Constructing Detectors in Schema Complementary Space for Anomaly Detection

Xiaoshu Hang and Honghua Dai

School of Information Technology, Deakin University, Victoria , 3125 , Australia
{xhan,hdai}@deakin.edu.au

Abstract: This paper proposes an extended negative selection algorithm for anomaly detection. Unlike previously proposed negative selection algorithms which directly construct detectors in the complementary space of self-data space, our approach first evolves a number of common schemata through coevolutionary genetic algorithm in self-data space, and then constructs detectors in the complementary space of the schemata. These common schemata characterize self-data space and thus guide the generation of detection rules. By converting data space into schema space, we can efficiently generate an appropriate number of detectors with diversity for anomaly detection. The approach is tested for its effectiveness through experiment with the published data set *iris*.

1 Introduction

The natural immune system has inspired scientists a great research interest because of its powerful information processing capability. It protects biologic bodies from disease-causing pathogens by pattern recognition, reinforcement learning and adaptive response[1]. The idea of applying computational immunology to the problem of computer/network security derives from the inspiration that virus, network intrusion is analogous to pathogens to human bodies[2]. By generating a number of detectors to monitor every data packet in a network or to monitor every transaction data in a financial system, an anomaly detection system detects, notifies and automatically responses to anomalous activities.

Negative selection algorithm, proposed by Stephanie Forest and her research group in 1994, has been considered to be a highly feasible technique to anomaly detection and has been successfully applied to computer viruses/ network intrusion detection[3], tool breakage detection[4], times-series anomaly detection[5], and Web document classification[6]. The most striking features of the algorithm are that it does not require any prior knowledge of anomalies and can implement distributed anomaly detection in a network environment. For a given data set *S* to be protected, a set of detectors R is generated in a way that each detector *d* (a binary string) does not match any string *s* in *S*. The negative selection algorithms work in a straightforward way to generate the repertories *R*. It randomly generates a string *d* and matches *d* against each string in S. If *d* does not match any string in S then store *d* in R. This process is repeated until we have enough detectors in *R* to ensure the desired protection. This generate-and-test method requires sampling a quite large number of candidate detectors and the computational complexity is exponential to the size of self data set *S* [7].

Variations of negative selection algorithm have also been investigated mainly focusing on representation schemes, detector generation algorithm and matching rules. Representation scheme has been explored including hyper-rectangle-rule detectors[8], fuzzy-rule detectors[9], and hyper-sphere-rule detectors[10]. Their corresponding detector generation algorithms are negative selection with detection rules, negative selection algorithm with fuzzy rules and randomised real-valued negative selection. Match rules in negative selection algorithm include r-contiguous matching, r-chunk matching, Hamming distance matching, and its variation Rogers and Tanimoto matching[11].

The problem that negative selection algorithms face is how to efficiently generate an appropriate number of detectors. The generate-and-test approach has the drawback of exponentially computational complexity with respect to the size of self data. The computational cost may be intolerable when the size of self data, especially the dimen sion size, is large. In this paper, we propose an extended negative selection algorithm which combines computational immunology and coevolutionary genetic algorithm for anomaly detection. Unlike previously proposed negative selection algorithms that directly construct detectors in the complementary space of self data space, it first transforms self-data space into self-schema space by evolving a group of schemata in self-data space, and then creates an appropriate number of detectors in the schema complementary space with the guidance of the common schemata evolved. By compressing self data space into self schema space, we can efficiently generate an appropriately sized set of detectors with diversity for anomaly detection.

The rest of the paper is organized as follows. In section 2, first, the GA schema and the representation of real-valued based detector and schema are introduced, and then a coevolutionary genetic algorithm for acquiring schemata from self data space is presented. Section 3 describes the extended negative selection algorithm. Section 4 demonstrates some experimental results, and section 5 presents the conclusions.

