3 |
| SKF | 6205 | Inner race | 0.007 | 0.01 | 1721 | Drive end | 3 |
| SKF | 6205 | Outer race | 0.007 | 0.01 | 1725 | Drive end | 3 |
| SKF | 6205 | Ball | 0.007 | 0.01 | 1725 | Drive end | 3 |

### 3.2.1. Data processing

Because bearing data is one-dimensional vibration signal, it cannot be directly used in the OBPCSCA. According to the processing method in [30], we have processed the data accordingly. Specifically, the normal and fault related features were also decomposed through seven layers "db3" wavelet transform and high frequency of wavelet energy feature extraction with length of each sample 2048 points. Then, many 7d energy eigenvectors representing the normal and fault conditions of bearing are formed. For comparison purpose, we got a $180 \times 7$ matrix for each working condition. In addition, in each working condition, 80 samples were randomly selected as training data, and the rest 100 samples were selected as testing data.

### 3.2.2. Application on data set obtained

After processing the bearing data, we obtained a data set containing 4 classes, which was named $\boldsymbol{3Hp}$. Specifically, the following experiments on the $\boldsymbol{3Hp}$ data set was performed and repeated the experiments were 3 times:

1) Min–max normalization was used on the $\boldsymbol{3Hp}$ data set;

2) 80 samples were randomly selected from 180 samples of each class to form the training set with a sample size of 320, and the remaining 400 samples comprised the testing set;

3) Collected prediction results of testing set on OBPCSCA.

### 3.2.3. Contrast and analysis

In this part, the classification accuracy of MOGICA [30] and OBPCSCA on the ***3Hp*** data set was compared. The two experimental treatments for comparison were the same, and the classification accuracy (which of each class was the average of three experiments) and standard deviation were shown in Table 5. From the results of Table 5, OBPCSCA had the worst diagnostic accuracy of 97.67% for outer race fault. Although the accuracy of OBPCSCA in the diagnosis of outer race fault is 0.63% lower than that of MOGICA, it has obvious advantages in the diagnosis of ball fault. Combined with the results of MOGICA, the data on "outer race fault" and "ball fault" are close in the feature space. In the three tests of randomly dividing training set and testing set, the memory B-cell pools obtained from the three trainings are different due to the incremental learning mode of OBPCSCA, which results in the missed diagnosis of very few data on "outer race fault". However, from another point of view, incremental learning mode lays the foundation for the realization of online learning, which is our next research direction.

**Table 5**. The bearing diagnosis accuracy of two methods on different fault type and the same level of fault severity.

| Type of samples | No. of training samples | No. of testing samples | MOGICA | OBPCSCA |
|---|---|---|---|---|
| | | | Accuracy rate (%) | Accuracy rate (%) |
| normal samples | 80 | 100 | 100 ± 0.00 | 100 ± 0.00 |
| inner race fault with size 0.007" | 80 | 100 | 100 ± 0.00 | 100 ± 0.00 |
| outer race fault with size 0.007" | 80 | 100 | 98.3 ± 0.20 | 97.67± 0.82 |
| ball fault with size 0.007" | 80 | 100 | 98.5 ± 0.10 | 100 ± 0.00 |

## 4. Conclusion and future work

In this study, we proposed an intelligent diagnostic method based on optimizing B-cell pool clonal selection classification algorithm. The algorithm inspired by immune system provides a method to construct B-cell pools, in which each B-cell has a scale-adaptive radius. In addition, the algorithm uses clonal selection mechanism to optimize memory B-cell pool, and greedy strategy is adopted in the whole optimization process. In order to verify the performance of the proposed algorithm, simulation experiments were conducted on four UCI benchmark data sets. The advantage of the algorithm is that it is suitable for the learning of small samples and has no hyperparameters. At the same time, the algorithm uses hyperspheres to divide the feature space, which makes the classification interface more flexible and has the potential of online learning. The comparison with some general algorithms results show that our method is promising. In addition, the OBPCSCA was applied to ball bearing diagnosis in Section 3.2 and the effectiveness of the method is further proved.

Future work lies in improving the classification performance of OBPCSCA for data sets with a large number of attributes and classes. Specifically, it will be the future research direction to realize online learning through incremental learning mode, to study the correlation between features, to establish the relationship between features, and to improve the processing ability of algorithm for high-dimensional data.

**Acknowledgment**

This work was supported by the National Natural Science Foundation of China under Grant 61603238, Grant 11802168, and Grant 51575331.

## References

[1] Liu Y. The research of anomaly detection and fault diagnosis based on artificial immune system. PhD, Shanghai University, Shanghai, China, 2013.

[2] Xiao S, Liu S, Jiang F, Song M, Cheng S. Nonlinear dynamic response of reciprocating compressor system with rub-impact fault caused by subsidence. Journal of Vibration and Control 2019; 25(11): 1737-1751. doi: 10.1177/1077546319835281

[3] Li B, Zhao Z, Guan Y, Ai N, Dong X et al. Task placement across multiple public clouds with deadline constraints for smart factory. IEEE Access 2018; 6: 1560-1564. doi: 10.1109/access.2017.2779462

[4] Zhang H. The study on artificial immune method of equipment abnormal degree detection and fault identification method. PhD, Shanghai University, Shanghai, China, 2014.

[5] Lei Y, Jia F, Kong D, Lin J, Xing S. Opportunities and challenges of machinery intelligent fault diagnosis in big data era, Chineese Journal of Mechanical Engineering 2018; 54(05): 94-104.

