The Crossover Closure and Partial Match Detection

Fernando Esponda, Stephanie Forrest, and Paul Helman

Department of Computer Science
University of New Mexico
Albuquerque, NM 87131-1386
{fesponda, forrest, helman}@cs.unm.edu

Abstract. In this paper we explore the crossover closure generation rule as a generalization scheme for artificial immune systems using partial match detection. The paper reviews previous results and extends the previously introduced notion of crossover closure to encompass additional matching rules. For concreteness, the discussion focuses on r-chunks matching, giving alternative ways that detectors can be used to implement the crossover closure.

1 Introduction

Much of the success of artificial immune systems (AIS), and other learning systems, depends on the choice of representation. In AIS, it is common to represent information using a population of detectors, analogous to immune cells in the body. A matching criterion is typically defined that determines for any given detector how closely it matches a certain data item. This is analogous to receptor/ligand binding in the immune system. We refer to this as *partial match detection*, that is, detection of a concept class based on partial matching rules. In this paper, we focus on a particular class of match rules, which have formed the basis of several earlier AIS implementations. We are interested in the properties of the match rule itself, independently of how it is employed in an immune system framework. For example, many of the results we present apply both to *positive detection* and *negative detection* systems, and some of the results can be used to examine tradeoffs between these two detection schemes. In an earlier paper, we developed an extensive formal framework for analyzing positive and negative detection under various matching criteria [12]. Much of this paper is devoted to summarizing those earlier results, illuminating their importance for artificial immune systems and extending them to cover a more general set of conditions.

The paper adopts some terminology from the machine learning literature, and accordingly we describe the problem addressed by the paper as follows: Given a collection of instances drawn from some concept class, where the instances are represented as attribute vectors or strings, we wish to derive some means for inductively determining the underlying concept (or in the case of negative detection, the counter concept(s)). The resulting characterization must be flexible enough to accommodate both concept drift and distributability.

Although the paper adopts a machine learning perspective, our approach draws its inspiration from an important component of the immune system. In the immune system a collection of T-cells are deployed throughout the body to monitor the well-being of an organism. T-cells are equipped with receptors able to bind peptides on the surface of cells, and if such a binding occurs an immune response may be initiated. T-cells are subjected to a selection process, before they are released into the body, which ensures they can recognize only nonself peptides. Analogously, the proposed system is comprised of a set of *detectors*, the counterpart of T-cells. Whenever a new string (instance) is observed a detector determines if the information it contains refers to it or not, i.e., if they bind. This said, we are interested in establishing a scheme whose generalization is readily understood and analyzed. In doing this we will be in a better position to establish mechanisms and policies to control it. In this paper we set forth the crossover closure generation rule as a learning paradigm and review a match rule and two detection schemes that implement it.

In the following sections, we first give a brief overview of related work. We then define a small extension of the crossover closure, giving an example and some motivation. We then review several variants of contiguous bits matching rules, introducing two new variants. In section 5, we summarize the detection schemes most relevant to the paper, and in section 6, we describe some of the interesting properties of the various r-chunks decompositions. Section 6.1 reviews our earlier results on overlapping fixed-size windows, and section 6.2 gives new results on nonoverlapping fixed size windows. We then discuss the significance of these results and conclude with a summary of the paper.

2 Related Work

The r-chunks match rule, on which our paper focuses, was introduced in [3, 12]. It is a simplification of the r-contiguous bits (rcb) match rule [13, 28, 29], which has been used in many artificial immune system projects. Modeling projects incorporating this rule include [13, 28, 29, 14], and application projects using the rule include [15, 8, 18, 26, 5–7, 30, 2]. Many formal studies of immune algorithms are based on systems employing r-contiguous bits, e.g. [10, 9, 32, 31, 33].

The problem discussed in this paper, broadly stated, has been studied in the machine learning and statistical learning literature, often under the term "one-class learning." In [23, 24] some theoretical results for learning without counter-examples are given. A review of unsupervised learning, of which many AISs are an example, is presented in [22]. Ref. [1] discusses a variety of instance-based learning algorithms, and a survey of on-line learning algorithms can be found in [4]. Finally a comprehensive explanation of the theoretical underpinnings of classification and learning in general, can be found in [11, 16, 27, 25].

