Chapter 4

Artificial Immune Systems

The different theories in the science of immunology inspired the development (design) of immune inspired algorithms, collectively known as artificial immune systems (AIS), which are either based on or inspired by a specific theory on immunology or a combination of the different theories. The application areas of AIS cover a broad spectrum. AISs have been applied to different problem domains which among others include classification [30, 53, 63, 93, 179], anomaly and fraud detection [53, 60, 89, 90, 111, 121], optimisation [27, 35, 37, 55, 129], scheduling [57, 79, 130], data analysis and clustering [24, 32, 36, 133, 139, 165, 169, 187], and robotics [23, 104, 177]. This chapter discusses some of the most familiar AIS models and their applications. Since the proposed AIS model in this thesis is inspired by and mostly based on the network theory, a more detailed overview is given on existing network based AIS models within the context of data clustering.
The rest of the chapter is organised as follows:

- Section 4.1 defines a general AIS framework to highlight the basic components of an AIS model.
- Section 4.2 gives an introduction to the *shape space* model and how an artificial lymphocyte (ALC) and antigen pattern are presented in a shape space.
- Section 4.3 discusses the different measures of affinity between an ALC and antigen pattern within a specific shape space. The section also gives an overview of the different matching rules to determine whether an ALC binds to an antigen pattern.
- Section 4.4 gives an overview of AIS models which are inspired by the self-tolerant T-Cells in the natural immune system (classical view).

- Section 4.5 discusses some of the AIS models which are inspired by the clonal selection theory.
- Section 4.6 gives an introduction to some of the theoretical network based models and a detailed discussion on different network theory inspired AIS models.
- Section 4.7 discusses the different theoretical approaches to determine the possible interactions in an idiotypic network.
- Section 4.8 briefly highlights the difference between danger theory inspired AIS models and those AIS models which are inspired by the classical view of the natural immune system. The section briefly discusses some of the applications of the danger AIS models.
- Section 4.9 concludes the chapter by giving an overall summary of the chapter and comparing existing network based AIS models to the network AIS proposed in this thesis.

4.1 A Basic AIS Framework

This section defines a general AIS framework which is based upon the functional and organisational behaviour of the natural immune system (NIS) as discussed in chapter 3. In this section, the terms "cell" and "molecule" are used interchangeably. The capabilities of the NIS within each theory are summarised below:

- In some cases the NIS knows the structure of *self*/normal cells and *non-self*/*foreign* cells. In other cases the NIS only knows the structure of *self*/normal cells.
- In cases where the NIS knows the structure of *self* and *non-self* cells, the NIS is capable of recognising *non-self* associated cell structures (innate immune system).
- In cases where the NIS only knows the structure of *self* cells, the NIS needs to learn the structure of the *non-self* cells (adaptive immune system).
- A *foreign* cell is capable of causing *damage*.
- Lymphocytes are cloned and mutated to learn and adapt to the structure of the encountered *foreign* cells.
- The build-up of a *memory* on the learned structures of the *foreign* cells.

- A faster secondary response to frequently encountered *foreign* cells, due to the built-up *memory*.
- The co-operation and co-stimulation among lymphocytes to learn and react to encountered *foreign* cells can result into the formation of lymphocyte networks.
- The NIS consists of layers of defense against *foreign* cells.
- The entities within the different layers communicate in response to encountered *foreign* cells by means of signaling.

The above capabilities imply that it is the co-operation between different lymphocytes in different layers of the natural immune system which results in an active immune response to detected pathogenic material. To recapitulate the discussion on the different theories and layers of the NIS in chapter 3: Foreign cells are detected by macrophages in the first layer of defense, known as the innate immune system. If a foreign cell is not detected in the innate immune layer, mature T-Cells and B-Cells react to the encountered foreign cell in the adaptive immune system, i.e. second layer of defense. The response within the adaptive layer can be either primary or secondary.

In the primary response, B-Cells and T-Cells co-operate and co-stimulate each other in an attempt for the B-Cell to secrete antibodies with a higher affinity to the detected foreign cell. If foreign cells with a similar structure are frequently detected in the primary response, a memory of the structure is built-up in the NIS by the B-Cells that proliferate. In the secondary response, these memory cells can then have a faster reaction to the foreign cell with a similar structure, thus there is no need for a primary response to adapt to the structure of the foreign cell.

Within the primary response, B-Cells adapt to the structure of the foreign cell through the process of affinity maturation. This can result in B-Cells detecting each others structure to form an idiotypic network, co-stimulating or suppressing each other in response to an encountered foreign cell. Before a B-Cell can proliferate or undergo the affinity maturation process, the helper T-Cell needs to secrete lymphokines which either promote or suppress B-Cell growth. When a helper T-Cell receives a danger signal from the innate immune layer, indicating necrotic cell death, the T-Cell secretes lymphokines which promote B-Cell growth. This in turn proliferate the B-Cell. If a T-Cell is presented with a peptide pattern by a B-Cell without receiving a danger signal from the innate immune system, it implies that although the detected cell is foreign, it is harmless to the body and the T-Cell can secrete lymphokines to suppress the proliferation of the B-Cell.

The interaction of T-Cells, B-Cells and signaling of danger, occurs within the lymph nodes which are connected with lymph vessels. T-Cells and B-Cells *meet* each other in the lymph nodes to *exchange* antigenic information. Thus, to model an AIS, there are a few basic concepts that must be considered:

- There are trained detectors (artificial lymphocytes) that detect non-self patterns with a certain affinity.
- The artificial immune system may need a good repository of self patterns or self and non-self patterns to train the artificial lymphocytes (ALCs) to be self-tolerant.
- The affinity between an ALC and a pattern needs to be measured. The measured affinity indicates to what degree an ALC detects a pattern.
- To be able to measure affinity, the representation of the patterns and the ALCs need to have the same structure.
- The affinity between two ALCs needs to be measured. The measured affinity indicates to what degree an ALC links with another ALC to form a network.
- The artificial immune system has memory that is built-up by the artificial lymphocytes that frequently detect non-self patterns.
- When an ALC detects non-self patterns, it can be *cloned* and the clones can be mutated to have more diversity in the search space.

Using the above concepts as a guideline, the pseudo code in algorithm 4.1 is a template for the AIS algorithms considered in this thesis. Each of the algorithm's parts is briefly explained next.

1. **Initialising $\mathcal{B}$ and determining $\mathcal{A}$:** The population $\mathcal{B}$ can be populated either with randomly generated ALCs or with ALCs that are initialised with a cross section of the data set to be learned. If a cross section of the data set is used to initialise the ALCs, the complement of the data set will determine the training set $\mathcal{A}$. These and other initialisation methods are discussed for each of the AIS models in the sections to follow.
2. **Stopping condition for the *while*-loop**: In most of the discussed AIS models, the stopping condition is based on convergence of the ALC population or a preset number of iterations.

Algorithm 4.1: AIS Algorithm Template

```
Initialise a set of ALCs as population $\mathcal{B}$;
Determine the antigen patterns as training set $\mathcal{A}$;
while some stopping condition(s) not true do
    for each antigen pattern $\mathbf{a}_j \in \mathcal{A}$ do
        Select a subset of ALCs for exposure to $\mathbf{a}_j$, as population $\mathcal{S} \subseteq \mathcal{B}$;
        for each ALC, $\mathbf{b}_i \in \mathcal{S}$ do
            Calculate the antigen affinity between $\mathbf{a}_j$ and $\mathbf{b}_i$;
        end
        Select a subset of ALCs with the highest calculated antigen affinity as population
        $\mathcal{H} \subseteq \mathcal{S}$;
        Adapt the ALCs in $\mathcal{H}$ with some selection method, based on the calculated
        antigen affinity and/or the network affinity among ALCs in $\mathcal{H}$;
        Update the stimulation level of each ALC in $\mathcal{H}$;
    end
    Adapt the ALCs in $\mathcal{H}$ with a network selection method, based on the calculated
    network affinity among ALCs in $\mathcal{H}$ (optional);
end
```

3. **Selecting a subset, $\mathcal{S}$, of ALCs**: The selected subset $\mathcal{S}$ can be the entire set $\mathcal{B}$ or a number of randomly selected ALCs from $\mathcal{B}$. Selection of $\mathcal{S}$ can also be based on the stimulation level (as discussed below).

4. **Calculating the *antigen affinity***: The antigen affinity is the measurement of similarity or dissimilarity between an ALC and an antigen pattern. The most commonly used measures of affinity in existing AIS models are the Euclidean distance, r-contiguous matching rule, hamming distance and cosine similarity.

5. **Selecting a subset, $\mathcal{H}$, of ALCs**: In some of the AIS models, the selection of *highest affinity* ALCs is based on a preset affinity threshold. Thus, the selected subset $\mathcal{H}$ can be the entire set $\mathcal{S}$, depending on the preset affinity threshold.

6. **Calculating the *network affinity***: This is the measurement of *affinity* between two ALCs. The different measures of network affinity are the same as those for *antigen affinity*. A preset network affinity threshold determines whether two or more ALCs are linked to form a network.

7. **Adapting the ALCs in subset $\mathcal{H}$**: Adaptation of ALCs can be seen as the maturation process of the ALC, supervised or unsupervised. Some of the *selection* methods that can be

used are *negative selection* (or *positive selection*), *clonal selection* and/or some evolutionary technique with mutation operators. ALCs that form a *network* can influence each other to adapt to an antigen. These *selection* methods are discussed for each of the AIS models considered in this chapter.

8. **Updating the stimulation level of an ALC**: The stimulation level is calculated in different ways in existing AIS models. In some AIS models, the stimulation level is seen as the summation of antigen affinities. The stimulation level determines the resource level of an ALC. The stimulation level can also be used to determine a selection of ALCs as the *memory* set. The *memory* set contains the ALCs that most frequently match an antigen pattern, thus *memory* status is given to these ALCs. The stimulation level is discussed for the different AIS models in the chapter.

The above listed concepts to model a basic AIS can be grouped into the different layers of an AIS framework as proposed by De Castro and Timmis [34]. The proposed layered AIS framework consists of three main parts [34]:

1. **Representation**: Defining the structure to represent an antigen or receptor (ALC) in the problem domain (search space).

2. **Interaction**: Selecting an affinity function to quantify the *quality* of a structure as defined in the *representation* layer. Typically this function measures the degree of *similarity* or *dissimilarity* between an ALC's (receptor) structure and a structure representing another ALC or antigen.

3. **Adaptation**: Selecting a strategy (immune process or theory) to guide the behaviour of the model.

In the above definition of a layered AIS framework, the structures within the *representation* layer forms part of the input to the *interaction* layer for affinity calculations (quality). In turn, the calculated quality within the *interaction* layer forms part of the input to the *adaptation* layer. Therefore the selected strategy within the *adaptation* layer guides the behaviour of the model based on the calculated affinities. The sections to follow discuss the different existing AIS models within the context of the above layered AIS framework.

4.2 Representation of Antigens and Antibodies

An antigen or receptor can be represented as a string of attributes in the *search space*. Perelson and Oster defined the notion of a *shape space* which is similar to a search space [147, 149]. Assume the shape space ϖ is a bounded region of $\mathcal{R}^N$ with volume V. Each receptor's binding site can be characterised by measuring a number of features. The grouping of these features is known as the receptor's *generalised shape* [147]. If the generalised shape of a receptor is described by a *feature vector* with N features, the generalised shape of the receptor can be represented as a point in N-dimensional space, and therefore an N-dimensional ALC can be represented in shape space ϖ.

Epitopes on the surface of an antigen also have a generalised shape and can be represented in the shape space ϖ. Assume receptor **b** has an exact complementary match to epitope **a**, then the binding affinity between the receptor and epitope is at its highest level. A less exact match results in a lower binding affinity. Thus, the generalised shape of a receptor can *match* more than one epitope with different binding affinity levels [147]. The epitope in ϖ with an exact complement of the receptor's generalised shape represents a region of epitopes with different levels of binding affinities with the receptor (ALC). The region is known as a *recognition region* and all epitopes within this region's radius have an affinity higher than a certain threshold. The radius of the *recognition region* is determined by the affinity threshold. A lower threshold results in a higher radius, which implies a larger region with a less specific binding between an epitope and receptor (ALC).

Figure 4.1 illustrates the shape space ϖ with an ALC **b** and a *recognition region* for a complementary match with antigen **a**. The average volume of the region covered by a *recognition region* in shape space ϖ is defined as $v_{\phi(r)}$ where ϕ is the radius function of the affinity threshold r [149]. From the above definition of a *shape space* with different *recognition regions* for each of the receptors, Perelson and Oster concluded that an infinite number of epitopes (antigen) are recognised by a finite collection or repertoire of receptors (ALCs) [147]. Thus, the initialisation of a repertoire with p random receptors covers a total volume of $p \times v_{\phi(r)}$. With $p \times v_{\phi(r)} > V$, the volume V of *shape space* ϖ is completely covered by the repertoire of ALCs with some overlap between the different *recognition regions* [147, 149].

Measuring the affinity between an ALC and an antigen pattern as a complementary match in-

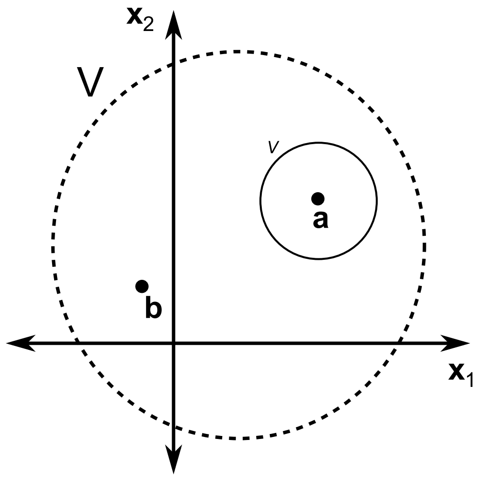


Figure 4.1 Shape space ϖ as bounded region $\mathcal{R}^2$ with volume V

dicates the *dissimilarity* between an ALC and an antigen pattern. The *similarity* between an ALC and antigen pattern can also be measured as the affinity. Thus, measuring the affinity as *similarity*, each ALC in shape space can be presented as a *recognition region* with a certain radius, i.e. affinity threshold. All antigen patterns within the area of an ALC's *recognition region* have a certain measured degree of *similarity* which adheres to the affinity threshold of the ALC.

There are different definitions and approaches to measure the *interaction* between an ALC and antigen pattern or between ALCs, i.e. affinity measurement. Different (*dis*)*similarity* measures for calculating the degree of affinity between an ALC and an antigen/ALC pattern have been proposed. The most commonly used measures of affinity are discussed in the next section.

4.3 Affinity as Quality Measure

Referring to section 3.2, an antibody/receptor binds to an antigen with a certain binding strength known as the *affinity*. The process of affinity maturation (refer to section 3.5), which consists of somatic hyper mutation and clonal selection, improves the affinity of an antibody with the detected antigen. Thus, measuring the affinity between an ALC and an antigen pattern or another ALC gives an indication of the quality (or fitness) of the ALC to *match* an antigen pattern. The adaptation of ALCs to learn the structure of the antigen patterns is guided by measuring the quality of the ALCs.

Affinity in AISs is measured as the spatial distance between an ALC and an antigen pattern or another ALC. The affinity between an ALC and an antigen pattern could be measured against an affinity threshold to determine whether the ALC *matches* the antigen pattern, i.e. whether the antigen pattern is within the radius of the ALC's *recognition region* (as defined in the shape space theory, discussed in section 4.2). Therefore, there are different matching rules based on affinity measurement and thresholding for different shape spaces (problem domains). This section discusses some of the most common matching rules proposed for nominal shape spaces. The *generalised shape* of an ALC or antigen pattern in a nominal shape space consists of features (attributes) which are nominal categories. The different categories for a nominal shape space are known as the *alphabet* of the shape space.

The different distance-based (*dis*)*similarity* measures between feature vectors in continuous shape space were discussed in section 2.2. The distance between two feature vectors in continuous shape space is also measured against an affinity threshold to determine whether the two feature vectors *match*. The most commonly used (*dis*)*similarity* measure in AISs, applied to continuous shape space problems, is the Euclidean distance (as defined in equation (2.3)).