2 From Data Space to Schema Space

2.1 GA Schema

What is a GA schema? A GA schema can be viewed a template specifying groups of chromosomes with a common characteristic, or a partition of genome space, or a description of hyperplanes through genome space or a set of search points sharing some syntactic feature. A schema is usually represented as a string of symbols taken from the alphabet {0,1,#}. The character # is explained as “don’t care”, so that a schema represents a common pattern contained in a number of chromosomes. For example, a number of chromosomes contain a schema 1#00##11 as follows:

```
1 1 0 0 0 0 1 1
1 0 0 0 1 0 1 1
 ...............
1 1 0 0 0 1 1 1
```

The number of non-# symbols in a schema H is called the order *O(H)*, and the distance between the outermost two non-# symbols is called the defining length *L(H)* of the schema. The length of a schema is important because it determines the likelihood of a member of the schema being disrupted by crossover--the further apart its genomes are, the less likely they are to stay together. Suppose the length of individuals be l, accordingly the number of variable genes in a chromosome is $l-O(H)$. In general, GAs employ binary encoding schemes, so the total search space is 2^l, where l denotes the length of chromosomes. If there exists one gene which is important and keeps invariable at the same position of all the chromosomes, the total search space is reduced to the half, and is further reduced to 2^{l-k} if there are k such important genes. In the process of evolution, a new schema is likely to be created by mutation rather than by crossover. Holland's schema theorem explains how schemata are expected to propagate from generation to generation by selection, crossover and mutation. Schemata theorem is also used to explain why GAs realise an optimum search strategy[14].

Generally there is more than one schema in the searching space. Each schema represents a common pattern hidden in a group of individuals. And one individual may be covered by multiple schemata. For example, considering in the three dimensional space of $2^{l=3}$ (see Figure 1), there are six surface hyperplains, each of which has one fixed gene at the same position of the four chromosomes such that a schema can be derived. The top hyperplain has the schema: #1# and the front hyperplain contains the schema: ##1. The two diagonal hyperplains do not form schemata.

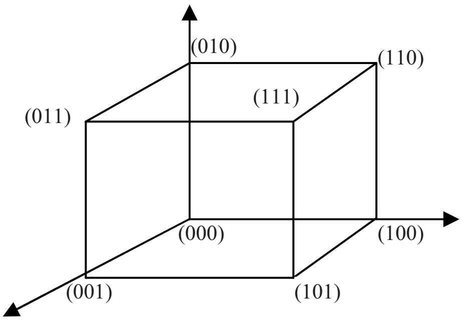

Fig. 1. Individuals and schemata in three dimensional space

Finding common schemata is also inspired from the natural immune system. Bacteria are inherently different from human cells, and many bacteria have cell walls made from polymers that do not occur in humans, so we can expect the immune system to recognize bacteria partially on the basis of the existence of these unusual molecules[15]. Common schemata represent generic properties of the antigen population. With this as motivation, we can construct antibodies which are complementary to the antigens in the bits of schema so that one antibody can recognise multiple antigens. This may explain to some extent why the natural immune system is able to recognize an enormous number of foreign pathogens with relatively limited resources.

2.2 Representation of Schemata

In this section, we first introduce the notation of anomaly detection, and then we present the representations of detectors and schemata which characterise both self and non-self data space.

A problem space $S=x_1\times x_2\times\cdots\times x_n$ is a n-dimensional vector space, where x_j is either a categorical feature or a numeric feature. $s_1^*, s_2^*,\cdots, s_n^*$ are the definite states in the space S, where $*\in\{\text{normal, abnormal}\}$. The normal state subspace is called self-state set denoted by $SS\subset S$, and the non-self state subspace NS is defined as the complement space of SS and thus $NS=S-SS$. The two subspaces are described as follows:

$$\textit{Self- state space: } SS=\{ s_i^* \mid *= normal\,,\ i=1,2,,,p\}$$
$$\textit{Non-self space: } NS=\{ s_j^* \mid *= \text{abnormal}, j=1,2,,,q\}$$

where $SS\cup NS=S$ *and* $SS\cap NS=\varnothing$.

The characteristic function χ_{self} for differentiating self and non-self is defined as follows:

$$\chi_{self}(s_j)=\begin{cases}1, & if\ s_j\in SS\\ 0, & if\ s_j\in NS\end{cases}$$

As we mentioned above, a detector can be represented as a detection rule, the structure of which is described as follows:

$$x_1\in[val_1^1, val_2^1]\wedge\ldots\wedge x_j=d_j\wedge x_m\in[val_1^m, val_2^m]\rightarrow abnormal$$

$$x_1\in[val_3^1, val_4^1]\wedge\ldots\wedge x_j=d_j\wedge x_m\in[val_3^m, val_4^m]\rightarrow abnormal$$

where $x_i\in[val_1^i, val_2^i]$ represents that feature x_i is a real-valued feature and $x_j=d_j$ indicates that feature x_j is a categorical feature. Each rule(self or non-self) defines a hypercube which covers some states in the descriptor space, The detection rules cover non-self data space, while self rules characterise self-data space (see Figure 2).