3 The Crossover Closure

Consider a sample S (for *self*) of strings taken from a concept class RS (real self) which we wish to characterize. We adopt a simple categorical division into "similar to S" versus "dissimilar from S" and distinguish these categories by means of a *generation rule* which attempts to characterize the underlying set RS from which S is likely drawn. It has been shown experimentally that this interpretation of the detection task often captures sufficient detail of process behavior to provide effective detection [20, 19].

Definition: A generation rule Q is a mapping from a set S of length l strings to a set $Q(S)$ of length l strings containing S.

We focus our attention on a generation rule that is both simple to analyze and intuitively appealing, the crossover closure. The crossover closure was introduced in [12], where it was restricted to contiguous *windows* of attributes. Here, we remove this restriction and extend it to a set of features. The basic idea of the crossover closure is as follows: Given examples of some concept class represented as a vector of features, the crossover closure is a hypothesis stating that only some combinations of the observed features define instances of the concept class.

Consider a simple example of the concept *vehicle*, exemplified by the following instances (S), where each row of the table corresponds to a single instance:

Wheels	Color	Max. Speed (mph)
4	red	100
2	black	200

Under the crossover operator the following are also valid instances of the class vehicle:

Wheels	Color	Max. Speed (mph)
4	black	100
4	red	200
4	black	200
4	red	100
2	black	100
2	red	200
2	black	200
2	red	100

Formally, given a set S of strings, and a fixed $1 \leq r \leq l$, the crossover closure $CC(S)$ of S is defined in terms of its features as:

$$CC(S) = \{u \in U | (\forall \textit{features}\ w)(\exists s \in S)\ u[w] = s[w]\} \tag{1}$$

where $u[w]$ is the projection of string U onto feature w. In words, a string u of length l is in the crossover closure of S if and only if each of u's features exactly matches the corresponding feature of some member of S. In the above example, instances of the concept vehicle are represented as strings with three features: Wheels, Color and Max. Speed. A string presenting any combination of the sampled values for this features is part of the crossover closure. When S is such that $CC(S) = S$ we say that S is *closed* under crossover closure. We understand a feature to be any combination or function of attributes of the instance vectors, and not necessarily a specific attribute. This broader interpretation of "feature" is the sense in which this definition of crossover closure extends our earlier definition.

An interesting motivation and justification for this rule stems from the similarities it has with some well known relational database operations [12]. The join operator and the crossover closure are closely related, under some partial match rules, leading us to believe that the crossover closure may be a useful characterization for many practical data sets.

The name "crossover closure" is partially inspired by the crossover operation in genetic algorithms (GA) [21]. However, these notions do not exactly correspond as the crossover discussed here depends on what a feature is defined to be. For the example presented above, in which features are non-overlapping the crossover closure defines all possible strings that can be generated using the crossover operator alone, in a GA, from a given population S. However, other decompositions such as the one discussed in section 6.1 don't have such a clear correspondence with the traditional one point (or multi-point) crossover operator, in the referenced case the crossover closure is a proper subset of the possible strings generated with the GA's crossover operator.

4 R-chunks matching

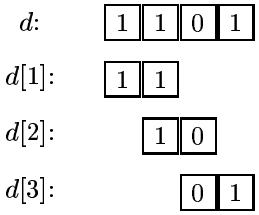
As outlined in Section 1, an AIS is typically composed of a collection of detectors or agents. The detectors are independent in the sense that each binding event is determined locally by each detector on its own, even if the classification of a string is ultimately resolved by some combination of matches. Thus, we are interested in match rules between detectors and data than are local, but which correspond in the aggregate to the crossover closure.

Further, a detector should be as simple as possible since we want its operation to be efficient in generation time, storage requirements, and run-time (cost of determining a match). Recall from the above that the instances from which the learning process is to take place are represented as attribute vectors or strings. In this context, a natural way to represent a detector is also as a string and to use string matching as the method of detection. In what follows, we give examples of a class of partial matching rules, some of which correspond to the crossover closure generation rule. The variations appear to be interesting because they open the possibility of more efficient algorithms for some AIS applications.