4.3.1 A Complementary Matching Rule

The Hamming distance measures the dissimilarity between two feature vectors in nominal shape space. The Hamming distance between two feature vectors, $\mathbf{a}$ and $\mathbf{b}$, counts the number of positions (features) which are different, as defined in equation (2.10). Therefore, a shape space alphabet of $\{0,1\}$, has feature vectors in binary space and the Hamming distance between these binary vectors counts the number of exclusive-or bits between the corresponding positions (defined as $\sigma(\mathbf{a},\mathbf{b})$ in equation (2.11)). The binary immune system was introduced by Farmer *et al.* [49]. A pattern has a complementary *match* with another pattern if the calculated Hamming distance is greater or equal to an affinity threshold, r, i.e.

$$\sigma(\mathbf{a},\mathbf{b}) \geq r \tag{4.1}$$

Thus, the affinity threshold, r, indicates the least number of differing positions for a pattern to *match* another pattern under the Hamming distance based matching rule. The reader is referred to [93] for an overview of Hamming distance based matching rules.

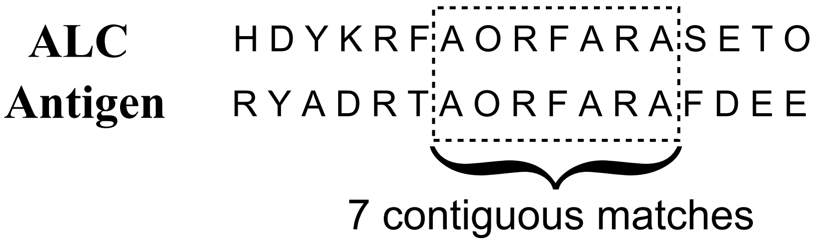

ALC
Antigen
H D Y K R F A O R F A R A S E T O
R Y A D R T A O R F A R A F D E E
7 contiguous matches

Figure 4.2 *r*-contiguous matching rule

4.3.2 The *r*-contiguous Matching Rule

The *r*-contiguous matching rule was proposed by Percus *et al.* [146]. Figure 4.2 illustrates the r-contiguous matching rule between two patterns, **a** and **b**. The r-contiguous matching rule is a partial matching rule. This means that a pattern matches another pattern if there are *r*-contiguous or more matches in the corresponding positions. *r* is the degree of affinity for a pattern to *match* another pattern. In figure 4.2 there are seven contiguous matches between the two patterns. Thus, if $r = 4$, the two patterns match each other in figure 4.2, since $7 > r$. If $r > 7$, there is no match between the patterns in the figure. A higher value of *r* indicates a stronger affinity between two patterns. As illustrated in figure 4.2, the *r*-contiguous matching rule can be seen as a window of width *r*, sliding from the left to the right over two patterns, searching for an *exact* match in the window. The *r*-contiguous matching rule is applied in [53, 183].

4.3.3 The *r*-chunks Matching Rule

The *r*-chunks matching rule is a variation of the above discussed *r*-contiguous matching rule, introduced by Balthrop *et al.* [11]. This matching rule is also known as the *r-contiguous templates* matching rule. The difference between *r*-chunks and the *r*-contiguous matching rule is that *r*-chunks generates template windows of size *r* from pattern **a**. Each template window consists of r-contiguous positions in **a**. A pattern, **b**, is matched by **a** if one of the template windows has an *exact* match by *r*-contiguous positions in **b**. The number of template windows of size *r*, generated from a pattern of size *N* is equal to $(N - r + 1)$ [11]. For example, a pattern, $\mathbf{a} = \langle 100011 \rangle$, in binary space of length 6, can be partitioned into 4 template windows of size 3. The template windows of **a** are $\langle 100 \rangle$, $\langle 000 \rangle$, $\langle 001 \rangle$ and $\langle 011 \rangle$ ($r = 3$).

Compared to the *r*-contiguous matching rule, instead of sliding a window of size *r* over two patterns to find an exact match within the window, *r*-chunks slides pattern **b** as a window over pattern **a** to align/match r-contiguous positions between the patterns. Thus, the *r*-chunks matching rule generates detectors of length N, consisting of a template window of size r, starting at position i in the detector. An r-chunk detector, $\mathbf{x}$, generated from template window $\langle 001 \rangle$ in the above example with $i = 3$ gives $\mathbf{x} = \langle \#\#001\# \rangle$ where # is a 'no care' symbol. The detector $\mathbf{x}$ matches pattern $\mathbf{a}$. If $i = 1$, then $\mathbf{x} = \langle 001\#\#\# \rangle$ and does not match pattern $\mathbf{a}$.

All of the above matching rules have mostly been used by *negative selection* based AIS models. ALCs trained with *negative selection* (discussed in section 4.4) are self-tolerant. When the set of self patterns, Υ, does not contain all patterns of self during the *censoring* process, the set of self-tolerant ALCs, $\mathcal{B}$, represents a generalised structure which results in some patterns being unmatched by the self-tolerant ALCs [11]. These unmatched patterns are known as *holes* and occur when the above matching rules are applied [42]. Balthrop *et al.* identified and defined two types of *holes*, namely *length-limited holes* and *crossover holes* [11].

Length-limited holes: Length-limited holes occur when applying the r-contiguous matching rule [11]. A length-limited hole, $\mathbf{x}^*$, is a pattern with at least one window of size r that does not exist among the distinct template windows in a set of patterns, Υ, and for which a detector cannot be generated [11]. Let $\Upsilon = \{\langle 0010 \rangle, \langle 1000 \rangle, \langle 0100 \rangle, \langle 1100 \rangle\}$, $N = 4$, $r = 3$ and $\mathbf{h} = \langle 0101 \rangle$. There are two template windows for $\mathbf{x}^*$. These template windows are $\langle 010 \rangle$ and $\langle 101 \rangle$. Therefore a detector starts with template window $\langle 010 \rangle$ and/or ends with template window $\langle 101 \rangle$. A detector that starts with template window $\langle 010 \rangle$ matches patterns in Υ and can therefore not be generated. A detector that ends with template window $\langle 101 \rangle$ can either represent pattern $\langle 0101 \rangle$ or pattern $\langle 1101 \rangle$, which both match patterns in Υ. Therefore detector $\mathbf{x}^*$ cannot be generated.

Crossover holes: Crossover holes occur when applying the r-chunks matching rule [11]. Two template windows are adjacent if the last position of the first template window is the first position of the second template window. A crossover hole, $\mathbf{x}^*$, is a pattern which is not part of a set, Υ, but the template windows of $\mathbf{x}^*$ are adjacent to the distinct template windows of the patterns in Υ [11]. Let $W_i^{\mathbf{x}} = (x_i, x_{i+1}, \ldots, x_{i+r-1})$ be the template window of pattern $\mathbf{x}$ starting at position i in $\mathbf{x}$. A crossover hole occurs between template window $W_i^{\mathbf{x}^*}$ of pattern $\mathbf{x}^*$ and a template window $W_{i+1}^{\mathbf{a}}$ of pattern $\mathbf{a} \in \Upsilon$ if $x_j^* = a_j, \forall j : i+1 \leq j \leq i+r-1$ [11].

4.4 Negative Selection Models

One of the main features in the classical view of the natural immune system is the mature T-Cells, which are self-tolerant, i.e. mature T-Cells have the ability to distinguish between *self* cells and foreign/*non-self* cells. The original negative selection algorithm proposed by Forrest *et al.* [53] is inspired by the maturation process of immature T-Cells in the thymus.

In the original model of Forrest *et al.* [53], all patterns and ALCs are represented as strings with a fixed length, N. The attributes of each string can have any value which is selected from a pre-defined alphabet with size κ. For example, each attribute of a binary string can only have a value of 0 or 1 since the valid values in the binary alphabet is defined as $\{0,1\}$, therefore $\kappa = 2$. A string generated from an alphabet defined as $\{G,A,T,C\}$ can only have combinations of $\{G,A,T,C\}$ attribute-values. The number of strings with length N that can be generated from an alphabet with size κ is N^κ.

A set of trained ALCs in the model represents the mature T-Cells in the natural immune system. A training set of self patterns is used to train the set of ALCs with the *negative selection* technique. Algorithm 4.2 lists the pseudo code for negative selection, explained in detail below.

For each randomly generated candidate ALC of length N, the affinity between the ALC and each self pattern (also of length N) in the training set is calculated. The affinity between an ALC and a pattern is measured with the r-contiguous matching rule. If the affinity between any self pattern and an ALC is higher than the affinity threshold, r, the candidate ALC is discarded and a new candidate ALC is randomly generated. The new ALC also needs to be measured against the training set of self patterns. If the affinity between all the self patterns and a candidate ALC is lower than the affinity threshold, r, the ALC is added to the self-tolerant set of ALCs. Thus, the set of ALCs is negatively selected, which means that only those ALCs with a calculated affinity less than the affinity threshold, r, will be included in the set of self-tolerant ALCs. This phase is known as *censoring*.

Algorithm 4.2: Training ALCs with *negative selection*

```
Set counter $b$ as the number of self-tolerant ALCs to train;
Create empty set of self-tolerant ALCs as $\mathcal{B}$;
Determine the training set of self patterns as $\Upsilon$;
while size of $\mathcal{B}$ not equal to $b$ do
    Randomly generate a candidate ALC, $\mathbf{x}$;
    matched=false;
    for each self pattern $\mathbf{s} \in \Upsilon$ do
        if affinity between $\mathbf{x}$ and $\mathbf{s}$ is higher than affinity threshold $r$ then
            matched=true;
            break;
        end
    end
    if not matched then
        Add $\mathbf{x}$ to set $\mathcal{B}$;
    end
end
```

The trained, self-tolerant set of ALCs is then presented with a test set of self and non-self patterns for classification. This phase is known as *monitoring*. The affinity between each training pattern and the set of self-tolerant ALCs is calculated. If the calculated affinity is below the affinity threshold, r, the pattern is classified as a self pattern; otherwise the pattern is classified as a non-self pattern. The training set is monitored by continually testing the ALC set against the training set for changes. A number of drawbacks of the proposed negative selection model are that,

- the training set needs to have a good representation of self patterns,
- an increase in the number of self patterns exponentially increases the number of randomly generated candidate ALCs [111],
- there is an exhaustive replacement of an ALC during the censoring of the training set until the randomly generated ALC is self-tolerant, and
- there is no validation/removal of redundant ALCs.

The above listed drawbacks were also highlighted by Ayara *et al.* [7]. Alternative *censoring* approaches to generate self-tolerant ALCs have been proposed to address some of the above drawbacks of the original model. These alternative models are briefly discussed next.

Linear model: The first of these alternative *censoring* models is the *linear* model proposed by D'haeseleer *et al.* [42]. The linear model runs in linear time with respect to the size of the self set, given that the string length, N, and the matching affinity threshold, r, are fixed. Binary strings are generated from a binary alphabet $\{0,1\}$. The model generates different matching templates to determine the number of unmatched strings in the self set. A template is a string of length N, where r contiguous positions of the template are set to a value. The remaining $N-r$ positions are set to 'no care' symbols. A set of self-tolerant ALCs are then randomly selected from the unmatched templates.

Greedy model: D'haeseleer *et al.* also proposed the *greedy* model [42]. The difference between the *greedy* model and the previously discussed *linear* model is that ALCs in the self-tolerant set, $\mathcal{B}$, are not randomly selected from the set of unmatched template strings. The selected set of self-tolerant ALCs have minimal overlap among each other and maximum coverage of the non-self space.

Binary tree template and the discriminative power: The templates which are used in the *greedy* model to generate self-tolerant ALCs can be assembled to form different binary trees. This results in the formation of general subtrees (templates) which reduces the number of self-tolerant ALCs to cover most of the patterns in non-self space. The formation of binary trees by these templates was observed and proposed by Wierchon, i.e. binary tree templates [182]. As discussed in section 4.3, compared to the hamming distance, the r-contiguous matching rule is symmetric and reflexive. Thus, Wierchon investigated the discriminative power of a candidate ALC containing a template which is matched by the r-contiguous matching rule [183]. The discriminative power of a candidate ALC is defined as the number of unique strings matched by the ALC using the r-contiguous matching rule [183].

NSMutation: This model differs in the *censoring* process of candidate ALCs by not immediately discarding a candidate ALC when matched with a certain affinity to a self pattern. Instead, guided mutation is performed on the self-matching candidate ALC pattern, away from the matched self pattern. This model was proposed by De Castro and Timmis [34] and is inspired by the process of affinity maturation in the natural immune system. Since the *r*-contiguous matching rule is applied, the candidate ALC pattern is only mutated in the *r* matching positions. The probability of mutation is proportional to the affinity with the self pattern. Thus a higher affinity between the self-matching ALC and the self pattern results in a higher rate of mutation and vice versa.

Mutation is performed on a self-matching candidate ALC pattern for only a number of times, the lifetime of a candidate ALC. The mutated candidate ALC is discarded if the specified lifetime is reached with no improvement. If a mutated candidate ALC does not match any self pattern, the mutated ALC is then added to the set of self-tolerant ALCs.

Evolutionary approaches: A different approach is proposed by Kim and Bentley [110] where candidate ALCs are not randomly generated and tested with negative selection, but an evolutionary process is used to evolve ALCs towards non-self and to maintain diversity and generality among the ALCs. The model by Potter and De Jong [150] applies a co-evolutionary genetic algorithm to evolve ALCs towards the selected class of non-self patterns in the training set and further away from the selected class of self patterns. Once the fitness of the ALC set evolves to a point where all the non-self patterns and none of the self patterns are detected, the ALCs represent a description of the concept. If the training set of self and non-self patterns is noisy, the ALC set will be evolved until most of the non-self patterns are detected and as few as possible self patterns are detected. The evolved ALCs can discriminate between examples and counter-examples of a given concept. Each class of patterns in the training set is selected in turn as self and all other classes as non-self to evolve the different concepts in the training set.

Gonzalez *et al.* [60] present a negative selection method which is able to train ALCs with continuously-valued self patterns. The ALCs are evolved away from the training set of self patterns and are well separated from one another to maximise the coverage of non-self. This results in the least possible overlap among the evolved set of ALCs. A similar approach is presented in the GAIS model of Graaff and Engelbrecht [63]. All patterns are represented as binary strings and the Hamming distance is used as affinity measure. A genetic algorithm is used to evolve ALCs away from the training set of self patterns towards a maximum non-self space coverage and a minimum overlap among existing ALCs in the set. The difference to the model of Gonzalez *et al.* [60] and the original negative selection model of Forrest *et al.* [53] is that each ALC in the set has a local affinity threshold. The ALCs are trained with an adapted negative selection method as illustrated in figure 4.3.

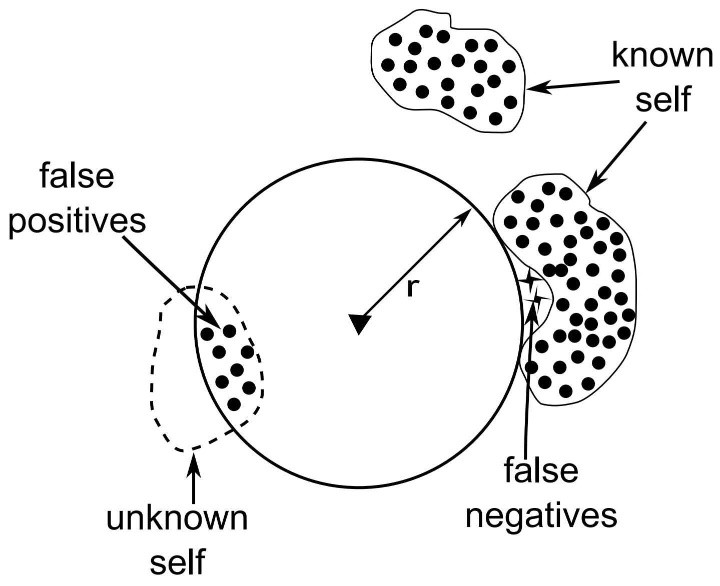


Figure 4.3 Adapted Negative Selection

With the adapted negative selection method the affinity threshold, *r*, of an ALC is determined by the distance to the closest self pattern from the ALC. The affinity threshold, *r*, is used to determine a match with a non-self pattern. Thus, if the measured affinity between a pattern and an ALC is less than the ALC's affinity threshold, *r*, the pattern is classified as a non-self pattern. The adaptive negative selection method is inspired by the definition of epitope-volumes in *shape space* (as discussed in section 4.2). Figure 4.3 also illustrates the drawback of *false positives* and *false negatives* when the ALCs are trained with the adapted negative selection method. These drawbacks are due to an incomplete static self set. The *known self* is the incomplete static self set that is used to train the ALCs and the *unknown self* is the self patterns that are not known during training. The *unknown self* can also represent self patterns which are outliers to the set of *known self* patterns.