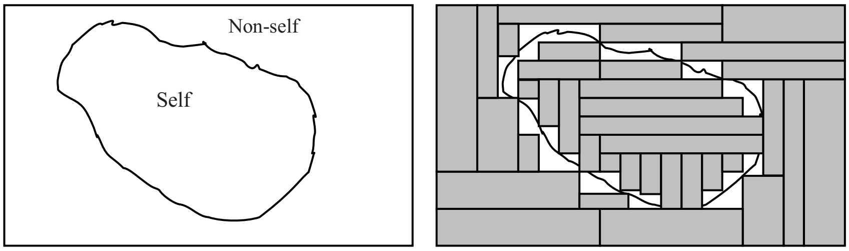

Fig. 2. A two-dimension space characterised by schemata

In reality, both self data space and non-self data space contain schemata. Generally, abnormal events of the same kind may have common characteristics which can be considered as common schemata. Thesecommon schema ta represent the dimension-reduced subspace to which the abnormal events belong. The same situation exists in self data space, that is, there may be some characteristic subspaces, each of which may contain a schema. For example, we get three rules which characterize normal events as follows:

$$x_1 \in [val_1^1, val_2^1] \wedge x_2 \in [val_1^2, val_2^2] \wedge x_3 \in [val_1^3, val_2^3] \wedge x_4 \in [val_1^4, val_2^4] \rightarrow normal$$

$$x_1 \in [val_1^1, val_2^1] \wedge x_2 \in [val_1^2, val_2^2] \wedge x_3 \in [val_1^3, val_2^3] \wedge x_5 \in [val_1^5, val_2^5] \rightarrow normal$$

$$x_1 \in [val_1^1, val_2^1] \wedge x_2 \in [val_1^2, val_2^2] \wedge x_4 \in [val_1^4, val_2^4] \wedge x_5 \in [val_1^5, val_2^5] \rightarrow normal$$

The common schema induced from the above three rules is:

$$x_1 \in [val_1^1, val_2^1] \wedge x_2 \in [val_1^2, val_2^2]$$

Definition 1. A real-valued based representation of a schema *r* is defined as the conjunction of feature-interval pairs as follows:

$$r = x_1 \in [val_1^1, val_2^1] \wedge ... \wedge x_k \in [val_1^k, val_2^k]$$

Definition 2. The coverage of a schema *r* is defined as the ratio of the number of self states contained in the hypercube determined by *r* to the total number of self states in self-state space *SS*:

$$Coverage(r) = \frac{|\{s_i \in SS | s_i \in r\}|}{|SS|} \qquad (1)$$

Definition 3. The volume of the hypercube that a schema $r = x_1 \in [val_1^1, val_2^1]$ and … and $x_l \in [val_1^l, val_2^l]$ determines is represented as:

$$Volume(r) = \prod_{i=1}^{l} (val_2^i - val_1^i) \qquad (2)$$

Since each numeric variable is normalized between 0 and 1, the volume that a schema determines is less than 1, and the longer a schema is, the smaller of the volume it determines. Thus the fitness of a schema *r* is represented as:

$$fitness(r) = \alpha\, Coverage(r) + \beta\, Volume(r) \qquad (3)$$

where α, β are the weights and $\alpha + \beta = 1$.

2.3 Finding Common Schemata by Coevolutionary GA

We use coevolutionary genetic algorithm to evolve a number of schemata that are randomly initialised and will finally cover the entire self-data space.

In [15], Stephanie Forrest *et al.* proposed and solved a problem that whether or not the conventional GAs can find multiple common schemata. As we know that the conventional GAs have the property of convergence, we face the problem that how conventional GAs can maintain multiple sub-population, or how conventional GAs can find multiple peaks in the meantime. Their experiments showed if the population of chromosomes are initialised with schemata of the correct answers, GAs can maintain the correct schemata after some generations. But the right problem to be solved is "Can the conventional GAs find all the schemata in different parts of the space with a randomly initialised population?" and if can, then "what are the key parameters?". They found that the conventional GAs can discover and maintain multiple schemata, and also found the population size is an important parameter to maintain the diversity, that is, a bigger size of population can find more schemata. Further more, the mutation rate plays an important role in maintaining the diversity of the population.