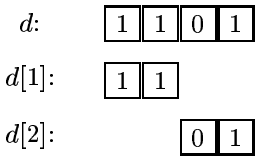
All of the rules studied in the following sections are derived from the r-contiguous bit matching rule (rcb) introduced in [28, 29, 17]. A detector under rcb is a string of length l, and is said to match another string, of the same length, if it has at least r consecutive bits in common. One simplification of rcb, which corresponds exactly to the crossover closure, is known as *r-chunks*. In r-chunks matching, only r contiguous positions are specified, rather than fully specifying all l attributes of a string. Thus, an r-chunks detector can be thought of as a string of r bits, together with its starting position within the string, known as its *window*. An r-chunks detector d is said to match a string x if all the symbols of d are equal to the r symbols of x in the window specified by d. More formally, if d is an r-chunk on window w, the matching rule considered is:

$$dMx \leftrightarrow x[w] = d.$$

where dMx denotes that detector d matches string x and $x[w]$ is the projection of string x onto window w. Therefore, an r-contiguous bit detector can be decomposed into $l - r + 1$ overlapping r-chunks detectors, as the following figure illustrates. Let $d = 1101$ be an r-contiguous bit detector with l=4, r=2 and $d[1], d[2], d[3]$ the corresponding r-chunk detectors:



Interesting variants of the *overlapping window* r-chunks rule can be constructed, and we are currently exploring such variants and their properties. For example, we can require that the windows not overlap (the *nonoverlapping* variant) as shown in the following figure.



A second variant can be constructed by relaxing the assumption that r is fixed, although we do not study it in this paper.

It was shown in [12] that the r-chunks and the rcb match rules are not equivalent in terms of the languages they recognize. However, the result depends on the details of how the match rules are deployed, a topic addressed in the next section. In particular, under certain detection schemes (section 5), it was shown that the r-chunks match rule corresponds exactly to the crossover closure and that rcb rules do not. To simplify the discussion of the crossover closure (eq.1), we focus on the special case when a feature is understood to be the r attributes encompassed by a window, and we consider the overlapping and nonoverlapping variants in sections 6.1 and 6.2 respectively.

5 Detection Schemes

In [12], a taxonomy of detection schemes was given in terms of the languages—the set of strings—that a detection scheme is able to recognize. This taxonomy is constructed along two dimensions. The first specifies whether detectors are tailored to match strings in self or nonself, denoted as P or N respectively. Many AIS systems use the idea of *negative detection*, in which a set of detectors is generated that recognizes the complement of self (known as nonself). Various claims have been made by various research groups about the efficacy of this detection scheme, and in particular, how it compares with a more traditional *positive detection* approach to pattern recognition, in which a representation of the positive instances is stored explicitly.

The second dimension specifies how many matches are required to determine the membership of a string. We considered two options, one in which a single match suffices and a second in which we require a detector to match in each window. We refer to these as disjunctive matching (D) and conjunctive matching (C) respectively, although we note the possibility of intermediate schemes. We restrict our attention to two such schemes, e.g., a scheme in which a match requires only a given fraction of the windows to match. Let Υ be a set of r-chunk detectors:

- Negative Disjunctive Detection $Scheme_{ND}$:
  $Scheme_{ND}(\Upsilon)$ is the set of strings x in U such that $(\forall windows\ w)(\nexists d \in \Upsilon)(dMx)$.
- Positive Conjunctive Detection $Scheme_{PC}$:
  $Scheme_{PC}(\Upsilon)$ is the set of strings x in U such that $(\forall windows\ w)(\exists d \in \Upsilon)(dMx)$.

Consider, for instance, the following sets of positive detectors: $\Upsilon_1 = \{01, 11\}$ detectors meant to match the first two bits in a string and $\Upsilon_2 = \{10, 11\}$ detectors for the second and third bits. Under $Scheme_{PC}$ only strings that are matched by a Υ_1 detector in their first window *and* a Υ_2 detector in their second window will be considered part of the language thus $Scheme_{PC}(\Upsilon_1 \cup \Upsilon_2) = \{010, 011, 110, 111\}$. Now consider the case where

Υ_1 and Υ_2 are negative detectors, then a string matched in its first window by a Υ_1 detector *or* in its second window by a Υ_2 detector will be considered outside the language: $Scheme_{ND}(\Upsilon_1 \cup \Upsilon_2) = \{000, 001, 100, 101\}$.

The class of languages recognized by $Scheme_{ND}$ and $Scheme_{PC}$ are identical, and are exactly the class of sets closed under crossover closure when r-chunks matching is used [12].