Surely all evolved ALCs will cover non-self space, but not all ALCs will detect non-self patterns. Therefore, Graaff and Engelbrecht [62, 63] proposed a transition function, the life counter function, to determine an ALC's status. ALCs with annihilated status are removed in an attempt to have only mature and memory ALCs with optimum classification of non-self patterns.

The V-detector model proposed by Ji and Dasgupta [99] as an alternative to the model of Gonzalez *et al.* [60] for continuously valued patterns in Euclidean space is similar to the model of

Graaff and Engelbrecht [61, 63], in that each generated ALC has a local affinity threshold which is determined by calculating the distance between the generated ALC and the closest self pattern. The V-detector model generates ALCs of variable length in continuous space compared to the fixed length ALCs in Hamming space of the proposed model by Graaff and Engelbrecht [63].

Smith *et al.* derived analogies between the memory capabilities of the immune system and the method of sparse distributed memory (SDM) [161]. SDM was proposed by Kanerva as a method to store a large number of large binary patterns using a small number of physical data addresses [105]. These physical addresses are known as physical or hard locations and binary patterns are stored in these locations in such a manner that any data can be accurately recalled. An SDM is composed of a set of these hard locations. Each location has a recognition radius. A location recognises data if the distance to the data is within the recognition radius of the location.

In addition, each location also has a set of counters, each representing a bit in the location. The counters are used to determine whether a recalled bit from the memory should be set to '1' or '0'. A data pattern is distributed to all locations which recognise it. If a location recognises a data pattern, the counter of each bit is incremented by '1' if the data pattern's corresponding bit is '1' and decremented by '1' if the data pattern's corresponding bit is '0'. When a pattern needs to be recalled from the memory, the counters of locations recognising the pattern are summed. The corresponding bit of the recalled pattern is set to '1' if the summed counters of that bit is greater than or equal to zero; otherwise the bit is set to '0'.

Inspired by the analogies between SDM and the immune system, Hart *et al.* proposed a co-evolutionary SDM (COSDM) model to cluster non-stationary data [77]. In COSDM an antigen represents the data pattern that needs to be stored and an ALC represents a hard location. The recognition radius of an ALC determines the size of a cluster. COSDM consists of a number of ALC populations. A co-evolutionary genetic algorithm was used to find the set of ALCs as well as the size of their individual recognition radii, which best clustered the current available data. Some of the drawbacks of COSDM were that the algorithm was relatively slow and had difficulty to set the correct recognition radius for each ALC [78].

Hart *et al.* proposed the self-organising SDM (SOSDM) to address the drawbacks of COSDM [76, 78]. Hart *et al.* highlighted the unsuitability of Kanerva's SDM to cluster data in [77] and based the SOSDM on an alternative SDM model as proposed by Hely *et al.* [82]. In SOSDM, an

ALC binds to an antigen pattern if the binding strength between the ALC and the antigen pattern is greater than a specified threshold. The binding strength is proportional to the calculated affinity between an ALC and an antigen pattern. The affinity is measured as the Hamming distance between an ALC and an antigen pattern. The binding strength of each ALC is calculated as the ratio of the calculated affinity to the maximum affinity in the set of ALCs. Therefore, the binding strength of an ALC with maximum affinity to an antigen pattern is one. Each ALC's set of counters is updated with the binding strength to an antigen pattern, where the binding strength is proportional to the calculated affinity.

In addition, each ALC also measures an accumulated error between an ALC (hard location) and all antigens presented to the ALC. The accumulated error of an ALC is updated with each antigen pattern that binds to the ALC. After all antigen patterns have been presented to the set of ALCs, the average error of each ALC is used to self organise the set of ALCs. This is done in such a manner that ALCs move towards positions in the search space, such that the average error is minimised. An ALC's associated set of counters decays over time. The results obtained by SOSDM on clustering stationary data are scalable with the size of the data set and with the length of the antigen pattern [78]. Clustering of non-stationary data delivered promising results, but highlighted a decrease in performance with more dynamic data [78]. SOSDM is an adaptive, scalable and self-organising model.

4.5 Clonal Selection Models

The natural immune system is able to adapt to unseen antigens and capable of keeping a memory of frequently encountered antigens. This is achieved by a process of affinity maturation which consists of clonal selection with somatic hyper mutation (as discussed in section 3.5). The former is the process of selecting the most stimulated (highest affinity) lymphocytes for clonal proliferation, and the latter the mutation process on these clones. The increase in size of some clones results in a decrease in size of other previously cloned lymphocytes, since the natural immune system regulates the total number of lymphocytes in the body. Clones are mutated in an attempt to have a higher affinity with the encountered antigen.

Frequently selected lymphocytes (through clonal selection) transition into a state of memory and these memory lymphocytes are used in a faster secondary response to frequently encountered antigens (as discussed in section 3.5). Those lymphocytes which are seldom selected (or never stimulated) transition into a state of annihilation and is eventually replaced by the natural immune system with newly generated lymphocytes. Through the process of affinity maturation the natural immune system is able to learn new antigens, keep a memory of frequently encountered antigens, and integrate newly generated lymphocytes. The learning capability of the natural immune system inspired the modelling of clonal selection with somatic hyper mutation in AISs.

Clonal selection in AISs is the selection of a set of ALCs with the highest calculated affinity with an antigen pattern. The selected ALCs are then cloned and mutated in an attempt to have a higher binding affinity with the presented antigen pattern. The mutated clones compete with the existing set of ALCs, based on the calculated affinity between the mutated clones and the antigen pattern, for survival to be exposed to the next antigen pattern. This section discusses some of the AIS models inspired by the clonal selection theory.

CLONALG: The CLONALG model is a general implementation of the clonal selection theory and was introduced by De Castro and Von Zuben as an algorithm that can perform machine-learning and pattern recognition tasks [35, 38]. ALCs and antigen patterns are presented as binary strings and therefore the affinity between an ALC and an antigen pattern is measured with the Hamming distance. A lower Hamming distance implies a higher affinity.

CLONALG evolves a population of randomly initialised ALCs over a number of generations to have a higher affinity with the presented antigen patterns. The population of ALCs is partitioned into a subset of memory ALCs and a remaining subset of ALCs (non-memory ALCs). CLONALG assumes that there is an ALC in the memory subset for each antigen pattern that needs to be recognised.

In a generation, each of the antigen patterns is presented to the population of ALCs. A number of highly stimulated ALCs (those with highest affinity with the presented antigen) are then selected for cloning. The number of clones generated for an ALC is directly proportional to the calculated affinity and is calculated as [38]

$$\eta(\mathbf{b}_i) = round\left(\frac{\Theta \times |\mathcal{B}|}{i}\right) \tag{4.2}$$

where Θ is a multiplying factor, *round* is a function that rounds a floating-point value to the closest integer, and i is the position of the ALC in the sorted set of highly stimulated ALCs (sorted in ascending order of affinity). Each of the clones is then mutated at a rate which is inversely proportional to the affinity. This means that clones from highly stimulated ALCs are mutated less than clones from less stimulated ALCs. The mutated clone with the highest affinity with the presented antigen replaces the ALC in the memory subset of the population if its corresponding memory ALC has a lower measured affinity. A percentage of the ALC population with the lowest affinities is replaced by randomly generated ALCs.

A modified version of CLONALG has been applied to multi-modal function optimisation [35]. In the optimisation model the entire population of ALCs is seen as the memory set (no subset of memory ALCs). All the ALCs are cloned with equal size, changing equation (4.2) to [38]

$$\eta\left(\mathbf{b}_i\right) = round\left(\Theta \times |\mathcal{B}|\right) \tag{4.3}$$

The affinity of an ALC is calculated as the objective function that needs to be optimised, since there are no antigen patterns to present to the population. The ALCs with the highest affinity is selected as the population for the next generation. The population of ALCs for the next generation is selected from the population of the previous generation and the mutated clones of ALCs.

DynamiCS: In some cases the problem that needs to be optimised consists of self patterns that change through time. To address these types of problems, the dynamic clonal selection algorithm (DCS) was introduced by Kim and Bentley [114]. The dynamic clonal selection algorithm is based on the AIS proposed by Hofmeyr [85]. The basic concept in [85] is to have three different populations of ALCs, categorised into immature, mature and memory ALC populations.

Kim and Bentley explored the effect of three parameters on the adaptability of the model to changing self [114]. These parameters were the tolerisation period, the activation threshold and the life span. The tolerisation period is a threshold on the number of generations during which ALCs can become self-tolerant. The activation threshold is used as a measure to determine if a mature ALC met the minimum number of antigen matches to be able to become a memory ALC. The life span parameter indicates the maximum number of generations that a mature ALC is allowed to be in the system.

If the mature ALC's life span meets the pre-determined life span parameter value, the mature ALC is deleted from the system. Experimental results with different parameter settings indicated that an increase in the life span with a decrease in the activation threshold resulted in the model to have an increase in detecting true non-self patterns. An increase in the tolerisation period resulted in less self patterns being detected falsely as non-self patterns, only if the self patterns were stable.

With a changing self the increase in tolerisation period had no remarkable influence in the false detection of self patterns as non-self patterns. Although the DCS could incrementally learn the structure of self and non-self patterns, it lacked the ability to learn any changes in unseen self patterns. The memory ALCs in the DCS algorithm had infinite lifespan. This feature was omitted in the extended DCS by removing memory ALCs which were not self-tolerant to newly introduced self patterns [112].

DCS was further extended by introducing *hyper mutation* on the deleted memory ALCs [113]. The deleted memory ALCs were mutated to seed the immature detector population, i.e. deleted memory ALCs form part of a gene library. Since these deleted memory ALCs contain information (which was responsible for giving them memory status), applying mutation on these ALCs will retain and fine tune the system, i.e. reinforcing the algorithm with previously trained ALCs.

MARIA: A different model, though similar to the above DCS model with regards to the three populations used, is the MARIA model proposed by Knight and Timmis [116]. The model consists of multiple layers and addresses some of the shortfalls of the AINE model [169] (discussed in section 4.6). The defined layers interact to adapt and learn the structure of the presented antigen patterns. The model consists of three layers which fulfills different roles in the adaptation process.

All patterns in the training set are seen as antigens. The affinity between an antigen pattern, $\mathbf{a}_j$, and a cell within a layer is measured using the Euclidean distance, σ. The different layers in sequential order are: the free antibody layer ($\mathcal{F}$), the B-Cell layer ($\mathcal{B}$) and the memory layer ($\mathcal{M}$). Each layer has an affinity threshold and a death threshold. The affinity threshold determines whether an antigen binds to a cell within a specific layer. The death threshold is the maximum elapsed time for a cell not to be stimulated. This means that if the length of time since a cell was last stimulated exceeds the death threshold of the specific layer, the cell dies and is removed from the population in the specific layer. The B-Cell layer has an additional stimulation threshold which determines whether a cell in the specific layer is cloned.

Each antigen pattern is first presented to a random selection of cells in $\mathcal{F}$. The number of free antibodies that bind to the antigen pattern is calculated as $free\_binding$. The antigen, $\mathbf{a}_j$, is then randomly presented to the B-Cells in $\mathcal{B}$ until one of the B-Cells, $\mathbf{b}_i$, binds to the antigen pattern. $\mathbf{b}_i$ is cloned as $\mathbf{b}_i^*$ if the calculated $free\_binding$ in $\mathcal{F}$ exceeds the stimulation threshold. The clone $\mathbf{b}_i^*$ is then mutated as $\mathbf{b}_i^{'}$ and added to $\mathcal{B}$. Mutated clones of $\mathbf{b}_i$ are then added to $\mathcal{F}$. The number of mutated clones (or rather free antibodies) produced by a stimulated B-Cell is given in [116] as

$$\eta\left(\mathbf{a}_j,\mathbf{b}_i\right) = \left(\sigma_{max} - \sigma\left(\mathbf{a}_j,\mathbf{b}_i\right)\right) \times k \tag{4.4}$$

where η is the number of antibodies that are added to the free-antibody layer, σ_{max} is the maximum possible Euclidean distance between a B-Cell and an antigen pattern in the data space (i.e. lowest possible affinity), and k is some constant.

If none of the B-Cells in $\mathcal{B}$ bind to the antigen pattern, a new B-Cell is created with the same presentation as the unbinded antigen. The new B-Cell is added to $\mathcal{B}$ resulting in a more diverse coverage of antigen data. The new B-Cell also produces mutated clones as free antibodies, which are added to the free-antibody layer.

The final layer, $\mathcal{M}$, only consists of memory cells and only responds to new memory cells. The generated clone, $\mathbf{b}_i^*$, in $\mathcal{B}$ is presented as a new memory cell to $\mathcal{M}$. The memory cell with the lowest affinity to $\mathbf{b}_i^*$ is selected as $\mathbf{m}_{min}$. $\mathbf{m}_{min}$ is replaced by $\mathbf{b}_i^*$ if the affinity between $\mathbf{b}_i^*$ and $\mathbf{m}_{min}$ is lower than the affinity threshold of the specific layer, and the affinity of $\mathbf{b}_i^*$ is less than the affinity of $\mathbf{m}_{min}$ with the antigen that was responsible for the creation of the new memory cell. If the affinity between $\mathbf{b}_i^*$ and $\mathbf{m}_{min}$ is higher than the affinity threshold, $\mathbf{b}_i^*$ is added to the memory layer. The multi-layered model, compared to the SSAIS model [140] (discussed in section 4.6), obtained better compression on data while forming stable clusters.

AIRS: A supervised learning AIS algorithm, the Artificial Immune Recognition System (AIRS) [178, 179] borrowed the concept of an artificial recognition ball (ARB) population within a resource limited environment as proposed by the network based resource limited AIS (AINE) [169]. Contrary to AINE and other unsupervised network based AIS algorithms (as discussed in section 4.6), AIRS does not model any network interactions between ARBs. Furthermore, most unsupervised network based AIS models are applied to the problem of data clustering whereas AIRS is an AIS classifier.

The ARBs in AIRS also compete for resources to survive. The ARBs undergo a clonal expansion and maturation process to evolve a set of memory ARBs which represents the different classes of the training patterns (antigens). The evolved set of memory ARBs is used to classify unseen patterns into multiple classes. The definition of an ARB and the concept of a resource limited environment are discussed in more detail in the AINE paragraph in the next section.

4.6 Idiotypic Network Models

A number of different theoretical network based models have been proposed by immunologists to formulate and capture the characteristics and interactions of the natural immune network system. One of these theoretical network based models was proposed by Farmer *et al.* [49]. Farmer *et al.* exploited the fundamental concepts of the network theory as proposed by Jerne [97] and proposed a simple model to simulate the dynamics of the natural immune network system and its memory capability [49]. Perelson also proposed a model to simulate the dynamics and production of a network based immune network system [148]. The theoretical model proposed by Farmer *et al.* is discussed next, since the earliest work in artificial immune systems are based on this model. The rest of the section discusses different network theory inspired AIS models.

The theory of clonal selection as part of the process of affinity maturation assumes that all immune responses are activated by encountered antigens. As explained in section 3.5, antigens *select* those lymphocytes with which the antigens have the highest binding affinity, resulting in clonal proliferation and somatic hyper mutation of the selected lymphocytes. As a result of somatic hyper mutation on the clones, the variable regions of the clones can become antigenic and invoke an immune response from neighbouring lymphocytes (as discussed in section 3.6). The recognition of idiotopes results in interconnected neighbouring lymphocytes, forming an idiotypic network.

Thus, lymphocytes in a network co-stimulate and/or co-suppress each other in reaction to an antigen. Therefore a lymphocyte is not only stimulated by an antigen, but also by neighbouring lymphocytes (as discussed in section 3.6). This results in the annihilation of some lymphocytes and the introduction of mutated lymphocyte clones into the population of lymphocytes. Highly stimulated lymphocytes remain part of the population whereas less stimulated lymphocytes are replaced/removed from the population. The population of lymphocytes is dynamic in such a way that the concentration of antibodies/lymphocytes at different points in time differ. In order to formulate the change in concentration of the population, based on the stimulation of the lymphocytes, Farmer *et al.* [49] identified three factors which influence the stimulation of a lymphocyte. These are:

- the binding affinity with the encountered antigen,
- the stimulation received from neighbouring lymphocytes, and
- the suppression received from neighbouring lymphocytes.