[6] Ben Ali J, Chebel-Morello B, Saidi L, Malinowski S, Fnaiech F. Accurate bearing remaining useful life prediction based on Weibull distribution and artificial neural network. Mechanical Systems and Signal Processing 2015; 56-57: 150-172. doi: 10.1016/j.ymssp.2014.10.014

[7] Chen X, Shen Z, He Z, Sun C, Liu Z. Remaining life prognostics of rolling bearing based on relative features and multivariable support vector machine. Proceedings of the Institution of Mechanical Engineers Part C-Journal of Mechanical Engineering Science 2013; 227(12): 2849-2860. doi: 10.1177/0954406212474395

[8] Kong D, Chen Y, Li N, Duan C, Lu L et al. Relevance vector machine for tool wear prediction. Mechanical Systems and Signal Processing 2019; 127: 573-594. doi: 10.1016/j.ymssp.2019.03.023

[9] Wu Z, Zhang M, Chen Z, Wang P. Youla parameterized adaptive vibration suppression with adaptive notch filter for unknown multiple narrow band disturbances. Journal of Vibration and Control 2019; 25(3): 685-694. doi: 10.1177/1077546318794539

[10] He B, Wang S, Liu Y. Underactuated robotics: A review. International Journal of Advanced Robotic Systems 2019; 16(4): 1-20. doi: 10.1177/1729881419862164

[11] Rush AM, Chopra S, Weston J. A neural attention model for sentence summarization. In: Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing; New York, USA; 2015. pp. 379-389.

[12] He B, Shao Y, Wang S, Gu Z, Bai K. Product environmental footprints assessment for product life cycle. Journal of Cleaner Production 2019; 233: 446-460. doi: 10.1016/j.jclepro.2019.06.078

[13] Bayar N, Darmoul S, Hajri-Gabouj S, Pierreval H. Fault detection, diagnosis and recovery using Artificial Immune Systems: A review. Engineering Applications of Artificial Intelligence 2015; 46: 43-57. doi: 10.1016/j.engappai.2015.08.006

[14] Benhamini E, Coico R, Sunshine G. Immunology-A Short Course, USA: Wiley-Liss, Inc., 2000.

[15] Forrest S, Perelson A, Allen L, Cherukuri R. Self-nonself discrimination in a computer. In: Proceedings of the International Symposium on Security and Privacy; New York, USA; 1995. pp.1-20. doi: 10.1109/RISP.1994.296580

[16] Laurentys CA, Ronacher G, Palhares RM, Caminhas WM. Design of an artificial immune system for fault detection: a negative selection approach. Expert Systems with Applications 2010; 37(7): 5507-5513. doi: 10.1016/j.eswa.2010.02.004

[17] González FA, Dasgupta D. Anomaly detection using real-valued negative selection. Genetic Programming and Evolvable Machines 2003; 4(4): 383-403. doi: 10.1023/A:1026195112518

[18] Ji Z, Dasgupta D. Real-Valued Negative Selection Algorithm with Variable-Sized Detectors. Berlin, Heidelberg: Springer, 2004.

[19] Stibor T, Mohr P, Timmis J. Is negative selection appropriate for anomaly detection. In: Gecco 2005: Genetic and Evolutionary Computation Conference; Washington DC, USA; 2005. pp. 321-328.

[20] Esponda F, Forrest S, Helman P. A formal framework for positive and negative detection schemes. Ieee Transactions on Systems Man and Cybernetics Part B-Cybernetics 2004; 34(1): 357-373. doi: 10.1109/tsmcb.2003.817026

[21] Timmis J. Artificial immune systems: A novel data analysis technique inspired by the immune network theory. PhD, University of Wales, England, 2000.

[22] De Castro LaVZF. An evolutionary immune network for data clustering. IEEE Explorer 2000; 1: 84-89. doi: 10.1109/SBRN.2000.889718

[23] Watkins A, Boggess LC. A resource limited artificial immune classifier. IEEE Explorer 2002; 1: 926 - 931. doi: 10.1109/CEC.2002.1007049

[24] Brownlee J. Clonal selection theory & clonalg the clonal selection classification algorithm (CSCA). Swinburne University of Technology, 2005.

[25] Aydin I, Karakose M, Akin E. Artificial immune classifier with swarm learning. Engineering Applications of Artificial Intelligence 2010; 23(8): 1291-1302. doi: 10.1016/j.engappai.2010.06.007

[26] Sompayrac L. LECTURE 1: An Overview. How the Immune System Works. 4nd ed. USA: John Wiley & Sons, Ltd, 2012.

[27] Watkins A, Timmis J, Boggess L. Artificial immune recognition system (AIRS): an immune-inspired supervised learning algorithm. Genetic Programming and Evolvable Machines 2004; 5(3): 291-317. doi: 10.1023/B:GENP.0000030197.83685.94

[28] Cortes C, Vapnik V. Support-vector networks. Machine Learning 1995; 20(3): 273-297. doi: 10.1023/A:1022627411411

[29] Mo H. Human Immune System: Artificial Immune System. 1st ed. Beijing, China: Science Press, 2009.

[30] Zhang HL, Zhai YY, Liu SL, Li D, Wang B et al. A mass optimizing group identification classification algorithm (MOGICA) used for intelligent fault diagnosis. Journal of Intelligent & Fuzzy Systems 2016; 31(3): 1745-1757. doi: 10.3233/jifs-152168

[31] Leung K, Cheong F, Cheong C. Generating compact classifier systems using a simple artificial immune system. IEEE Transactions on Systems Man and Cybernetics Part B-Cybernetics 2007; 37(5): 1344-1356. doi: 10.1109/tsmcb.2007.903194