6 R-chunks Decompositions

In this section we discuss two decompositions based on the r-chunks match rule under the detection schemes of the previous section. We will examine the overlapping and nonoverlapping variants in terms of the size of their detector sets and the size of the generalization they induce (the crossover closure).

6.1 Overlapping Fixed Size Windows

In the following two sections we summarize some properties of $Scheme_{PC}$ and $Scheme_{ND}$ under the r-chunks match rule when a sliding window decomposition is used. Given a set of instances S it is straightforward to compute exactly how many distinct detectors can be generated. For $Scheme_{PC}$, it requires counting the number of distinct patterns for each of the t windows that comprise the strings in S, whereas for $Scheme_{ND}$, enumerating the distinct patterns that are not present in each window will result in the number of detectors. We are interested in how the number of detectors behaves as a function of the window size r and the number of training instances. In the following, we consider the case when S is a random, uniformly generated collection of strings defined over some finite alphabet $\mathbb{A}$. In such a scenario, the expected number of positive detectors E_{pos} and the expected number of negative detectors E_{neg} are given by:

$$E_{pos} = tE_r \tag{2}$$

$$E_{neg} = t(\mathbb{A}^r - E_r) \tag{3}$$

where $E_r \approxeq \mathbb{A}^r - \mathbb{A}^r(1 - \mathbb{A}^{-r})^{|S|}$, $t = l - r + 1$ is the number of windows.

With these formulas it is possible to determine when one scheme is beneficial over the other with regards to the number of detectors they require. For this purpose, it suffices to compute the number of strings in S for which both schemes yield the same number of detectors (i.e., when $E_{pos} = E_{neg}$), and note that a sample smaller than this value will require fewer detectors for the positive scheme, and fewer negative detectors if the sample exceeds it. Both schemes have an equal number of detectors when:

$$|S| \approx (0.693)\mathbb{A}^r \tag{4}$$

Note that this value depends only on the choice of r and the cardinality of the alphabet and not on the actual string length.

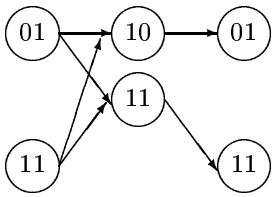
A detector set generated in this manner is highly likely to be redundant. That is, it will likely contain some detectors whose removal would not change the recognized language recognized. Redundancy might be a desirable feature, especially if detectors are to be distributed. However, it is important to understand how much redundancy is present and address the related question of finding an efficient detector set. Intuitively, redundancy arises from the fact that for some strings a match in window i implies a match in window $i + 1$. For example, for $l = 3$ and $r = 2$ consider the following sample $S = \{000, 101\}$. The implied generalization consists of strings $CC(S) = \{000, 101, 001, 100\}$. Clearly, the positive detectors for window 1, $\{00, 10\}$, match every string in the closure, and only such strings. Thus, it is unnecessary to check for a match in window 2. It is easily verified that the same holds if negative detectors are used, the case most relevant to AIS. The following two equations eliminate this source of redundancy from the detector set when an binary alphabet is used:

$$E_{minN} = 2^r - E_r + (l - r)(E_r - 2(E_r - E_{r-1})) \tag{5}$$

$$E_{minP} = E_r + (l - r)(E_r - 2(E_r - E_{r-1})) \tag{6}$$

The size of the sample S for which E_{minN} and E_{minP} yield the same number of detectors is the same as with the full repertoire eq.(4).

The Crossover Closure and its Expected Size In order to examine the size of the crossover closure as a function of the sample size $|S|$ and window size r, a similar assumption is made as in the preceding section, namely we consider the case of a random sample of strings over a binary alphabet. This problem can be mapped into a graphical representation, a directed acyclic graph (DAG), where each level contains nodes corresponding to the positive detectors derived from S for each of its windows. Nodes in consecutive levels are connected if the detectors to which they correspond overlap (crossover) i.e., this is if they match in their common $r-1$ positions. Take, for instance, a self set S comprised of the following two strings $S = \{0101, 1111\}$ with $l = 4$, $r = 2$. Its DAG:



In this example, where $r = 2$, there is only one bit position that overlaps between adjacent windows and whose symbols must match for the corresponding nodes to be connected. Thus, nodes whose label ends with symbol 1 are connected to nodes in the next level with label starting with symbol 1, and nodes that end in 0 are connected to nodes that start with 0. Under this representation, the crossover closure is exactly the set of strings formed by traversing the graph form level 1 to level t (the number of windows) $CC(S) = \{0101, 1111, 0111, 1101\}$.