The model of Farmer *et al.* defines a lymphocyte, $\mathbf{b}$, as two binary strings which represent the lymphocyte's epitope, $\mathbf{e}$, and paratope, $\mathbf{p}$. The change in concentration, ν, of a lymphocyte, $\mathbf{b}_i$, relative to time, t, is simulated with the following differential equation as proposed in [49]:

$$\frac{d\nu(\mathbf{b}_i)}{dt} = c\left[\sum_{j=1}^{|\mathcal{B}|} m_{j,i}\nu(\mathbf{b}_i)\nu(\mathbf{b}_j) - k_1\sum_{j=1}^{|\mathcal{B}|} m_{i,j}\nu(\mathbf{b}_i)\nu(\mathbf{b}_j) + \sum_{j=1}^{|\mathcal{A}|} m_{j,i}\nu(\mathbf{b}_i)\nu(\mathbf{a}_j)\right] - k_2\nu(\mathbf{b}_i) \tag{4.5}$$

where $|\mathcal{B}|$ is the number of lymphocytes and $|\mathcal{A}|$ the number of antigens. $m_{i,j}$ denotes the interaction strength between epitope $\mathbf{e}_i$ of lymphocyte $\mathbf{b}_i$ and paratope $\mathbf{p}_j$ of lymphocyte $\mathbf{b}_j$. The interaction strength (affinity) between two lymphocytes is calculated as the complementary match between the respective paratope and epitope strings. Two lymphocytes interact (bind) if their calculated interaction strength, $m_{i,j}$, is above a certain threshold.

In the above differential equation, the first term signifies the stimulation of paratope $\mathbf{p}_i$ by epitope $\mathbf{e}_j$; the second term represents the suppression of lymphocyte $\mathbf{b}_i$ whose epitope $\mathbf{e}_i$ is recognised by paratope $\mathbf{p}_j$; and the third term signifies the recognition of antigen $\mathbf{a}_j$. k_1 is a constant which regulates the inequality between stimulation and suppression and k_2 is the rate of annihilation. The constant rate c depends on the rate of lymphocyte/antibody production which is stimulated by an interaction. Therefore, c also depends on the number of interactions per time unit. Cloning and mutation of lymphocytes are proportional to the stimulation level. A higher stimulated lymphocyte produces more clones. This results in a diverse set of lymphocytes.

Hunt and Cooke developed a network based AIS model for classification of DNA strings into promoter or non-promoter classes [30, 93]. Each ALC in the model consists of a binary string which represents the ALC's paratope, a library of genes from which antibodies are generated, the DNA sequence and the level of stimulation. The antibodies are used for classification of unseen

DNA sequences. The affinity between two ALCs (or between an ALC and antigen) is calculated as the complementary match between the respective paratope and epitope strings. The model makes use of an affinity threshold to determine whether ALCs are linked to form a network or whether an ALC binds to an antigen. If the calculated affinity is greater than the affinity threshold, binding/linking occurs.

The population of ALCs is initialised by a cross-section of randomly selected DNA sequences from the training set. The remainder of the training set is used as antigen patterns. Antigen patterns are randomly selected and presented to a randomly selected ALC. The antigen is then presented to a percentage of the ALC's linked neighbours to determine whether any of the linked ALCs can bind to the antigen. If an ALC binds to the antigen, the ALC's stimulation level is calculated. The stimulation level of an ALC determines whether the ALC becomes active. If none of the linked ALCs are activated, an ALC is generated by using the presented antigen as template and added to the population of ALCs. Activated ALCs are cloned and mutated. Cloned ALCs are integrated into the network at the ALCs with which the clones have the highest affinity.

The stimulation level of an ALC, **b**, is based on the differential equation as proposed by Farmer *et al.* [49] (as defined in equation (4.5)). Cooke and Hunt adapted equation (4.5) such that

$$\vartheta(\mathbf{b}) = c\left[\sum_{j=1}^{|\mathcal{B}|} m\left(\mathbf{b},\mathbf{e}_j\right) - k_1 \sum_{j=1}^{|\mathcal{B}|} m\left(\mathbf{b},\mathbf{p}_j\right) + k_2 \sum_{j=1}^{|\mathcal{A}|} m\left(\mathbf{b},\mathbf{a}_j\right)\right] - k_3 \tag{4.6}$$

where $|\mathcal{B}|$ is the number of lymphocytes, $|\mathcal{A}|$ is the number of antigens, and m denotes the affinity between ALC **b** and the paratope **p** (or epitope **e**) of linked ALC $\mathbf{b}_j$.

The DNA classification model of Hunt and Cooke [30, 93] was improved and applied to case base reasoning [91, 92]. Each ALC in the model represents a case. ALCs are linked as a network if they represent similar cases, which could result in generalised cases. These generalised cases represented trends in data. The model was also applied to data mining [89], but there were however a few drawbacks which are discussed next.

The increasing size of the network made the model less scalable and a randomly initialised network of ALCs increased the time to built generalised cases. These drawbacks were addressed in [89] and the improved model was applied to fraud detection [89, 90, 141]. The proposed fraud detection system in [90] was called JISYS and implemented different matching techniques compared to the original model in [89]. Although JISYS delivered promising results, the model was limited to a specific domain of mortgage fraud detection.

Timmis developed a basic network based AIS model which was domain-independent [165]. The basic AIS model was based on the work of Hunt and Cooke [93]. The model performed cloning and mutation operations on a set of linked ALCs. The affinity between an ALC and an antigen pattern (or another ALC) is measured using the Euclidean distance. Two ALCs are linked if the calculated affinity between them is below the network affinity threshold (NAT). The model was able to produce three distinct clusters when applied to Fisher's Iris data set [51]. Timmis also developed a tool named aiVis to visualise the formed clusters [166].

A few drawbacks to the basic AIS model were highlighted in [169]. A drawback is that there is an exponential growth in the size of the network due to the incapability of the proposed mechanism to control the size of the ALC population. The exponential growth of the network also resulted in an unscalable model, increasing the computational complexity with each iteration. Furthermore, the formed networks are difficult to interpret. The model is also very sensitive to the NAT value. The authors proposed the Resource Limited AIS as an improvement to the basic AIS model, discussed below [169].

Another approach to enhance the model of Timmis was proposed by Wierchoń and Kużelewska [184]. The model of Timmis [165] was adapted in such a way that the set of ALCs is randomly initialised. Furthermore, the network affinity threshold in [184] is calculated as the average distance between the $|\mathcal{A}| \times k$ lowest distances in the antigen set, where $|\mathcal{A}|$ is the size of the antigen set and k some constant. The adapted model in [184] also improves on the model in [165] in that the maximum network size is limited to the number of training patterns in the training set and stable clusters are formed with a minimal number of control parameters.

Resource Limited AIS (AINE): AINE presented a new concept of artificial recognition balls (ARBs), bounded by a resource limited environment. A resource limited environment is defined as the maximum number of available B-Cells that is shared among ARBs in a population. Thus, each ARB allocates a number of resources based on the ARB's overall stimulation level. In summary, AINE consists of a population of ARBs, links between the ARBs, a set of antigen training patterns (of which a cross section is used to initialise the ARBs) and some clonal operations for learning. An ARB represents a region of antigen space that is covered by a certain type of B-Cell.

ARBs which are close to each other (similar) in antigen space are connected with weighted edges to form a number of individual network structures. The similarity (affinity) between ARBs and between the ARBs and an antigen is measured using Euclidean distance. Two ARBs are connected if the affinity between them is below the network affinity threshold (NAT). The NAT is the average distance between all the antigen patterns in the training set, calculated as [168]

$$NAT = \frac{2k}{|\mathcal{A}| \times (|\mathcal{A}|-1)} \sum_{i=1}^{|\mathcal{A}|-1} \sum_{j=i+1}^{|\mathcal{A}|} \sigma\left(\mathbf{a}_i, \mathbf{a}_j\right) \tag{4.7}$$

where σ is the Euclidean distance and k is a constant value such that $0 \leq k \leq 1$. The value of the NAT determines the linking between ARBs and therefore influences the number of formed networks. Algorithm 4.3 lists the pseudo code for AINE.

For each iteration, all training patterns in set $\mathcal{A}$ are presented to the set of ARBs, $\mathcal{B}$. After each iteration, each ARB, $\mathbf{b}$, calculates its stimulation level, ϑ, and allocates resources (B-Cells) based on its stimulation level as defined in equation (4.12). The stimulation level, ϑ, of an ARB, $\mathbf{b}$, is calculated as the summation of the antigen stimulation, ps, the network stimulation, ns, and the network suppression, nn. The stimulation level of an ARB is defined as follows [169]

$$\vartheta(\mathbf{b}) = ps(\mathbf{b}) + ns(\mathbf{b}) + nn(\mathbf{b}) \tag{4.8}$$

$$ps(\mathbf{b}) = \sum_{i=1}^{|\alpha_{\mathbf{b}}|} 1 - \alpha_i \tag{4.9}$$

$$ns(\mathbf{b}) = \sum_{j=1}^{|\lambda_{\mathbf{b}}|} 1 - \lambda_j \tag{4.10}$$

$$nn(\mathbf{b}) = -\sum_{j=1}^{|\lambda_{\mathbf{b}}|} \lambda_j \tag{4.11}$$

where $|\alpha_{\mathbf{b}}|$ is the normalised set of affinities between an ARB, $\mathbf{b}$, and all antigen $\mathbf{a} \in \mathcal{A}$ for which $\sigma(\mathbf{a}_i, \mathbf{b}) < NAT$. The antigen stimulation, ps, is thus the sum of all antigen affinities below the NAT threshold and $0 \leq \alpha_i \leq 1$; $\alpha_i \in \alpha_{\mathbf{b}}$. The network stimulation, ns, and the network suppression, nn, are the sum of affinities between an ARB and all the ARB's connected neighbours, as defined in equation (4.10) and equation (4.11) respectively. In equations (4.10) and (4.11), $|\lambda_{\mathbf{b}}|$ is the normalised set of affinities between an ARB, $\mathbf{b}$, and all other ARBs in set $\mathcal{B}$. The ns and nn terms are based on the summation of the distances to the $|\lambda_{\mathbf{b}}|$ linked neighbours of an

Algorithm 4.3: Artificial Immune Network (AINE)

```
Normalise the training data;
Initialise the ARB population, $\mathcal{B}$, using a randomly selected cross section of the
normalised training data;
Initialise the antigen set, $\mathcal{A}$, with the remaining normalised training data;
Set the maximum number of available resources, $R_{max}$;
for each ARB, $\mathbf{b}_i \in \mathcal{B}$, at index position i in $\mathcal{B}$ do
    for each ARB, $\mathbf{b}_j \in \mathcal{B}$, at index position j in $\mathcal{B}$ do
        Calculate the ARB affinity, $\sigma\left(\mathbf{b}_i, \mathbf{b}_j\right)$;
        if $\sigma\left(\mathbf{b}_i, \mathbf{b}_j\right) < NAT$ and $i \neq j$ then
            Add $\sigma\left(\mathbf{b}_i, \mathbf{b}_j\right)$ to the set of network stimulation levels, $\lambda_{\mathbf{b}_i}$;
        end
    end
end
while not stopping condition do
    for each antigen, $\mathbf{a}_i \in \mathcal{A}$, at index position i in $\mathcal{A}$ do
        for each ARB, $\mathbf{b}_j \in \mathcal{B}$, at index position j in $\mathcal{B}$ do
            Calculate the antigen affinity, $\sigma\left(\mathbf{a}_i, \mathbf{b}_j\right)$;
            if $\sigma\left(\mathbf{a}_i, \mathbf{b}_j\right) < NAT$ then
                Add $\sigma\left(\mathbf{a}_i, \mathbf{b}_j\right)$ to the set of antigen stimulation levels, $\alpha_{\mathbf{b}_j}$;
            end
        end
    end
    Allocate resources (see algorithm 4.4) to the set of ARBs, $\mathcal{B}$;
    Clone and mutate the remaining ARBs in $\mathcal{B}$;
    Integrate mutated clones into $\mathcal{B}$;
end
```

ARB, **b**. The network suppression, *nn*, is the dissimilarity between an ARB and neighbouring ARBs in the network. Network suppression keeps the size of the ARB population under control.

Algorithm 4.4: Resource allocation in AINE

```
Set the number of allocated resources, $R_T = 0$;
for each ARB, $\mathbf{b}_i \in \mathcal{B}$, at index position i in $\mathcal{B}$ do
    Allocate resources, $R(\mathbf{b}_i)$;
    $R_T = R_T + R(\mathbf{b}_i)$;
end
Sort the set of ARBs, $\mathcal{B}$, in ascending order of $R$;
if $R_T > R_{max}$ then
    $z = R_T - R_{max}$;
    for each ARB, $\mathbf{b}_i \in \mathcal{B}$, at index position i in $\mathcal{B}$ do
        $q = R(\mathbf{b}_i)$;
        if $q = 0$ then
            Remove $\mathbf{b}_i$ from set $\mathcal{B}$;
        end
        else
            $q = q - z$;
            if $q \leq 0$ then
                Remove $\mathbf{b}_i$ from set $\mathcal{B}$;
                $z = -q$;
            end
            else
                $R(\mathbf{b}_i) = q$;
                break;
            end
        end
    end
end
```

Algorithm 4.4 lists the pseudo code for resource allocation to the set of ARBs. The number of resources allocated to an ARB is calculated as

$$R(\mathbf{b}) = R_k \times \left(\vartheta'(\mathbf{b})^2\right) \tag{4.12}$$

where ϑ' is the normalised stimulation level and R_k some constant. Since the stimulation level of the ARBs in $\mathcal{B}$ are normalised, some of the ARBs will have no resources allocated. Thus, after the resource allocation, the weakest ARBs (zero resources) are removed from the population of

ARBs. Each of the remaining ARBs in the population, $\mathcal{B}$, is then cloned and mutated if the calculated ϑ of the ARB is above a certain threshold. These mutated clones are then integrated into the population by re-calculating the network links between the ARBs in $\mathcal{B}$.

The *stopping condition* can be based on whether the maximum size of $\mathcal{B}$ has been reached. Since ARBs compete for resources (based on their stimulation level), an upper limit is set to the number of resources (B-Cells) available in the model. The specified upper limit of resources has an influence on the performance of AINE. If the number of available resources is too large, the network will become too large and difficult to interpret (as in the case of the basic network based AIS [165]). A too small value will result in small networks, which are a premature representation of the antigen patterns. The Self Stabilising AIS model was proposed by Neal [140, 142] to address the drawback of setting the upper limit of available resources in AINE. The population of ARBs is also overtaken by a few ARBs with high stimulation levels that match a small number of antigen, resulting in the premature convergence of the population of ARBs [115]. The fuzzy AINE proposed by Nasraoui *et al.* [139, 137] improves AINE on this drawback.

Self Stabilising AIS: AINE was improved and simplified by a model proposed by Neal, namely the self stabilising AIS [140]. The main difference between these two models is that the SSAIS has no shared/distributed pool with a fixed number of resources that ARBs must compete for. The resource level of an ARB is increased if the ARB has the highest stimulation for an incoming pattern. Each ARB calculates its resource level locally. After a data pattern has been presented to all of the ARBs, the resource level of the most stimulated ARB is increased by addition of the ARB's stimulation level. Algorithm 4.5 lists the pseudo code for the self stabilising AIS. The differences between algorithm 4.3 (AINE) and algorithm 4.5 (SSAIS) are discussed next.

SSAIS defines the stimulation level, ϑ, of an ARB as [140]

$$\vartheta(\mathbf{b},\mathbf{a}) = ps(\mathbf{b},\mathbf{a}) + sns(\mathbf{b}) \tag{4.13}$$

$$ps(\mathbf{b},\mathbf{a}) = 1 - \sigma(\mathbf{b},\mathbf{a}) \tag{4.14}$$

$$sns(\mathbf{b}) = \frac{ns(\mathbf{b})}{|\lambda_{\mathbf{b}}|} \tag{4.15}$$

where ns is defined in equation (4.10) and $\sigma(\mathbf{b},\mathbf{a})$ is the Euclidean distance between an ARB, $\mathbf{b}$, and a training pattern, $\mathbf{a}$, in normalised data space, i.e. $0 \leq \sigma(\mathbf{b},\mathbf{a}) \leq 1$. The sns term is based on the average of the summation of the distances to the $|\lambda_{\mathbf{b}}|$ linked neighbours of an ARB, $\mathbf{b}$.