In this paper, we exploit coevolutionary genetic algorithm proposed by Potter and De Jong in [16]. The population consists of a number of non-interbreeding subpopulation of species. Each species represents only a partial solution to the problem, and there is neither cooperation nor competition among subpopulations. Although nothing explicitly prevent multiple subpopulations from containing the identical schema, in practice, each subpopulation tends to be dominated by each species. In our work, each subpopulation is randomly initialised with a species which will convrege on a specific schema after some generations of evolution. All the schemata form a schema space of self data, and the detectors are produceded in the schema complementary space. This approach has been proven to use a variety of settings and has been applied to concept learning[16] and Web document classification[6]. The chromosome is designed to be composed of 4 genes as follows.

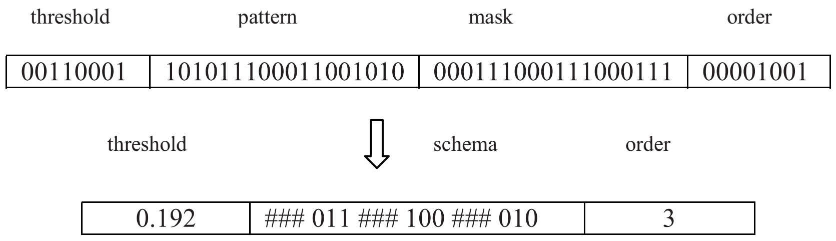

Fig. 3. Binary scheme for evolving common schemata

The first 8-bit gene is the threshold. Its real value is calculated first by converting the gene to a decimal integer and then dividing this integer by 255. A match between a schema and an instance is considered to occur when the binding strength is greater than the threshold. The pattern gene and the mask gene are the same length and are combined to form a schema. To employ binary coding scheme, each numeric feature is discretized into several different intervals, maximally 8 in our approach so that it can be represented by a 3-bit binary string(000~111). Therefore, both the pattern and the mask gene have 3 times bits as many as the number of the feature vector. The mask gene is used to overlaid the pattern gene, that is, a three-mask bits of “111” keeps the corresponding bit in the pattern gene unchanged, and a three-mask bits of “000” generates a “don’t care ” schema value. The advantage that is captured by this representation is the ability of a schema matching a wider range of instances. In this way the schema is formed by copying the pattern gene and is modified by mutating the mask gene. The last gene in the genome represents the order of the potential schema. Its value is the total number of non-# in the schema. As we mentioned above, each feature in our approach is represented by 3 binary bits, so the actual order of the schema is the integer divided by 3. The fitness of each individual is calculated by formula (3), and the algorithm for evolving schemata is described as follows:

```
Algorithm: Coevolutionary genetic algorithm for evolving schemata
   Input :  A feature vector table, a group of parameters
   Output: a group of schemata
   1   discretize numeric feature vector and encode in a binary string;
   2   create the first species;
   3   while the number of species is less than a given number
   4       For each species
   5           Bind each individual to all the feature vectors;
   6           Calculate each individual's fitness;
   7           Do selection, crossover, and mutation;
   8       endfor
   9       Calculate the total fitness of the population;
   10      if the total fitness fails to increase for a few consecutive generations
   11          add a new species to the population;
   12          remove the individuals that do not contribute to the total fitness;
   13  endwhile
   14  decode each species into a common schema;
```

The algorithm starts with initializing a single species which represents a potential common schema, and new species are added into the population till the total number of species reaches the specified value. In each generation, the fitness of a single individual is determined by its binding ability, and the fittest individual in each species is generated and then the total fitness of the population is calculated. Child species are created by selecting two parents from the same species using fitness-proportionate selection with balanced linear scaling, and then by using uniform crossover and bit flipping mutation. As we mentioned above, the population tends to maintain the correct schema if it has initially been given a correct schema, and also has the capability to evolve out a new schema in the case of being randomly initialized.