In order to compute the number of strings in the closure, note from the DAG that the number of paths departing from a given node doubles if it has two outgoing edges, i.e., if the corresponding detector crosses-over with two detectors in the following window. The closed form solution for the expected number of paths in such a graph and hence the size of the crossover closure is given by:

$$CC(t) = E_r(\bar{o})^{(t-1)} \tag{7}$$

where E_r is the expected number of positive detectors given by eq. 2, $\bar{o}$ is the expected outdegree of each node[1] and t is the number of windows.

The size of the generalization, $CC(S)$, can be determined by the size of S or by the size of the detector set (obtaining an estimate for $|S|$ from either eq. 3 or eq. 2 and substituting in eq. 7). It is important to note that the actual size will depend on the structure of the specific self set. Nevertheless, the analysis provides insight into its behavior and enables us to ascertain the impact of allowing novel strings into the sample. This can be useful for determining, in a dynamic scenario, when (or at what rate) detectors should be added or deleted from the working set.

6.2 Nonoverlapping Fixed Size Windows

As noted in section 4 the r-chunks match rule does not necessarily require the use of a sliding window nor is it restricted to a fixed window size. As an illustration of another matching scheme for which the crossover closure is the characteristic generalization, we discuss an alternative decomposition based on the r-chunks match rule.

This decomposition restricts detector creation to non-overlapping windows but maintains a fixed size for each window. For simplicity we assume that r exactly divides l. In the case of positive detection, all distinct patterns in non-overlapping windows are potential detectors as are all absent patterns potential negative detectors. It is straightforward to see that the generalization implied by this design is the crossover closure of the strings in the training sample S (according to eq.1). In fact its size is easier to compute than with the sliding window model and is simply the product of the number of detectors for each window:

$$CC(S) = \prod_i |\Upsilon[i]| \tag{8}$$

[1] For an estimate of the out degree see [12].

where $\Upsilon[i]$ is the set of detectors for the ith window.

Likewise the number of detectors can be obtained as the sum of the number of detectors for each window:

$$\sum_i |\Upsilon[i]| \tag{9}$$

If we consider, as we did above, a random sample of strings S, the expected number of detectors is simply tE_r for positive detectors and $t(\mathbb{A}^r - E_r)$ for negative detectors, where E_r is given by eq.2 and $t = \frac{l}{r}$ is the number of windows. Interestingly, the tradeoff point between negative and positive detectors is the same as with the sliding window model (eq.4). Lastly, the number of strings encompassed in the generalization follows directly from eq.8: $CC_r(S) = (E_r)^t$. Figure 1 plots the size of the crossover closure of the two decompositions reviewed in this paper.

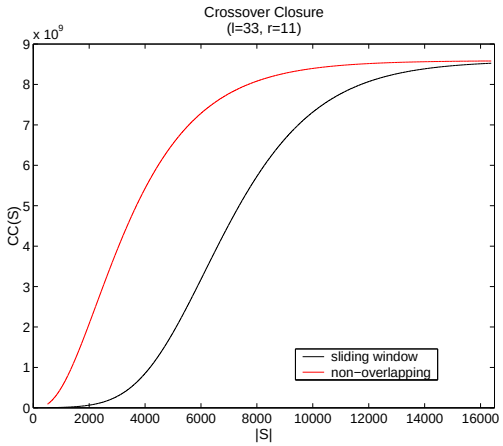


Fig. 1. The crossover closure as a function of the size of a random sample of binary strings for both the sliding window and non-overlapping decompositions.

7 Discussion

It is difficult if not impossible to determine in general what the best generation rule is and how best to represent a detector that implements it. However, having a clear understanding of what the generalization is allows us to exert more precise control over its performance, balancing its benefits and shortcomings. The results presented in this paper still leave some questions unanswered. In particular, how is r to be determined? Although the above analysis might provide some guidelines for setting this parameter in terms both of the size of the detector set and the size of the generalization, implicit in this rule is the prior assumption that important attributes are contiguous within the string. In order to improve and adjust classification performance as well as the space requirements, it will be necessary to define a mechanism that is able to change the relative position of attributes within the string and an algorithm for finding suitable arrangements. One such mechanism, permutation masks, has been proposed before but was used as a means of reducing the size of the generalization [12, 17]. However, this mechanism naturally alters the contiguity of bits and hence can be used for finding better decompositions. What remains is an algorithm for finding a good permutation. This is the focus of our current research.