Algorithm 4.5: Self Stabilising AIS

```
Normalise the training data;
Initialise the ARB population, $\mathcal{B}$, using a cross section of the normalised training data;
Initialise the antigen set, $\mathcal{A}$, with the remaining normalised training data;
for each antigen, $\mathbf{a} \in \mathcal{A}$ do
    Present $\mathbf{a}$ to each $\mathbf{b} \in \mathcal{B}$;
    Calculate stimulation level, $\vartheta$, for each ARB, $\mathbf{b}$;
    Select the ARB with the highest calculated stimulation level, $\vartheta$, as $\mathbf{h}$;
    Increase the resource level of $\mathbf{h}$;
    for each ARB, $\mathbf{b} \in \mathcal{B}, \mathbf{b} \neq \mathbf{h}$ do
        Deplete resources of $\mathbf{b}$;
    end
    Remove ARBs with the number of allocated resources less than the mortality
    threshold, $R_\Lambda$;
    Generate $\eta$ clones of $\mathbf{h}$ and mutate;
    Integrate clones (mutated or not) into $\mathcal{B}$;
end
```

The *nn* term defined in equation (4.11) is discarded to prevent premature convergence of ARBs to dominating training patterns.

For each training pattern, $\mathbf{a} \in \mathcal{A}$, presented to the network of ARBs, $\mathcal{B}$, the resource level of each ARB, $\mathbf{b}$, that does not have the highest calculated ϑ is geometrically decayed by the following function [140]

$$R(\mathbf{b}, \mathbf{a}_i) = R_\gamma \times R(\mathbf{b}, \mathbf{a}_{i-1}) \tag{4.16}$$

where $R(\mathbf{b}, \mathbf{a}_i)$ is the number of resources for an ARB, $\mathbf{b}$, after being presented to i training patterns. R_γ is the decaying rate of resources for an ARB. All ARBs with a resource level less than the fixed predefined mortality threshold, R_Λ, are culled from the network. Resources are only allocated by the ARB, $\mathbf{h}$, with the highest calculated stimulation level, ϑ. The number of resources allocated to ARB $\mathbf{h}$ with the highest ϑ is calculated as [140]

$$R(\mathbf{h}, \mathbf{a}_i) = R_\gamma \times (R(\mathbf{h}, \mathbf{a}_{i-1}) + \vartheta(\mathbf{h}, \mathbf{a}_i)) \tag{4.17}$$

where $\vartheta(\mathbf{h}, \mathbf{a}_i)$ is the stimulation level of ARB $\mathbf{h}$ after being presented to i training patterns.

The highest stimulated ARB, **h**, generates η clones. The number of clones generated is given as [140]

$$\eta = \frac{R(\mathbf{h}, \mathbf{a}_i)}{R_\Lambda \times 10}$$

where R_Λ is the mortality threshold. Thus, the number of clones generated by an ARB is proportional to the resource level of the ARB. The generated clones are mutated with a fixed mutation rate. If a clone is mutated, the clone is assigned $R_\Lambda \times 10$ resources from the ARB's resource level. Clones (mutated or not) are integrated with the network of ARBs, $\mathcal{B}$.

The SSAIS resulted in a model that can adapt to continuously changing data sets and a genuinely stable AIS. A drawback to SSAIS is that the final networks that are formed have poor data compression and the SSAIS model has a time lag to adapt to the introduction of a new region of data due to the lack of diversity of the network of ARBs. The stable memory artificial immune network (SMAIN) was proposed by Neal as a simplification of the SSAIS model [142] and is explained next.

Stable Memory Artificial Immune Network (SMAIN): The main difference between SSAIS and SMAIN is the elimination of the mutation operator [142]. The ARB population, $\mathcal{B}$, is initialised with a cross section, $\mathcal{B}_{init}$, of the training data. Each ARB is initialised with R_{init} resources. Furthermore, cloning in SMAIN is only performed on the ARB, **h**, with the closest distance to an antigen pattern, **a**, if the measured distance is greater than the NAT threshold. Whenever an antigen triggers the cloning of an ARB, the antigen is initialised as a cloned ARB. Half of the parent ARB's resources is then assigned to the cloned ARB and the clone is integrated with the ARB population. There is no mutation operator on the clone. Algorithm 4.6 lists the pseudo code for SMAIN. The differences between algorithm 4.5 (SSAIS) and algorithm 4.6 (SMAIN) are further discussed. The stimulation level, ϑ, in equation (4.13) is redefined in SMAIN as [142]

$$\vartheta(\mathbf{b}, \mathbf{a}) = ps(\mathbf{b}, \mathbf{a}) + sns(\mathbf{b}) \tag{4.18}$$

where

$$ps(\mathbf{b}, \mathbf{a}) = \frac{1}{1 + \sigma(\mathbf{b}, \mathbf{a})} \tag{4.19}$$

and

$$sns(\mathbf{b}) = \sum_{j=1}^{|\lambda_{\mathbf{b}}|} \lambda_j \tag{4.20}$$

Algorithm 4.6: Stable Memory Artificial Immune Network

```
Initialise the ARB population, $\mathcal{B}$, using a cross section of the training data;
Initialise the antigen set, $\mathcal{A}$, with the remaining training data;
for each antigen, $\mathbf{a} \in \mathcal{A}$ do
    Present $\mathbf{a}$ to each $\mathbf{b} \in \mathcal{B}$;
    Calculate stimulation level, $\vartheta$, for each ARB, $\mathbf{b} \in \mathcal{B}$;
    Increase the resource level of each $\mathbf{b} \in \mathcal{B}$;
    Select the ARB with the lowest measured distance as $\mathbf{h}$;
    if $\sigma(\mathbf{h},\mathbf{a}) \geq NAT$ then
        Initialise $\mathbf{a}$ as a clone of ALC $\mathbf{h}$;
        Add clone to $\mathcal{B}$;
    end
    for each ARB, $\mathbf{b} \in \mathcal{B}$ do
        Deplete resources of $\mathbf{b}$;
        Remove $\mathbf{b}$ from $\mathcal{B}$ if the number of allocated resources are less than the mortality
        threshold, $R_\Lambda$;
    end
end
```

where *sns* is still based on the neighbours of an ARB but simplified by removing the need to normalise the distances to the neighbours as in equation (4.15). The resource level of an ARB is calculated as [142]

$$R(\mathbf{b},\mathbf{a}_i) = R(\mathbf{b},\mathbf{a}_{i-1}) + \left(R_k \times \left(R_{max} - R_{decay}\right)\right) \times \vartheta(\mathbf{b},\mathbf{a}_i) \qquad (4.21)$$

where $R_k \in (0,1)$ is a constant, R_{max} is the maximum number of resources an ARB can allocate and R_{decay} is the decayed resource level of an ARB as defined in equation (4.16). Similar to SSAIS, all ARBs with a resource level less than the mortality threshold, R_Λ, are culled from the network. SMAIN generates stable memory networks which represents structures inherent in complex data sets.

Fuzzy Artificial Immune Network (Fuzzy AINE): Another enhancement to the AINE model is the fuzzy AINE proposed by Nasraoui *et al.* [136, 139]. The fuzzy AINE was applied to the clustering (profiling) of session patterns for a specific web site. ARBs in the fuzzy AINE are referred to as fuzzy ARBs, since each training pattern is grouped with all fuzzy ARBs to a certain degree of membership (similar to the Fuzzy C-means algorithm which was explained in section 2.3.2). Compared to an ARB in AINE, a fuzzy ARB represents a single pattern as center to a set of member antigen patterns which is within the fuzzy ARB's influence radius. The membership function (or degree of membership) between fuzzy ARB $\mathbf{b}_i$ and antigen pattern $\mathbf{a}_j$ is calculated as [136, 139]

$$m_{ij} = \exp\left(-\frac{\sigma\left(\mathbf{b}_i, \mathbf{a}_j\right)^2}{2\phi_i^2}\right) \tag{4.22}$$

where σ is the distance between $\mathbf{b}_i$ and $\mathbf{a}_j$. The radius of influence, ϕ, is similar to the NAT threshold in [168], but local to each fuzzy ARB. Thus, each fuzzy ARB has a different radius of influence. The membership function decreases with an increase in distance between the fuzzy ARB and the antigen pattern. This results in the gradual exclusion of distant antigen patterns from the fuzzy ARB, resulting in a robust weight function which decreases the influence of outliers. The radius of influence, ϕ_i^2, of each fuzzy ARB $\mathbf{b}_i$ is updated after each iteration using [139]

$$\phi_{i,J}^2 = \frac{\sum_{j=1}^{J} m_{ij}\sigma\left(\mathbf{b}_i, \mathbf{a}_j\right)^2 - \beta(t)\sum_{k=1}^{|\mathcal{B}|} m_{ik}\sigma\left(\mathbf{b}_i, \mathbf{b}_k\right)^2}{\sum_{j=1}^{J} m_{ij} - \beta(t)\sum_{k=1}^{|\mathcal{B}|} m_{ik}} \tag{4.23}$$

where $J = |\mathcal{A}|$. The second term in both the numerator and denominator denote the suppression of similar fuzzy ARBs with overlapping radii of influence. Therefore the stimulation level of a fuzzy ARB consists of the density of the antigen patterns surrounding the fuzzy ARB (as antigen stimulation) and the density of neighbouring fuzzy ARBs (as penalty for suppression). The stimulation level, ϑ, of $\mathbf{b}_i$ is calculated as [139]

$$\vartheta(\mathbf{b}_i) = s_i(\mathcal{A}, |\mathcal{A}|) - \beta(t)\, s_i(\mathcal{B}, |\mathcal{B}|) \tag{4.24}$$

where

$$s_i(X, J) = \frac{\sum_{j=1}^{J} m_{ij}}{\phi_{i,J}^2} \tag{4.25}$$

$s_i(X, |X|)$ calculates the density of patterns within set X surrounding $\mathbf{b}_i$; s_i calculates the antigen stimulation for $X = \mathcal{A}$ and the suppression for $X = \mathcal{B}$. Algorithm 4.7 provides the pseudo code for fuzzy AINE.

Algorithm 4.7: Fuzzy Artificial Immune Network (Fuzzy AINE)

```
Normalise the training data;
Initialise the fuzzy ARB population, $\mathcal{B}$, using a randomly selected cross section of the
normalised training data;
Initialise $\phi^2$ for each fuzzy ARB;
Initialise the antigen set, $\mathcal{A}$, with the remaining normalised training data;
Set the maximum number of available resources, $R_{max}$;
while not stopping condition do
    for each antigen, $\mathbf{a}_j \in \mathcal{A}$, at index position $j$ in $\mathcal{A}$ do
        for each ARB, $\mathbf{b}_i \in \mathcal{B}$, at index position $i$ in $\mathcal{B}$ do
            Update membership function $m_{ij}$ of fuzzy ARB $\mathbf{b}_i$;
        end
    end
    for each ARB, $\mathbf{b}_i \in \mathcal{B}$, at index position $i$ in $\mathcal{B}$ do
        Calculate the simulation level $\vartheta(\mathbf{b}_i)$;
        Update the radius influence $\phi_i^2$ of fuzzy ARB $\mathbf{b}_i$;
    end
    Allocate resources (see algorithm 4.4) to the set of fuzzy ARBs, $\mathcal{B}$;
    Clone and mutate remaining fuzzy ARBs in $\mathcal{B}$;
    Integrate mutated clones into $\mathcal{B}$;
end
```

To avoid the premature convergence of the population of fuzzy ARBs, the number of resources allocated to a fuzzy ARB is calculated as [136, 137]

$$R(\mathbf{b}_i) = R_k \times \left(\log\left[\vartheta'(\mathbf{b}_i)\right]\right) \tag{4.26}$$

where ϑ' is the normalised stimulation level of a fuzzy ARB and R_k some constant. This modification to the number of resources allocated to a fuzzy ARB will limit the influence of those fuzzy ARBs with high stimulation to slowly overtake the population.

A cloned fuzzy ARB also inherits the radius of influence ϕ^2 value of the parent fuzzy ARB. After the integration of the mutated clones, the fuzzy ARBs with the same B-Cell representation are merged into one fuzzy ARB. The merging of identical fuzzy ARBs limit the high rate of population growth. The merging of two fuzzy ARBs, $\mathbf{b}_i$ and $\mathbf{b}_j$, is done through a crossover operator on the fuzzy ARBs' attributes, defined as [137, 139]

$$\mathbf{b}_{k,n} = avg\left(\mathbf{b}_{i,n}, \mathbf{b}_{j,n}\right)$$

where $\mathbf{b}_{k,n}$ is the value for attribute n of merged ARB, $\mathbf{b}_k$, and avg is the average value between the n-th attributes in $\mathbf{b}_i$ and $\mathbf{b}_j$ respectively.

In the context of data clustering, fuzzy AINE maintains a diverse set of fuzzy ARBs to represent the different clusters within a data set, compared to a few good ARBs in AINE which dominates the population of ARBs, and therefore the population prematurely converges. The fuzzy AINE proved to be scalable and diverse in profiling web usage session patterns. The Euclidean distance in AINE (affinity measurement) was replaced by the calculation of the cosine similarity between two session patterns in fuzzy AINE [137, 139]. The cosine similarity between two vectors is defined in equation (2.8).

A drawback to fuzzy AINE in the context of web usage profiling is the assumption that all web usage sessions are available beforehand. This is a disadvantage in environments with limited resources (like system memory), making the fuzzy AINE model less scalable. Another drawback, which is common to most clustering algorithms, is that any change in web usage sessions results in the re-application of the model to cluster the data. Nasraoui *et al.* highlighted these drawbacks in [134, 138] and improved the fuzzy AINE with the Dynamic Weighted B-Cell AIS model which is able to cluster streaming non-stationary web usage sessions [134].

Dynamic Weighted B-Cell AIS: Nasraoui *et al.* proposed a scalable AIS model which can be applied to the clustering of non-stationary data [134, 135, 138]. The model is similar to fuzzy AINE in that each training pattern is grouped with all ARBs to a certain degree of membership. An ARB in this model is known as a dynamic weighted B-Cell (DWB-cell). The membership function of fuzzy AINE (as defined in equation (4.22)) is adapted in [135] to be more applicable for non-stationary environments. The adapted membership function includes the time when an antigen pattern was presented to the network of DWBs. The membership function for DWB, $\mathbf{b}_i$, is calculated as [135]

$$m_{ij} = \exp\left[-\left(\frac{\sigma\left(\mathbf{b}_i, \mathbf{a}_j\right)^2}{2\phi_i^2} + \frac{j}{\tau}\right)\right] \tag{4.27}$$

where σ is the distance between $\mathbf{b}_i$ and the j-th antigen pattern, $\mathbf{a}_j$. Thus the antigen index j increases monotonically with time. τ controls the rate at which previously presented antigens contribute to the degree of membership as well as the relevance of the network. The above membership function not only decreases with an increase in distance between a DWB-cell and an antigen pattern as in fuzzy AINE, but also with the time since an antigen pattern has been presented to the network of DWBs. Therefore the most recent antigen patterns presented to a DWB-cell have a higher degree of membership compared to less current antigen patterns.