3 Generating Detectors

We propose an extended negative selection algorithm in which detection rules are not completely randomly generated. First, it prefers choosing the features that appear more frequently in the set of schemata to the features that do not appear at all or appear less frequently. The reason for this is that a feature occurring more frequently in the schemata must be more important in the problem space and thus has a higher likelihood to be included in the detection rules. That is, a frequently-occurred feature with its infrequently-occurred interval-values is more likely to be selected into a detection rule. Second, when a detection rule is generated, it matches against the common schemata (see Figure 4). If a rule does not contain any common schema, it is considered as a detector and is stored in the detection rule set, otherwise it is rejected. This process is repeated until an appropriate number of detection rules are obtained. In the monitoring phase, each instance is matched against the detection rules. A change is considered to be detected if any match occurs.

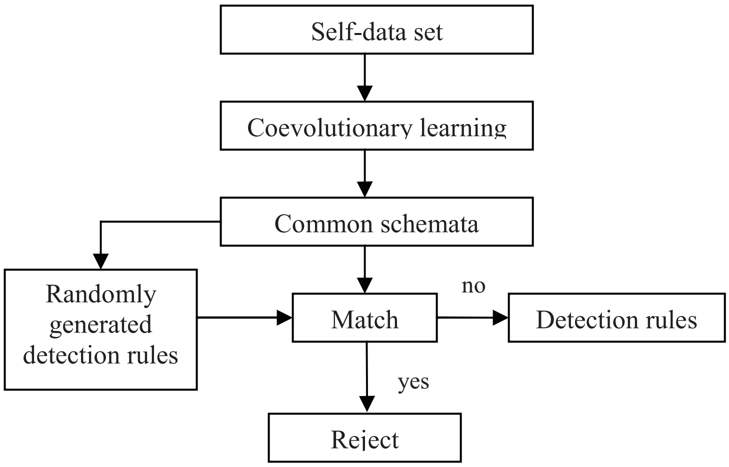

Fig. 4. Diagram of generating detection rules

4 Experimental Results

We perform this experiment with the published data set *iris* which consists of 5 numeric features and 150 instances. The last feature is the class label of three classes. We use the 100 instances of class 1 and class 3 as self data, and take the 50 instances of class 2 away as non-self data. Each of the four conditional numeric features is discreted as maximally 8 different intervals and encoded by 3-bit binary strings. The data set in coevolutionary phase includes the four conditional features of class 1 and class 3. The experimental parameters are set up as: Species number=9, Species size= 100, crossover rate=0.6, mutation=0.3, threshold=1.0, $\alpha = 0.7$ and β=0.3.

Table 1. Evolved common schemata covering class 1 and class 3

No.	Schema	Class	Coverage	Fitness
1	PL∈[1.00, 1.90] and PW∈[0.10, 0.40]	1	0.49	0.371
2	PL∈ [5.00,5.60] and SL∈ [5.70, 6.30]	3	0.12	0.181
3	PL∈[5.00, 5.60] and SL∈ [6.40, 6.90]	3	0.13	0.182
4	PL∈[5.00, 5.60] and PW∈[1.80, 2.10]	3	0.14	0.182
5	PL∈[5.00, 5.60] and PW∈[2.20, 2.50]	3	0.08	0.110
6	PL∈[5.70, 6.90] and PW∈[1.80, 2.10]	3	0.09	0.114
7	PL∈[5.70, 6.90] and PW∈[2.20, 2.50]	3	0.09	0.114
8	PL∈[5.00, 5.60] and SW∈[2.40, 3.20]	3	0.22	0.298
9	PW∈[2.20, 2.50] and SL∈[6.40, 6.90]	3	0.09	0.113