These are only the first steps toward building an AIS with a firm formal footing. It remains to establish what are the policies and mechanisms that will permit the tracking of a shifting self and the distribution of detectors. These are important issues as they are some of the potential benefits that an immune inspired design might have over other paradigms.

8 Conclusion

In this paper we explored the crossover closure generation rule as a learning model for artificial immune systems. We discussed two detection schemes, together with a match rule that implement this generalization. A summary of theoretical results regarding the partial matching rule described in this work was presented, including results for the size of the detector set and the size of the generalization as a function of its parameters and the sample size. We covered a positive detection scheme for which detectors are customized to match instances of self and a negative detection scheme where they match instances of nonself. We reviewed how both schemes partition the instance space equivalently and some of the tradeoffs involved in using one scheme over another.

Acknowledgments

The authors gratefully acknowledge the support of the National Science Foundation (grants ANIR-9986555 and DBI-0309147), the Office of Naval Research (grant N00014-99-1-0417), Defense Advanced Projects Agency (grant AGR F30602-00-2-0584), the Intel Corporation, and the Santa Fe Institute. F.E. thanks Consejo Nacional de Ciencia y Tecnología (México) (grant 116691/131686) for its financial support.

References

1. D.W. Aha, D. Kibler, and M.K. Albert. Instance-based learning algorithms. *Machine Learning*, (6):37–66, 1991.
2. M. Ayara, J. Timmis, R. de Lemos, L. N. de Castro, and R. Duncan. Negative selection: How to generate detectors. In J Timmis and P J Bentley, editors, *Proceedings of the 1st International Conference on Artificial Immune Systems (ICARIS)*, pages 89–98, University of Kent at Canterbury, September 2002. University of Kent at Canterbury Printing Unit.
3. J. Balthrop, F. Esponda, S. Forrest, and M. Glickman. Coverage and generalization in an artificial immune system. In *Proceedings of the Genetic and Evolutionary Computation Conference (GECCO-02)*, pages 3–10, 2002.
4. Avrim Blum. On-line algorithms in machine learning. In *Online Algorithms*, pages 306–325, 1996.
5. D. W. Bradley and A. M. Tyrrell. The architecture for a hardware immune system. In D. Keymeulen, A. Stoica, J. Lohn, and R. S. Zebulum, editors, *The Third NASA/DoD Workshop on Evolvable Hardware*, pages 193–200, Long Beach, California, 12-14 July 2001. IEEE Computer Society.
6. D. W. Bradley and A. M. Tyrrell. A hardware immune system for benchmark state machine error detection. In *Proceedings of the Congress on Evolutionary Computation 2002 (CEC2002)*, Honolulu, Hawaii, May 2002.
7. D. W. Bradley and A. M. Tyrrell. Immunotronics: Novel finite state machine architectures with built in self test using self-nonself differentiation. *IEEE Transactions on Evolutionary Computation*, 6(3):227–238, June 2002.
8. D. Dasgupta, editor. *An agent based architecture for a computer virus immune system.* GECCO 2000 Workshop on Artificial Immune Systems, 2000.
9. P. D'haeseleer. An immunological approach to change detection: theoretical results. In *Proceedings of the 9th IEEE Computer Security Foundations Workshop.* IEEE Computer Society Press, 1996.
10. P. D'haeseleer, S. Forrest, and P. Helman. An immunological approach to change detection: algorithms, analysis and implications. In *Proceedings of the 1996 IEEE Symposium on Computer Security and Privacy.* IEEE Press, 1996.
11. R.O. Duda, P.E. Hart, and D.G. Stork. *Pattern Classification.* John Wiley and Sons, 2001.
12. F. Esponda, S. Forrest, and P. Helman. A formal framework for positive and negative detection schemes. *IEEE Transactions on Systems, Man and Cybernetics Part B: Cybernetics*, In press.
13. J. D. Farmer, N. H. Packard, and A. S. Perelson. The immune system, adaptation, and machine learning. *Physica D*, 22:187–204, 1986.
14. S. Forrest, B. Javornik, R. Smith, and A. Perelson. Using genetic algorithms to explore pattern recognition in the immune system. *Evolutionary Computation*, 1(3):191–211, 1993.
15. S. Forrest, A. S. Perelson, L. Allen, and R. CheruKuri. Self-nonself discrimination in a computer. In *Proceedings of the 1994 IEEE Symposium on Research in Security and Privacy*, Los Alamitos, CA, 1994. IEEE Computer Society Press.
16. T. Hastie, R. Tibshirani, and J. Friedman. *The Elements of Statistical Learning: Data Mining, Inference, and Prediction.* Springer, 2001.