The DWB population, $\mathcal{B}$, has a maximum of $\mathcal{B}_{max}$ DWB-cells and is initialised with the first $\mathcal{B}_{max}$ of incoming antigen training patterns. The radius of influence, ϕ^2, of each DWB-cell is initialised with ϕ_{init}. The antigen stimulation level of a DWB-cell after J antigen patterns is given as [135]

$$s_{i,J} = s_i\left(\mathcal{A}, J\right) \tag{4.28}$$

where $s_i\left(\mathcal{A}, J\right)$ is defined in equation (4.25) and the radius of influence, ϕ^2, is given as

$$\phi^2_{i,J} = \frac{\sum_{j=1}^{J} m_{ij}\sigma\left(\mathbf{b}_i, \mathbf{a}_j\right)^2}{2\sum_{j=1}^{J} m_{ij}} \tag{4.29}$$

In order to calculate a DWB-cell's stimulation level and radius of influence after each antigen pattern has been presented to the DWB-cell, the following derivatives from the above equations are given in [135] as approximate incremental updates for the stimulation level and radius of influence of $\mathbf{b}_i$ respectively:

$$s_{i,J} = \frac{\exp\left(-\frac{1}{\tau}\right) M_{i,J-1} + m_{iJ}}{\phi^2_{i,J}} \tag{4.30}$$

where

$$\phi^2_{i,J} = \frac{\exp\left(-\frac{1}{\tau}\right) \phi^2_{i,J-1} M_{i,J-1} + m_{iJ}\sigma\left(\mathbf{b}_i, \mathbf{a}_J\right)^2}{2\left[\exp\left(-\frac{1}{\tau}\right) M_{i,J-1} + m_{iJ}\right]} \tag{4.31}$$

and

$$M_{i,J-1} = \sum_{j=1}^{J-1} m_{ij} \tag{4.32}$$

Algorithm 4.8 lists the pseudo code for the dynamic weighted B-Cell model. The model also proposed the incorporation of a dynamic stimulation/suppression factor into the stimulation level of a DWB-cell to control the proliferation and redundancy of DWB-cells in the network. Thus, old sub-nets die if not re-stimulated by current incoming antigen patterns. The total stimulation

Algorithm 4.8: Dynamic Weighted B-Cell

```
Set the maximum size of the DWB population as $\mathcal{B}_{max}$;
Initialise the DWB population, $\mathcal{B}$, using the first $\mathcal{B}_{max}$ of incoming antigen training
patterns in set $\mathcal{A}$;
Initialise $\phi^2 = \phi_{init}$ for each DWB;
Compress the population of DWBs into $k_{compress}$ sub-nets using K-means clustering;
for each antigen, $\mathbf{a}_j \in \mathcal{A}$, at index position $j$ in $\mathcal{A}$ do
    Present $\mathbf{a}_j$ to the centroid $\mathbf{c}_k$ of each sub-net $C_k$ and calculate the weight $m_{kj}$ and
    update $\phi_k^2$ using equations (4.27) and (4.31) respectively;
    Select the sub-net with the maximum $m_{kj}$ as the most activated subnet $C_w$;
    if $\forall \mathbf{b}_i \in C_w$ the weight $m_{ij} < m_{min}$ then
        Create new DWB-cell $\mathbf{x} = \mathbf{a}_j$ and set the new cell's $\phi^2 = \phi_{init}$;
        Add new DWB-cell to the population;
    end
    else
        for each DWB, $\mathbf{b}_i \in C_w$ do
            if $m_{ij} \geq m_{min}$ then
                Reset $age_i = 0$ of DWB $\mathbf{b}_i$;
            end
            else
                Increment $age_i = age_i + 1$ of DWB $\mathbf{b}_i$;
            end
            Calculate the stimulation level of $\mathbf{b}_i$ using equation (4.34);
            Update the radius of influence, $\phi_i^2$, using equation (4.35);
        end
    end
    Clone and mutate DWB-cells;
    if $|\mathcal{B}| > \mathcal{B}_{max}$ then
        Sort population of DWBs in ascending order of their stimulation levels;
        Remove top $|\mathcal{B}| - \mathcal{B}_{max}$ DWB-cells from the sorted population;
    end
    Compress the population of DWBs every $A$ antigens into $k_{compress}$ sub-nets using
    K-means clustering with the previous centroids as initial centroids;
end
```

of a DWB-cell consists of the antigen stimulation as well as the co-stimulation and suppression from other DWB-cells in the network. The total stimulation of $\mathbf{b}_i$ is given as [135]

$$\vartheta(\mathbf{b}_i) = s_{i,J} + \alpha(age_i)\frac{\sum_{n=1}^{|\mathcal{B}|} m_{in}}{\phi_{i,J}^2} - \beta(age_i)\frac{\sum_{n=1}^{|\mathcal{B}|} m_{in}}{\phi_{i,J}^2} \tag{4.33}$$

where $\alpha(age_i) = \left(1 + \frac{age_i}{\tau_\alpha}\right)^{-1}$ is the co-stimulation coefficient, $\beta(age_i) = \left(1 + \frac{age_i}{\tau_\beta}\right)^{-1}$ is the network suppression coefficient and age_i records the age of $\mathbf{b}_i$.

Another drawback of existing immune network based learning models is that the number of interactions between the B-Cells in the network and a specific antigen are immense. The model of [135] clusters the DWB-cells into $k_{compress}$ sub-nets using K-means clustering (as discussed in section 2.3.2) to decrease the number of interactions between an antigen pattern and the DWB-cells in the network. The centroids of each of these formed clusters (or sub-nets) are used to represent the sub-nets and interact with the presented antigen pattern. Therefore the network of DWB-cells is compressed in different sub-nets and the total stimulation level and radius of influence for each DWB-cell in a specific sub-net C_k is calculated as [135]

$$\vartheta(\mathbf{b}_i) = s_{i,J} + \alpha(age_i)\frac{\sum_{\forall \mathbf{b}_n \in C_k} m_{in}}{\phi_{i,J}^2} - \beta(age_i)\frac{\sum_{\forall \mathbf{b}_n \in C_k} m_{in}}{\phi_{i,J}^2} \tag{4.34}$$

where $\mathbf{b}_i \in C_k$ and

$$\phi_{i,J}^2 = \frac{D_{i,J}^2 + \alpha(age_i)\sum_{\forall \mathbf{b}_n \in C_k} m_{in}\sigma(\mathbf{b}_i,\mathbf{b}_n)^2 - \beta(age_i)\sum_{\forall \mathbf{b}_n \in C_k} m_{in}\sigma(\mathbf{b}_i,\mathbf{b}_n)^2}{2\left[M_{i,J} + \alpha(age_i)\sum_{\forall \mathbf{b}_n \in C_k} m_{in} - \beta(age_i)\sum_{\forall \mathbf{b}_n \in C_k} m_{in}\right]} \tag{4.35}$$

where

$$D_{i,J}^2 = \exp\left(-\frac{1}{\tau}\right)\phi_{i,J-1}^2 M_{i,J-1} + m_{iJ}\sigma(\mathbf{b}_i,\mathbf{a}_J)^2 \tag{4.36}$$

$$M_{i,J} = \exp\left(-\frac{1}{\tau}\right) M_{i,J-1} + m_{iJ} \tag{4.37}$$

A DWB-cell is only cloned if it reached maturity and is activated by an antigen pattern. A

DWB-cell is activated when $m_{ij} > m_{min}$, where m_{min} is the minimum degree to become activated. Maturity of a DWB-cell is reached when the cell's age age_i is within a specified range, $a_{min} \leq age_i \leq a_{max}$. Cloning of a DWB-cell is proportional to the cell's stimulation level. If a DWB-cell's age exceeds the maximum time threshold ($age_i > a_{max}$), cloning of the cell is prevented, thus increasing the probability to clone newer DWB-cells. The number of clones generated for an activated and mature DWB-cell, $\mathbf{b}_i$, is given as [135]

$$\eta(\mathbf{b}_i) = k_{clone} \times \frac{\vartheta(\mathbf{b}_i)}{\sum_{n=1}^{|\mathcal{B}|} \vartheta(\mathbf{b}_n)} \qquad \text{if} \quad a_{min} \leq age_i \leq a_{max} \tag{4.38}$$

Whenever the maximum size, $\mathcal{B}_{max}$, of the network of DWB-cells has been reached, the DWB-cells are sorted in ascending order of their stimulation levels and starting from the top, the DWB-cells with the lowest stimulation levels are removed until the size of the network is equal to the maximum size, $\mathcal{B}_{max}$.

The mechanism of somatic hyper mutation is computationally expensive and replaced in the DWB-model by a different concept, namely *dendritic injection*. When the immune network encounters an antigen that the network cannot react to, the specific antigen is initialised as a DWB-cell. Thus, new information is *injected* into the immune network, i.e. *dendritic injection*. The dendritic cell system was discussed in section 3.8. The DWB-model has proven to be robust to noise, adaptive, and scalable in learning antigen structures in a non-stationary environment.

aiNet: De Castro and Von Zuben proposed a novel network based AIS model which evolves a population of linked memory ALCs through clonal selection [32, 36]. The model is applied to the problem of data clustering. A typical ALC network consists of ALC nodes (the B-Cells or antibodies) which are connected by edges to form node pairs. A weight value (connection strength) is assigned to each edge to indicate the similarity between two nodes. Thus, the ALC network that is formed during training is presented by an edge-weighted graph. Edges in the ALC network are pruned by measuring the weight of each edge against a *similarity* threshold. Pruning the ALC network results in the formation of a number of sub-networks. Each of the formed sub-networks represents a cluster within the data set. Thus, the evolved population of linked memory ALCs contains a number of ALC networks, each representing a cluster in the data set.

The data patterns in the training set, $\mathcal{A}$, are seen as antigens. Training of the memory ALC population, $\mathcal{B}$, follows an iterative two phase approach. The first phase implements the process of clonal selection as proposed by the CLONALG model of De Castro and Von Zuben [35]. The second phase is the *network formation* with *network suppression* between the evolved memory ALCs. During training, each pattern is presented to the population of memory ALCs. The affinity between an antigen pattern and an ALC is inversely proportional to the measured Euclidean distance between the antigen pattern and the ALC. Thus, a higher measured Euclidean distance results in a lower affinity measurement and vice-versa. Euclidean distance was defined in equation (2.3). The affinity between an ALC, **b**, and an antigen pattern, **a**, is calculated as [36]

$$f(\mathbf{b},\mathbf{a}) = \frac{1}{\sigma(\mathbf{b},\mathbf{a})} \tag{4.39}$$

where σ is the Euclidean distance. After the affinity to each ALC is calculated, a subset of the n highest affinity ALCs is selected for cloning. The number of clones to generate for an ALC is proportional to the ALC's affinity to the antigen pattern, **a**. Therefore, a higher affinity results in more clones. The number of clones for the n-th selected ALC is defined as [36]

$$\eta(\mathbf{b}_n) = round\left(|\mathcal{B}| - \sigma(\mathbf{b}_n,\mathbf{a}) \times |\mathcal{B}|\right) \tag{4.40}$$

where $|\mathcal{B}|$ is the cardinality of the ALC set $\mathcal{B}$ and σ is the Euclidean distance between antigen **a** and the n-th selected ALC, $\mathbf{b}_n$. Each of the generated clones are then mutated. An ALC clone, $\mathbf{b}^*$, is mutated as [32, 36]

$$\mathbf{b}^{'} = \mathbf{b}^* - \varsigma(\mathbf{b}^* - \mathbf{a}) \tag{4.41}$$

where ς is the mutation rate on ALC clone $\mathbf{b}^*$. The mutation rate of a clone is inversely proportional to the affinity of the clone's parent ALC. Thus, a higher affinity level results in a smaller mutation rate. The affinity between the antigen pattern and each of the mutated clones is then calculated.

A clonal memory set is then selected from the mutated clones. The clonal memory set contains ζ% of the mutated clones with the highest affinity. The Euclidean distance between the antigen pattern and each memory clone in the clonal memory set is measured against a death threshold, ε_{death}. Memory clones with a measured Euclidean distance above ε_{death} are removed from the clonal memory set. The Euclidean distance between each of the remaining memory clones is then calculated as the network distance. Clones with a network distance below the network suppression threshold, $\varepsilon_{network}$, are also removed from the clonal memory set, i.e. clonal suppression. The remaining clonal memory set is then concatenated with the set of memory ALC networks, $\mathcal{B}$.

After all the antigen patterns have been presented to the memory set of ALC networks, the Euclidean distance between ALCs in the memory set is measured against $\varepsilon_{network}$. ALCs with a measured Euclidean distance below the $\varepsilon_{network}$ threshold are removed from the memory set, i.e. network suppression. A percentage (φ%) of the lowest affinity (dissimilar) ALCs in $\mathcal{B}$ is replaced with randomly generated ALCs. The remaining memory set is then used in the next iteration. Algorithm 4.9 provides the pseudo code of the aiNet model.

The *stopping condition* of the *while*-loop can be one of the following [32, 36]:

1. **Setting a *loop* counter**: A counter can be set to determine the number of loops.
2. **Setting the maximum size of the network**: The while-loop can be stopped when the size of the network reaches a maximum.
3. **Testing for convergence**: The loop terminates when the average error between the training patterns in $\mathcal{A}$ and ALCs in $\mathcal{B}$ rises after a number of consecutive loops.

The final network of memory ALCs, $\mathcal{B}$, is partitioned with an agglomerative hierarchical clustering technique with single linkage (as discussed in section 2.3.1) to determine [36]

- the number of clusters presented by the network of memory ALCs $\mathcal{B}$,
- the spatial distribution of these clusters, and
- which ALCs belong to the same cluster.

De Castro and Von Zuben also proposed the minimal spanning tree as an alternative technique to determine the above goals [36]. Since the network of memory ALCs can be presented as a graph with weighted edges, a minimal spanning tree is generated as a sub-graph of the network of memory ALCs, such that the summed weights of the tree is minimised.

As discussed in chapter 2, each cluster can be represented by a centroid. De Castro and Von Zuben proposed the application of a fuzzy membership function to determine the degree of membership between the centroids and each of the memory ALCs (similar to the fuzzy C-means clustering algorithm as explained in section 2.3.2).

Algorithm 4.9: aiNet Learning Algorithm

```
Determine the antigen patterns as training set $\mathcal{A}$ and initialise the set of memory ALCs, $\mathcal{B}$;
while stopping condition not true do
    for each antigen pattern, $\mathbf{a}_j \in \mathcal{A}$ do
        for each ALC, $\mathbf{b}_i \in \mathcal{B}$ do
            Calculate the antigen affinity $f\left(\mathbf{b}_i, \mathbf{a}_j\right)$;
        end
        Select $n$ of the highest affinity ALCs as set $\mathcal{H}$;
        for each $\mathbf{b}_n \in \mathcal{H}$ do
            Create $\eta\left(\mathbf{b}_n\right)$ mutated clones of $\mathbf{b}_n$ and add to set $\mathcal{H}^{'}$;
        end
        for each $\mathbf{b}_n^{'} \in \mathcal{H}^{'}$ do
            Calculate the antigen affinity, $f\left(\mathbf{b}_n^{'}, \mathbf{a}_j\right)$;
        end
        Select $\zeta$% of the highest affinity antibodies as set $\mathcal{M}$;
        for each $\mathbf{y}_m \in \mathcal{M}$ do
            if $f_a\left(\mathbf{a}_j, \mathbf{y}_m\right) > \varepsilon_{death}$ then
                Remove $\mathbf{y}_m$ from $\mathcal{M}$;
            end
        end
        for each $\mathbf{y}_{m_1} \in \mathcal{M}$ do
            for each $\mathbf{y}_{m_2} \in \mathcal{M}$ do
                if the network affinity $f_a\left(\mathbf{y}_{m_1}, \mathbf{y}_{m_2}\right) < \varepsilon_{network}$ then
                    Remove $\mathbf{y}_{m_1}$ and $\mathbf{y}_{m_2}$ from $\mathcal{M}$;
                end
            end
        end
        $\mathcal{B} = \mathcal{B} \cup \mathcal{M}$;
    end
    for each $\mathbf{b}_{i_1} \in \mathcal{B}$ do
        for each $\mathbf{b}_{i_2} \in \mathcal{B}$ do
            if the network affinity $f_a\left(\mathbf{b}_{i_1}, \mathbf{b}_{i_2}\right) < \varepsilon_{network}$ then
                Remove $\mathbf{b}_{i_1}$ and $\mathbf{b}_{i_2}$ from $\mathcal{B}$;
            end
        end
    end
    Replace $\varphi$% of the lowest affinity ALCs in $\mathcal{B}$ with randomly generated ALCs;
end
```

Drawbacks of the aiNet model include the number of parameters that need to be specified and that the cost of computation increases as the dimension of the training patterns increases. The minimum spanning tree will also have difficulty in determining the network clusters if there are intersections between the clusters in the training data set. The aiNet is capable of reducing data redundancy and obtaining a compressed representation of the data.

There exists many different document clustering techniques, but these techniques have the main drawback that they directly apply the clustering techniques to the raw data (collection of documents). Larger collections contain more noise in the data which result in the formation of clusters with inferior quality. Tang and Vemuri [164] used the aiNet as a data preprocessing algorithm since aiNet is capable of reducing data redundancy and obtaining a compressed representation of the data. Each document is treated as an antigen in aiNet.