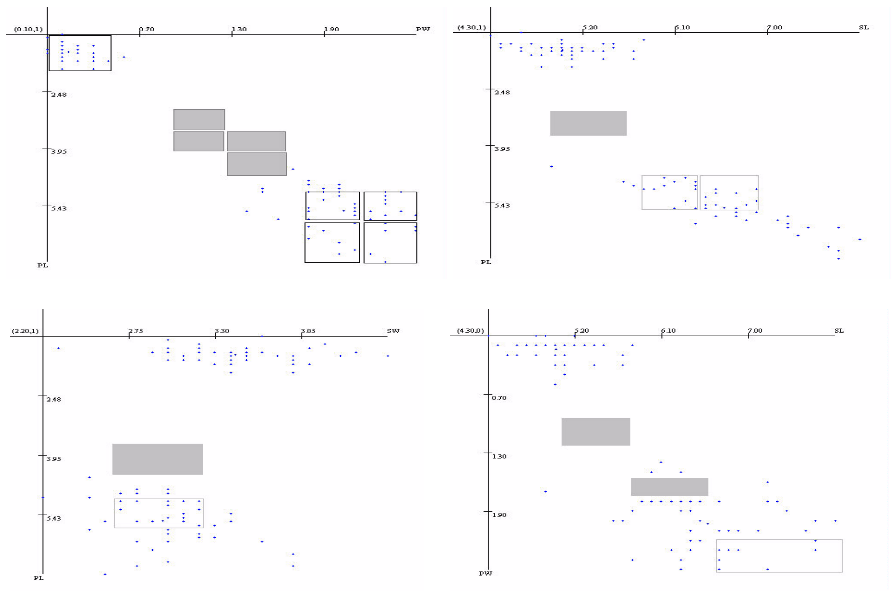

Fig. 5. Distributions of schemata and detectors in the two dimensional space

It is interesting to find that 49 of 50 instances in class 1 are covered by the schema: PL∈[1.00, 1.90] and PW∈[0.10, 0.40], and a group of schemata for class 3 are produced after some generations of evolution. Note that the binding threshold is setup to 1.0, which means an exact matching between a schema and an instance is required. In the second phase, the schemata guide the generation of the detection rules by getting rid of any detector which matches any of the above schemata. Eight generated detection rules are shown in Table 2, and the schemata and detectors are distributed in the two dimensional space showed in Figure 5, from which we can see that some small schemata and detectors can be merged into big ones. The transparent rectangles are the schemata in the two dimensional self data space, while the gray-filled rectangles are the detection rules.

Table 2. Generated detection rules

No	Detectors
1	PL∈[3.00, 3.60] and PW∈[0.90, 1.20] → class 2
2	PL∈[3.70, 4.30] and PW∈[0.90, 1.20] → class 2
3	PL∈[3.70, 4.30] and PW∈[1.30, 1.50] → class 2
4	PL∈[4.40, 4.90] and PW∈[1.30, 1.50] → class 2
5	PL∈ [3.70,4.30] and SW∈[2.40, 3.20] and SL∈[5.00, 5.60]→ class 2
6	SL∈[5.00, 5.60] and PW∈[0.90, 1.20] → class 2
7	PL∈[3.00, 3.60] and SL∈ [5.00, 5.60] and PW∈[0.90,1.20]→ class 2
8	SL∈[5.70,6.30] and PW∈[1.60, 1.70] → class 2

The number of instances detected by each detection rule is shown in the left chart of figure 6 and the total number of instances detected by all the detection rules exceeds 50, which means some instances are detected by more than one detectors. The right curve shows the detection rate and the false alarm rate, which are computed at different numbers of detectors. The first false positive error begins to occur when the number of detectors exceeds 7, which means a self data is detected as an anomaly. The false alarm rate increases as the number of detection rules grows.

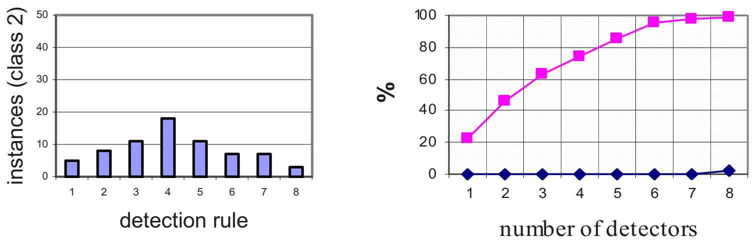

Fig. 6. Coverage of each detection rule(left), detection rate and false alarm

5 Conclusions and Discussions

To overcome the drawback of inefficiency that is created by the approach generate-and-test, we propose an extended negative selection algorithm for anomaly detection, which first evolves a number of common schemata through coevolutionary genetic algorithm in self-data space, and then constructs detectors in the schema complementary space. These common schemata characterize self-data space and thus guide the creation of detection rules. We use the published data set *iris* to test the effectiveness of our approach. The preliminary conclusions are obtained as follows:

- Converting data space into schema space and then constructing detectors in the schema complementary space is a novel and effective method. It eliminates the exponentially computational cost that is created by generate-and-test.
- Some methodologies of computational intelligence can be exploited to learn schemata from self data set. In this paper, we use coevolutionary genetic algorithm to evolve a group of schemata in the self data space.
- The discreteness of numeric features affects the number of schemata evolved and thus affects the number of the final detection rules. Generally, the more intervals a numeric feature is discretized, the more schemata and thus the more detection rules are generated.
- Our approach does not rely on the structured representation of the data and thereby is applied to the problem of general anomaly detection.