17. S. Hofmeyr. *An immunological model of distributed detection and its application to computer security.* PhD thesis, University of New Mexico, Albuquerque, NM, 1999.
18. S. Hofmeyr and S. Forrest. Immunity by design: An artificial immune system. In *Proceedings of the Genetic and Evolutionary Computation Conference (GECCO)*, pages 1289–1296, San Francisco, CA, 1999. Morgan-Kaufmann.
19. S. Hofmeyr and S. Forrest. Architecture for an artificial immune system. *Evolutionary Computation Journal*, 8(4):443–473, 2000.
20. S. Hofmeyr, A. Somayaji, and S. Forrest. Intrusion detection using sequences of system calls. *Journal of Computer Security*, 6:151–180, 1998.
21. J.H. Holland. *Adaptation in Natural and Artificial Systems.* The University of Michigan Press, 1975.
22. A.K. Jain, M.N. Murty, and P.J. Flynn. Data clustering: A review. *ACM Computing Surveys*, 31(3):264–323, 1999.
23. N. Japkowicz. Are we better off without counter-examples? In *Proceedings of the First International ICSC Congress on Computational Intelligence Methods and Applications (CIMA-99)*, pages 242–248, 1999.
24. N. Japkowicz. *Concept-Learning in the Absence of Counter-Examples: An Autoassociation-Based Approach to Classification.* PhD thesis, University of New Jersey, New Brunswick, NJ, 1999.
25. M.J. Kearns and U.V. Vazirani. *An Introduction to Computational Learning Theory.* MIT Press, 1994.
26. J. Kim and P. J. Bentley. An evaluation of negative selection in an artificial immune system for network intrusion detection. In *Proceedings of the Genetic and Evolutionary Computation Conference (GECCO)*, pages 1330–1337, San Francisco, CA, 2001. Morgan-Kauffman.
27. T.M. Mitchell. *Machine Learning.* McGraw-Hill, 1997.
28. J. K. Percus, O. Percus, and A. S. Perelson. Probability of self-nonself discrimination. In A. S. Perelson and G. Weisbuch, editors, *Theoretical and Experimental Insights into Immunology*, NY, 1992. Springer-Verlag.
29. J. K. Percus, O. Percus, and A. S. Perelson. Predicting the size of the antibody combining region from consideration of efficient self/non-self discrimination. *Proceedings of the National Academy of Science*, 90:1691–1695, 1993.
30. S. Singh. Anomaly detection using negative selection based on the r-contiguous matching rule. In Jonathan Timmis and Peter J. Bentley, editors, *Proceedings of the 1st International Conference on Artificial Immune Systems (ICARIS)*, pages 99–106, University of Kent at Canterbury, September 2002. University of Kent at Canterbury Printing Unit.
31. S. T. Wierzchon. Discriminative power of the receptors activated by k-contiguous bits rule. *Journal of Computer Science and Technology*, 1(3):1–13, 2000.
32. S. T. Wierzchon. Generating optimal repertoire of antibody strings in an artificial immune system. In M. A. Klopotek, M. Michalewicz, and S. T.Wierzchon, editors, *Intelligent Information Systems*, pages 119–133, Heidelberg New York, 2000. Physica-Verlag.
33. S. T. Wierzchon. Deriving concise description of non-self patterns in an artificial immune system. In S. T. Wierzchon, L. C. Jain, and J. Kacprzyk, editors, *New Learning Paradigms in Soft Computing*, pages 438–458, Heidelberg New York, 2001. Physica-Verlag.