Principle component analysis (PCA) is also introduced in the proposed framework of Tang and Vemuri [164] to reduce the dimension of the vectors in the data after which the aiNet algorithm is applied to the compressed vectors. PCA resulted in a speedup of the compression of the data and further reduced the noise in the data. The result of the aiNet (compressed representation of the data) is then either clustered with K-means clustering or Hierarchical Agglomerative Clustering (HAC). Experimental results in [164] have shown that aiNet as a data preprocessing and compression algorithm obtained better clustering results (more compact clusters) than directly clustering the raw data with K-means clustering or HAC.

Another model which is based on the multipopulation aspect of aiNet is the proposed multi-objective multipopulation artificial immune network (MOM-aiNet) model by Coelho *et al.* [28]. Further investigation into MOM-aiNet led to an improved model, MOM-aiNet+ [29]. Contrary to aiNet, MOM-aiNet+ not only keeps the best individual of each subpopulation but several within each subpopulation. MOM-aiNet+ was applied to the biclustering problem which is a multi-objective optimisation problem. The biclustering technique is capable of finding several subsets (biclusters) in a data set in such a way that each subset contains patterns with a certain shared similarity [25, 80]. The quality of each bicluster can be measured by the volume of the bicluster (number of patterns $\times$ number of features) and the degree of similarity among the patterns within the bicluster. Both of these measurements need to be maximised. Furthermore, the number of patterns in a data set which is covered by the different biclusters and the degree of overlap between the biclusters also need to be measured.

Minor refinements to the aiNet model led to versions of the model which were respectively applied to the initialisation of centers of a radial basis function neural network [37] and the optimisation of multi-modal functions [33, 55]. The refined aiNet model is known as opt-aiNet. The dynamic opt-aiNet (dopt-aiNet) was recently proposed as an improvement to opt-aiNet for non-stationary environments [39, 40]. Both of these models are discussed next.

opt-aiNet: The aiNet model was adapted to solve multi-modal function optimisation problems and is known as *opt-aiNet* [33]. A few observations on the originally proposed opt-aiNet model in [33] were summarised in [167]. Some of the features of opt-aiNet listed are among others a dynamic population size, the capability to explore and exploit the search space and the capability to maintain multiple optima solutions [167]. Algorithm 4.10 lists the pseudo code of the opt-aiNet model with minor modifications as proposed in [167] (assuming minimisation of the objective) and is discussed next. The differences to the original model are also highlighted.

Algorithm 4.10: opt-aiNet Learning Algorithm

```
Randomly initialise a population of ALCs, $\mathcal{B}$, with size $\mathcal{B}_{init}$;
while stopping condition not true do
    Determine the fitness, $f$, of each ALC, $\mathbf{b}$, in $\mathcal{B}$;
    Normalise the fitnesses of population $\mathcal{B}$;
    repeat
        Generate $\eta$ ALC clones for each $\mathbf{b}$;
        Mutate each ALC clone proportionally to the normalised fitness of its parent ALC;
        Determine the fitness of each mutated clone, $\mathbf{b}'$, and select the mutated clone with
        the lowest fitness as $\mathbf{b}^*$;
        if $\mathbf{b}^*$ has a lower fitness than $\mathbf{b}$ then
            Replace $\mathbf{b}$ with $\mathbf{b}^*$;
        end
        Determine the fitness, $f$, of each ALC, $\mathbf{b}$, in $\mathcal{B}$;
        Normalise the fitnesses of population $\mathcal{B}$;
    until the difference in average fitness of $\mathcal{B}$ is less than a pre-defined threshold $\varepsilon_{fitness}$;
    Determine the network affinity between each pair of ALCs in $\mathcal{B}$;
    If the calculated network affinity is below the network suppression threshold $\varepsilon_{network}$,
    remove the ALC with the lowest fitness from $\mathcal{B}$;
    Add $\varphi$% (of the size of $\mathcal{B}$) of randomly generated ALCs to $\mathcal{B}$;
end
```

The network affinity between two ALCs is calculated as the Euclidean distance, as defined in equation (2.3). The fitness of an ALC, **b**, is calculated using the fitness function, f, which is the objective that needs to be optimised. An ALC clone is mutated proportionally to the normalised fitness, f^*, of its parent **b** as [33]

$$\mathbf{b}' = \mathbf{b} + \left(\frac{1}{\varsigma}\right) \exp\left[-\left(1 - f^*\left(\mathbf{b}\right)\right)\right] N(0,1) \ . \tag{4.42}$$

where $N(0,1)$ is a Gaussian random variable with zero mean and standard deviation of one. ς controls the decay of the inverse exponential function. The above mutation results in that highly fit ALCs are mutated less than less fit ALCs. Therefore poor ALCs are mutated more to explore the search space and good ALCs are mutated less to exploit the search space. The *stopping condition* of the *while*-loop is set to a maximum number of iterations, t_{max}.

The differences between the original opt-aiNet model and algorithm 4.10 are [167]:

- the assumption in the original model that the average fitness always decreases (assuming minimisation of the objective). The degree of similarity between the current average fitness and the previous average fitness is measured against a threshold value. A disadvantage of this assumption is that the degree of similarity will always be less than the threshold if the previous average fitness is greater than the current average fitness. This holds true even if there is a large difference between the previous and current average fitness. This drawback is addressed in algorithm 4.10 by calculating the difference between the previous and current average fitness and measuring the result against a similarity threshold value.

- the suppression of ALCs in the original model always results in the first ALC to be removed whenever the calculated network affinity (Euclidean distance) between two ALCs are below the network suppression threshold. A drawback of this approach is that the removal of an ALC is not based on its fitness, which can result in the removal of a potential optimum solution. Therefore the fitness of ALCs in algorithm 4.10 are first evaluated and the ALC with the worst fitness is removed.

In the context of data clustering as an optimisation problem, each ALC in the population represents a possible partitioning of the data set (similar to Clustering PSO as discussed in section 2.7.1). Thus, an ALC represents K centroids, one for each cluster. An ALC is defined as $\mathbf{b}_i = (\mathbf{c}_{i,1}, \mathbf{c}_{i,2}, \ldots, \mathbf{c}_{i,K})$ where $\mathbf{c}_{i,k}$ is the cluster centroid of the k-th cluster, $C_{i,k}$, represented by the *i*-th ALC in the population. The objective function that needs to be optimised is the quantization error as defined in equation (2.75) and is thus the fitness function of the ALCs, i.e. $f = J_{PSO}$.

Dynamic opt-aiNet: The dynamic opt-aiNet (dopt-aiNet) model was proposed by de França *et al.* [39, 40] and improved the opt-aiNet model to be more suitable for non-stationary environments. Two drawbacks of the opt-aiNet model are the large number of function evaluations to find good solutions and the possibility of an excessive increase in the size of the population over time. The recommended modifications in [40] to address these drawbacks and enhance opt-aiNet are discussed next.

In dopt-aiNet the population size is preset to a maximum. Whenever the size of the population reaches this maximum a percentage of the ALCs with the worst fitness are removed from the population. Another recommendation is to keep a separate memory population. The memory population contains ALCs which have not been replaced by their mutated clones for a certain peroid of time. Each ALC therefore needs to be initialised with a rank value. The rank value is incremented each time a mutated ALC clone replaces its parent ALC and decremented if not. An ALC is replaced by its mutated ALC clone if the mutated ALC clone improves the parent ALC's fitness. An ALC is moved to the memory population when the rank value reaches zero. The memory ALC then receives a new rank value which follows the same process. When the rank of a memory ALC reaches zero, it does not undergo any mutation.

The ς parameter for the Gaussian mutation in opt-aiNet sometimes require pre-analysis of the function landscape to be set properly. Small ς-values may lead to slower convergence whereas too large ς-values may lead to a mutated ALC clone which diverges even further from an optimum solution. The *golden section* technique in [13] is recommended to find the optimal value for ς. The golden section technique divides a search space into two and selects the interval with the best fitness. The selected interval is then again divided and the sub-interval with the best fitness is selected for division. The process of division is recursive and applied to the selected interval until the interval reaches a given length. Each interval is divided with the *golden ratio* which is found on many nature structures. The golden section is only guaranteed for continuous, convex and unimodal objective functions. Since the model has no prior knowledge of the objective function, the initial search interval is divided into four segments. The golden section technique is then applied to each of these segments. Algorithm 4.11 lists the pseudo code of the dopt-aiNet model with the recommended modifications and additional mutation operators as proposed in [40] (assuming minimisation of the objective) and is discussed next.

Two new mutation operators are proposed and applied to ALCs with a rank value that is greater than zero in the current ALC population and the memory population. These mutation operators are:

- One-dimensional mutation where each dimension of an ALC is individually mutated. This means that an N-dimensional ALC will generate N mutated clones, each clone mutated in exactly one dimension. Two additional ALC clones are also mutated in the direction of the unitary vectors $\mathbf{1}$ and $-\mathbf{1}$, respectively. Algorithm 4.12 lists the pseudo code for one dimensional mutation. Matrix $\mathbf{D}^{(N+2)\times N}$ contains the identity matrix of size N and two rows with the unitary vectors $\mathbf{1}$ and $-\mathbf{1}$, respectively. ς is calculated with the golden section technique.

- Gene duplication is inspired by the duplication of genes in nature whenever a chromosome is read [86, 144]. A dimension of an ALC is randomly selected and its value is copied into another randomly selected dimension if the fitness of the ALC is improved. Algorithm 4.13 lists the pseudo code for gene duplication mutation.

Another drawback of opt-aiNet is the network suppression threshold, $\varepsilon_{network}$. Since the Euclidean distance between ALCs determine the network affinity, some knowledge of the fitness landscape or pre-analysis should be done to adjust $\varepsilon_{network}$ to an optimal and appropriate value. The cell line supression technique is recommended as an enhancement to the Euclidean distance measure. Algorithm 4.14 lists the pseudo code for cell line suppression between two ALCs. The middle point, $\mathbf{p}_m$, of the line segment from $\mathbf{p}_i$ to $\mathbf{p}_j$ is calculated. The middle point is then projected onto the line segment to calculate the nearest point, $\mathbf{p}_{projection}$, to the line segment. If $\mathbf{p}_{projection}$ is inside the line segment then the network affinity is calculated as $\sigma(\mathbf{p}_m, \mathbf{p}_n)$, which means that the nearest point is at the point $\mathbf{p}_n$ where $\overline{\mathbf{p}_m\mathbf{p}_n} \perp \overline{\mathbf{p}_j\mathbf{p}_i}$. If $\mathbf{p}_{projection}$ is outside the line segment then the network affinity is calculated between $\mathbf{p}_m$ and $\mathbf{p}_i$ (in the case where $\mathbf{d}_m \otimes \mathbf{d}_j \leq 0$) or $\mathbf{p}_j$ (in the case where $\left|\mathbf{d}_j\right| \leq \mathbf{d}_m \otimes \mathbf{d}_j$).

Other Network Based Models: Gaspar and Collard proposed the *Simple Artificial Immune System* (SAIS) which is inspired by the network formation and adaptability of the immune system to foreign antigens [58], specifically the primary and secondary responses to foreign antigens (as discussed in section 3.3.3). SAIS is applied to problems within a binary space and therefore measures the affinity between an ALC and an antigen pattern using the Hamming distance (as

Algorithm 4.11: dopt-aiNet Learning Algorithm

```
Randomly initialise a population of ALCs, $\mathcal{B}$, with size $\mathcal{B}_{max}$;
while stopping condition not true do
    Determine the fitness, $f$, of each ALC, $\mathbf{b}$, in $\mathcal{B}$;
    Normalise the fitnesses of population $\mathcal{B}$;
    repeat
        Generate $\eta$ ALC clones for each $\mathbf{b}$;
        Gaussian mutate each ALC clone proportionally to the normalised fitness of its
        parent ALC;
        Determine the fitness of each mutated clone, $\mathbf{b}'$, and select the mutated clone with
        the lowest fitness as $\mathbf{b}^*$;
        if $\mathbf{b}^*$ has a lower fitness than $\mathbf{b}$ then
            Replace $\mathbf{b}$ with $\mathbf{b}^*$;
            Increment the rank value of $\mathbf{b}^*$ by one;
        end
        else
            Decrement the rank value of $\mathbf{b}$ by one;
        end
        if the rank value of $\mathbf{b}$ equals zero then
            Remove $\mathbf{b}$ from $\mathcal{B}$ and add to the memory population, $\mathcal{M}$;
        end
        Apply one-dimensional mutation on each ALC in $\mathcal{B}$ (Algorithm 4.12);
        Apply gene duplication on each ALC in $\mathcal{B}$ (Algorithm 4.13);
        Apply one-dimensional mutation on each memory ALC in $\mathcal{M}$ (Algorithm 4.12);
        Apply gene duplication on each memory ALC in $\mathcal{M}$ (Algorithm 4.13);
        for each memory ALC do
            if the memory ALC has improved after mutation then
                Increment the rank value of the memory ALC by one;
            end
            else
                Decrement the rank value of the memory ALC by one;
            end
        end
        Determine the fitness, $f$, of each ALC, $\mathbf{b}$, in $\mathcal{B}$;
        Normalise the fitnesses of population $\mathcal{B}$;
    until the difference in average fitness of $\mathcal{B}$ is less than a pre-defined threshold $\varepsilon_{fitness}$;
    Determine the network affinity between each pair of ALCs in $\mathcal{B}$ and suppress the ALC
    network using the cell line suppression as listed in algorithm 4.14;
    Add $\varphi$% (of $\mathcal{B}_{max}$) of randomly generated ALCs to $\mathcal{B}$;
    if $|\mathcal{B}| > \mathcal{B}_{max}$ then
        Remove $(\mathcal{B}_{max} - |\mathcal{B}|)$ ALCs with the worst fitness from population $\mathcal{B}$;
    end
end
```

Algorithm 4.12: One-dimensional Mutation Algorithm

```
Generate $n+2$ ALC clones for ALC $\mathbf{b}$;
for each ALC clone $\mathbf{b}^*$ do
    for each row $\mathbf{d}$ in matrix $\mathbf{D}$ do
        Generate a mutated clone $\mathbf{b}^{'}$ from $\mathbf{b}^*$ using $\mathbf{b}^{'} = \mathbf{b}^* + \mathbf{d} \times \varsigma$;
    end
end
Determine the fitness of each mutated clone, $\mathbf{b}^{'}$, and select the mutated clone with the
lowest fitness as $\mathbf{b}^*$;
if $\mathbf{b}^*$ has a lower fitness than $\mathbf{b}$ then
    Replace $\mathbf{b}$ with $\mathbf{b}^*$;
end
```

Algorithm 4.13: Gene Duplication Mutation Algorithm

```
Initialise gene to the value of a randomly selected dimension of ALC $\mathbf{b}$;
for each dimension n of ALC $\mathbf{b}$ do
    $oldval = \mathbf{b}_n$;
    $\mathbf{b}_n = gene$;
    if the fitness of ALC $\mathbf{b}$ does not improve then
        $\mathbf{b}_n = oldval$;
    end
end
```

UNIVERSITEIT VAN PRETORIA
UNIVERSITY OF PRETORIA
YUNIBESITHI YA PRETORIA

Algorithm 4.14: Cell Line Suppression Algorithm

```
Select two ALCs $\mathbf{b}_i$ and $\mathbf{b}_j$;
$\mathbf{p}_i = [\mathbf{b}_i, f(\mathbf{b}_i)]$;
$\mathbf{p}_j = [\mathbf{b}_j, f(\mathbf{b}_j)]$;
$\mathbf{p}_m = [\mathbf{b}_i + 0.5(\mathbf{b}_j - \mathbf{b}_i), f(\mathbf{b}_i + 0.5(\mathbf{b}_j - \mathbf{b}_i))]$;
$\mathbf{d}_j = \mathbf{p}_j - \mathbf{p}_i$;
$\mathbf{d}_m = \mathbf{p}_m - \mathbf{p}_i$;
$\mathbf{p}_{projection} = \frac{\mathbf{d}_m \otimes \mathbf{d}_j}{|\mathbf{d}_j|}$ (where $\otimes$ is the dot product);
if $\mathbf{d}_m \otimes \mathbf{d}_j \leq 0$ then
    $netaff = \sigma(\mathbf{p}_m, \mathbf{p}_i)$;
end
else if $|\mathbf{d}_j| \leq \mathbf{d}_m \otimes \mathbf{d}_j$ then
    $netaff = \sigma(\mathbf{p}_m, \mathbf{p}_j)$;
end
else
    $\mathbf{p}_n = \mathbf{p}_i + \mathbf{p}_{projection} \otimes \mathbf{d}_j$;
    $netaff = \sigma(\mathbf{p}_m, \mathbf{p}_n)$ (since $\overline{\mathbf{p}_m\mathbf{p}_n} \perp \overline{\mathbf{p}_j\mathbf{p}_i}$);
end
if $netaff < \varepsilon_{network}$ then
    Remove the ALC with the worst fitness;
end
```

discussed in section 4.3.1). The algorithm initially has a randomly generated population of ALCs which at each generation goes through three different phases to adapt to the training patterns (antigens). These phases are evaluation, cloning and recruitment, discussed next.