References

1. Steven A. Hofmeyr, and S. Forrest, “Architecture for an artificial immune system”, IEEE transaction on Evolutionary Computation, 8(4) (2000) 443-473.
2. Dipankar Dasgupta and Fabio Gonzalez “An immunity-based Technique to Characterize Intrusions in Computer Networks”, IEEE transaction on evolutionary computation 6(3),pages 1081-1088 June 2002.
3. Paul K. harmer, Paul D. Williams, Gregg H.Gunch and Gary B.Lamont, “An Artificial Immune System Architure for Computer Security Application” IEEE transaction on evolutionary computer, vol.6. No.3 June 2002.
4. Dipankar Dasgupta and Stephanie Forrest, “Artificial immune system in industrial application”, In the proceeding of *International conference on Intelligent Processing and Manufacturing Material (IPMM). Honolulu, HI (July 10-14, 1999).*
5. Dipankar Dasgupta and Stephanie Forrest, “Novelty Detection in Time Series data using ideas from Immunology”, *In the proceedings of the 5th International Conference on Intelligent Systems, Reno, June 19-21, 1996*

6. Jamie Twycross and Steve Cayzer, “An Immune-based approach to document classification”, http://citeseer.nj.nec.com/558965.html.
7. S.Forrest, A.Oerelson, L.Allen, and R.cherukuri. “Slef-nonself discrimination in a computer”, *In the proceedings of IEEE symposium on research in security and privacy, 1994*.
8. Fabio A.Gonzalez and Dipankar Dasgupta, “An Immunogenetic Technique to detect animalies in network traffic”, *In the proceeding of GECCO 2002: 1081-1088*
9. Jonatan Gomez, Fabio Gonzalez and Dipankar Dasgupta, “An Immune-Fuzzy Approach to Anomaly detection”, *In Proceedings of The IEEE International Conference on Fuzzy Systems, St. Louis, MO, May 2003.*
10. Fabio Gonzalez, Dipankar Dasgupta and Luis Fernando Nino, “A Randomized Real-Value Negative Selection Algorithm”, ICARIS-2003.
11. M. Ayara, J. Timmis, R. de Lemos, L. deCastro and R. Duncan, “Negative Selection: How to Generate Detectors”, 1st ICARIS, 2002.
12. D. Dasgupta, Z.Ji and F.Gonzalez, “Artificial Immune System Research in the last five years”. *In the proceedings of the international conference on Evolutionary Computation Conference (CEC), Canbara, Australia, December 8-12, 2003.*
13. Jungwon Kim and Peter Bentley, “Negative selection and Niching by an artificial immune system for network intrusion detection”, *In the proceeding of Genetic and Evolutionary Computation Conference (GECCO '99), Orlando, Florida, July 13-17.*
14. Riccardo Poli and William B. Langdon. *Schema theory for genetic programming with onepoint crossover and point mutation*. Evolutionary Computation, 6(3):231-252, 1998.
15. Stephanie Forrest, Brena Javornik, Robert E.Smith and Alan S.Perelson, “Using genetic algorithm to explore pattern recognition in the immune system”, *Evolutionary Computation, 1(3) (1993) 191-211*
16. Mitchell. A. Potter and Kenneth A.De. Jong, The Coevolution of Antibodies for concept Learning. In the proceeding of the *fifth international conference on parallel problem solving form nature*, September 1998,Amsterdam, The Netherlands.
17. M. A. Potter, K. A. De Jong, and J. J. Grefenstette. *A coevolutionary approach to learning sequential decision rules*. In Larry J. Eshelman, editor, Proceedings of the 6th International Conference on Genetic Algorithms (ICGA95), pages 366--372. Morgan Kaufmann Publishers, 1995.