The evaluation phase differentiates the ALCs into exogenic activated or endogenic activated ALCs. Exogenic ALCs are activated by antigens in the current environment (at time t). Endogenic ALCs are not activated by the current environment and are equally reinforced by their individual densities. A number of the best exogenic ALCs are then selected in the cloning phase for cloning and mutation. The remaining endogenic ALCs are cloned without mutation. Mutated exogenic ALCs which do not improve the activation level of their corresponding parent exogenic ALCs are discarded.

In the final recruitment phase, the new population of ALCs consists of all the cloned endogenic ALCs and a selection of mutated exogenic ALCs. The latter selection process is based on a tournament selection where a number of mutated exogenic ALCs challenge each ALC in the previous generation's population.

The SAIS model is applied to time dependent optimisation problems to test the adaptability of the model in non-stationary environments. SAIS is able to track changing optima as well as memorising previously encountered optima. Scenarios do occur where previously encountered optima are forgotten. Thus the memory of SAIS is unstable and the authors proposed *Yet Another SAIS* (YASAIS) as an enhancement [58].

In YASAIS the population of ALCs are partitioned into sub-populations. In YASAIS there are no endogenic ALCs. Instead, the remaining ALCs (non-exogenic) are preserved within their respective sub-populations to the next generation. A single exogenic ALC is selected from each sub-population which goes through the same cloning and recruitment phase as in SAIS. The YASAIS did not deliver the expected improved results and was further enhanced in [59].

4.7 Idiotypic Network Topologies

The formation of idiotypic networks between lymphocytes (or their corresponding antibodies) can be defined by different network topologies. In the preceding section on network based artificial immune systems, the network interaction or network formation between artificial lymphocytes is either determined by a proximity matrix of network affinities or the grouping of similar artificial lymphocytes in sub-networks. The former is normalised with a network affinity threshold to determine the network links between the artificial lymphocytes and the latter utilises a clustering algorithm. There are, however, alternative and less familiar network topologies to determine the possible interactions in an idiotypic network of lymphocytes. The different theoretical approaches to determine the possible interactions in an idiotypic network are discussed next. Each of these network topologies specifies the interconnections between lymphocytes and the binding strength of these connections.

The Linear Topology: Lymphocytes in the linear topology are positioned as a sequence of different idiotypic levels of interaction. The linear topology was introduced by Richter and proposed as a *chain-reaction* between lymphocytes at different idiotypic levels [152, 153]. Figure 4.4 illustrates the linear topology of an idiotypic network with l idiotypic levels.

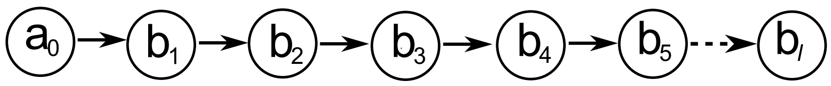


Figure 4.4 Linear Network Topology

The antigen, $\mathbf{a}_0$, is positioned at idiotypic level 0. Lymphocytes, $\mathbf{b}_i$, at idiotypic level i interact with lymphocytes at idiotypic levels $i-1$ and $i+1$ by either stimulating or suppressing neighbouring lymphocytes in the sequence. In figure 4.4, lymphocytes in idiotypic layer i stimulate the lymphocytes in layer $i+1$, which in turn stimulate the lymphocytes in layer $i+2$ and so forth. Lymphocytes in an idiotypic layer also suppresses the layer of lymphocytes responsible for its stimulation. Thus, suppression between idiotypic layers follows a *chain-reaction* in the reverse order to that of stimulation between layers. Lymphocytes in idiotypic layer l suppress the lymphocytes in layer $l-1$, which in turn suppress the lymphocytes in layer $l-2$ and so forth.

The Simple Cyclic Topology: Hiernaux discovered that the dynamical behaviour of the linear topology is dependent on whether l is odd or even [84]. Hiernaux converted the linear topology into a cyclic topology, as illustrated in figure 4.5.

The Affinity Matrix Topology: Figure 4.6 illustrates an example of an affinity matrix. Each element in the matrix specifies the interaction between two lymphocytes, while the magnitude of the element specifies the strength (binding affinity) between two lymphocytes.

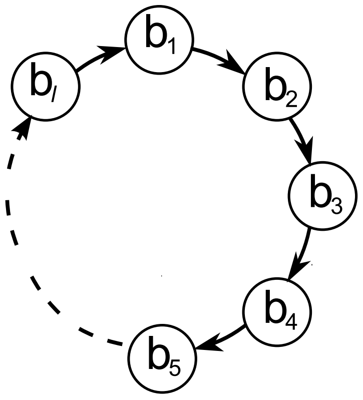
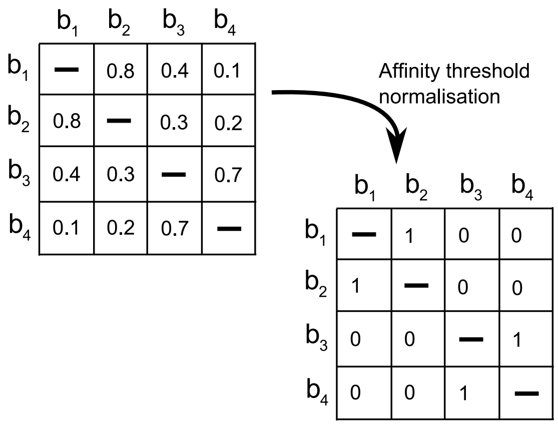


Figure 4.5 Simple Cyclic Network Topology


Affinity threshold normalisation

	b_1	b_2	b_3	b_4
b_1	—	0.8	0.4	0.1
b_2	0.8	—	0.3	0.2
b_3	0.4	0.3	—	0.7
b_4	0.1	0.2	0.7	—

	b_1	b_2	b_3	b_4
b_1	—	1	0	0
b_2	1	—	0	0
b_3	0	0	—	1
b_4	0	0	1	—

Figure 4.6 Affinity Matrix Topology with Normalisation

Thus, for a set of numbered lymphocytes, the element at position $(1,2)$ in the affinity matrix, specifies the binding affinity between lymphocyte number 1 and lymphocyte number 2 in the set [18]. Each of the elements in the affinity matrix can be measured against an affinity threshold value, normalising the affinity value of each element into either 0 or 1. A value of 1 implies idiotypic interaction between the lymphocytes and a value of 0 implies no interaction.

The Cayley Tree Topology: A Cayley tree is a loop-less tree. The node which contains the lymphocyte with the highest affinity with an antigen forms the root node of a Cayley tree [149, 181]. Only the root node, $\mathbf{b}_1$, reacts to the antigen, $\mathbf{a}$. The number of idiotypic network connections, z, is the number of connected neighbours for each node. z determines the complexity of the Cayley tree topology. Figure 4.7 illustrates the Cayley tree topology for an idiotypic network of lymphocyte nodes with $z = 3$ [181].

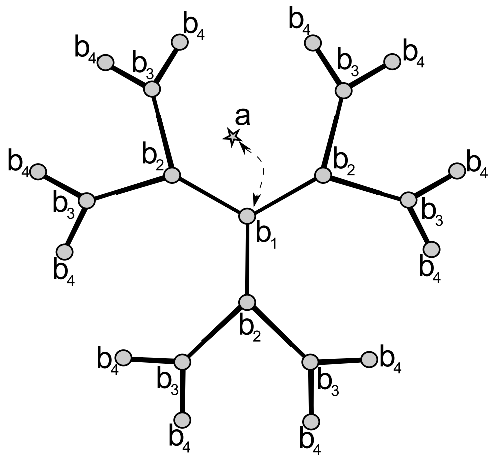


Figure 4.7 Cayley Tree Network Topology

Connectivity between the root node and the remaining lymphocyte nodes are determined with the affinity matrix topology (as discussed above). The distance between the root node and each of the connected lymphocyte nodes determines the idiotypic level for each of the connected lymphocyte nodes. Thus, with the root node at idiotypic layer 1 ($\mathbf{b}_1$), connected lymphocytes in idiotypic layer 2 ($\mathbf{b}_2$) have a closer distance to $\mathbf{b}_1$ than lymphocytes in idiotypic layer 3 ($\mathbf{b}_3$). The lymphocytes in $\mathbf{b}_3$ have a closer distance to $\mathbf{b}_1$ than lymphocytes in $\mathbf{b}_4$, etc. [149]. Therefore, the nodes are organised in a hierarchical manner, based on the measured distance to the root node. The total stimulation received by lymphocytes within a node at idiotypic layer i, is defined as [181]

$$\vartheta^i = \nu^{(i-1)} + (z-1)\,\nu^{(i+1)} \tag{4.43}$$

where ν^j denotes the concentration of lymphocytes in idiotypic layer j.

4.8 Danger Theory Models

The classical view of natural immunity is able to distinguish between *self* and *non-self* cells (as discussed in section 3.1). This ability is realised through the maturation process of T-Cells to become self-tolerant. In contrast to the classical theory, the danger theory further distinguishes the *non-self* cells as *dangerous* or *non-dangerous* (as discussed in section 3.7). The danger theory considers a cell to be *dangerous* if the cell instigates a danger signal of necrotic cell death to activate the antigen presenting cells. The activated antigen presenting cells co-stimulate the mature T-Cells, which in turn stimulate the B-Cells to react to the *foreign* cell. One of the motivations for the danger theory is that the natural immune system is able to adapt to a changing *self*, since the natural immune system only reacts to *dangerous non-self* cells. This inspired the modelling of the danger theory in AIS.

The main difference of the danger AIS models to those AIS models inspired by the classical view (as discussed in section 4.4), is the inclusion of a *signal* to determine whether a *non-self* pattern is *dangerous* or not. Therefore, a danger AIS model needs to define a signal of *death* or *danger* which is problem specific. The remainder of this section briefly discusses some of the applications of the danger AIS models to highlight the significance of the *danger* signal.

A familiar application of classical AIS models is in the field of network intrusion detection [53, 111, 114]. Intrusion detection AIS models create profiles of the *normal* incoming traffic at different nodes in the network. These *normal* profiles are seen as *self*. Through the application of the negative selection technique, abnormal traffic detectors (self-tolerant detectors) are generated from these *normal* profiles. The incoming traffic at each node is then monitored and the model signals an alarm of intrusion whenever an abnormal traffic detector is activated. A major drawback to this kind of traffic profiling is the assumption that *normal* traffic patterns never change. The fact that *normal* traffic patterns do change over time, results in outdated profiles with obsolete detectors. Therefore, an intrusion detection system needs to be adaptable.

An alternative approach to adapt to changes in *normal* traffic flow is to only signal an alarm of intrusion when the monitored host senses *danger*. In this context, *danger* can be defined as the sensing of any abnormal CPU load, memory usage, excessive I/O reads and writes, or security attacks. Whenever an abnormal traffic detector is activated without a danger signal from the host, the profile of *normal* traffic is adapted to accommodate the detected *normal* traffic pattern, resulting in an adaptive intrusion detection system. Danger AIS models as adaptive intrusion detection systems are proposed in [2, 4].

In a network with a dynamic topology, a change in *normal* traffic can also occur whenever a node is removed or added to the network, or when a node *misbehaves*. A mobile ad-hoc network is an example of such a network. A mobile ad-hoc network consists of terminal nodes, each with a radio as communication device to transmit information to other terminal nodes in the network, i.e. no infrastructure between nodes. Thus, nodes not only function as terminals, but also as relays of the transmitted information in the network. A node can misbehave whenever the node does not relay information to neighbouring nodes, or the node experiences hardware failures, or malicious software (like viruses) on the node try to overthrow the network. Sarafijanovic and Le Boudec proposed a danger theory inspired AIS to detect misbehaving nodes in a mobile ad-hoc network [155]. In this context a misbehaving node is dangerous.

Each node monitors the traffic from neighbouring nodes as *normal/self* and generates self-tolerant detectors using negative selection. Whenever a detector detects an incoming self pattern from a neighbouring node, the detector is replaced by a newly generated self-tolerant detector. Each node keeps a buffer of incoming traffic patterns from neighbouring nodes and self-tolerant detectors are frequently generated from the buffer. If a source node experiences danger (misbehaving node due to packet loss), the source node will generate an observation with a danger signal along the route where the packet loss was experienced. The action taken by the neighbouring nodes is to discard the observation from the buffered observations through correlation with the danger signal (also observed). This prevents the generation of detectors on *non-self* observations.

A similar buffering approach of *normal* patterns is taken in [158]. The danger theory inspired AIS by Secker *et al.* [158] simulates an adaptive mailbox. The proposed AIS classifies *interesting* from *uninteresting* emails. Initially, the user's actions on the incoming mail is monitored. If an email is deleted by the user, a detector is generated to detect the deleted email and added to a set of detectors. After adding a new detector to the set, the existing detectors in the set are cloned and mutated to improve the generalisation of the set. Thus, the set represents *non-self*/uninteresting email. The process continues until the size of the detector set reached a certain maximum.

The detector set adapts to the changing behaviour patterns of the user by buffering deleted emails as a set of *non-self* emails. As soon as the buffered set reaches a specific size, it is represented to the detector set of uninteresting emails. The detector set adapts to the buffered set through clonal selection.

Danger in this model is defined and measured as the number of unread emails in the inbox. Danger is signalled when the number of unread emails reaches a limit. When the model receives a danger signal, the unread emails are presented to the set of detectors for detection of *uninteresting* emails. The *uninteresting* classified emails are then moved to a temporary folder or deleted.

Table 4.1 List of Immunological Terms Mapped to Data Analysis Terms

Immunology	**Data Analysis**
Antigen set	Data set of patterns that needs to be clustered
ALC population	Clustered data set
Affinity	Measure of (dis)similarity between patterns
Network of ALCs	Cluster of patterns
Mean vector of ALC network (or representative ALC in ALC network)	Centroid of a cluster

4.9 Conclusion

This chapter highlighted the basic components of an AIS model. These components were categorised within an AIS framework. These categories are the representation of an ALC and antigen structure within a search space, the interaction between these structures (affinity measures), and the adaptation of the ALC structures through a selection strategy. The categories of the framework were then discussed with reference to proposed theoretical AIS models.

The chapter continued with an overview of the *shape space* model wherein the structure of an ALC and/or antigen can be defined and represented. In order to determine the affinity between the structure representing an ALC and the structure representing an antigen in *shape space*, the chapter gave a discussion on the different affinity measures within different *shape spaces*. Three of the most familiar affinity matching rules in a binary *shape space* were highlighted with their drawbacks of *holes*. After the discussion of affinity measures, the different selection strategies to adapt the ALCs structures were discussed.

Each selection strategy gave an overview of existing AIS models based on the specific strategy with a more detailed overview of AIS models which are based on the idiotypic network formation strategy. The discussion of the idiotypic network formation also introduced different theoretical approaches/network topologies to determine the possible interactions in an idiotypic network.

Table 4.1 provides a list of immunological terms which are mapped to data analysis terms. In the context of data clustering, the data set that needs to be clustered by an AIS model is seen as the set of antigen patterns. The clustered data set is represented by the population of ALCs. The measure of (dis)similarity between feature vectors (as discussed in section 2.2) determines the affinity between an ALC and an antigen pattern or another ALC (in the case of network based AIS models). In network based AIS models, each ALC network is a potential cluster in the data set (antigen set). The centroid of the cluster (ALC network) is calculated as the mean vector of ALCs or is a representative ALC in the ALC network.

The next chapter proposes and presents a novel network theory inspired artificial immune system.