Meta-stable memory in an artificial immune network

Neal, Mark

Publication date:
2003

Citation for published version (APA):
Neal, M. (2003). *Meta-stable memory in an artificial immune network*. 168-180. http://hdl.handle.net/2160/35

Meta-Stable Memory in an Artificial Immune Network

Mark Neal

Department of Computer Science, University of Wales, Aberystwyth. UK
mjn@aber.ac.uk

Abstract. This paper describes an artificial immune system algorithm which implements a fairly close analogue of the memory mechanism proposed by Jerne(1) (usually known as the Immune Network Theory). The algorithm demonstrates the ability of these types of network to produce meta-stable structures representing populated regions of the antigen space. The networks produced retain their structure indefinitely and capture inherent structure within the sets of antigens used to train them. Results from running the algorithm on a variety of data sets are presented and shown to be stable over long time periods and wide ranges of parameters. The potential of the algorithm as a tool for multivariate data analysis is also explored.

1 Introduction

This paper presents an exploration of the capabilities of an unsupervised algorithm for generating networks of artificial recognition balls which capture structure and relationships within data sets. The algorithm represents the culmination of a series of attempts to produce a self-limiting, self-organizing, meta-stable network generation algorithm. The author believes that all of these goals have been satisfactorily achieved and has conducted extensive experiments to test this. A selection of these experiments are presented here using a simple graph-layout algorithm to present results, and principal component plots (PCA) to highlight equivalent structure within data sets. The algorithm also has the property of retaining its structure in the absence of any stimulation as was shown in (2), this is not explicitly shown here. The source code, a simple graph-layout program and a demonstration are available on request from the author.

2 Immune Network Theory

The immune network theory proposes that the B-cells in the body interact with each other to maintain the immune memory. The mechanism proposed is that B-cells which are capable of recognising similar (but not necessarily identical) pathogens are also capable of recognising and stimulating each other(1). Thus a dynamic feedback mechanism can maintain parts of the immunological memory which are not frequently stimulated. Clearly however not all B-cells have sufficient stimulation to survive indefinitely and thus some will die out. In the human immune system T-cells both perform a surveillance role and interact

with B-cells which complicates the mechanism somewhat. In our artificial immune system the role of T-cells is currently ignored. In the real immune system there are very large numbers of identical B-cells to deal with each type of infection. In an artificial system such repetition can be coded without representing all the identical cells individually. Fortunately the concept of a *recognition ball* which represents a region of antigen space that is covered by a particular type of B-cell can replace the repetition of individuals(3). So our AIS consists of a network of artificial recognition balls (ARB) which are linked together if they are close to each other in antigen space (see (2) for an earlier version of the algorithm). Pathogens (data items) can be considered to be points in this antigen space, and thus proximity can be defined as a simple distance function.

3 The Artificial Immune Network Algorithm

The algorithm used here is a development (in fact a simplification) of an earlier algorithm presented in (2). The new algorithm uses a very similar stimulation function and resource allocation mechanism, but the details vary in some important details. The algorithm now captures directly the ideas of *primary* and *secondary* response and is completely deterministic in its operation. The removal of the stochastic mutation operator was motivated by the wish to demonstrate the basic mechanisms and their important properties; namely the stability of the structures produced and the self-limiting and organizing growth of the networks.

3.1 The network affinity threshold (NAT)

As in previous work the most important and sensitive control parameter for the algorithm is the NAT. This value dictates when ARBs in the network are to be connected. The rule is simply that if the distance between two ARBs is less than the NAT value then they are connected. Thus, simple Euclidean distance between the patterns represented by ARBs dictates the connectivity of the immune network. This is directly analagous to the concept of the *recognition ball* as presented in (3).

In this algorithm the NAT plays a further rôle, in that it provides a threshold for the cloning process. If an antigen is presented to the network which is further from the most stimulated ARB than the NAT dictates, then cloning is performed.

Thus the NAT is a measure of the distance in antigen shape-space beyond which recognition by an ARB is deemed to be insufficiently precise.

Fortunately, from the point of view of data-analysis, this measure has a direct meaning in terms of the data under examination. It is the Euclidean distance between items beyond which it is deemed appropriate to make a distinction. Although this may not be known in advance, it is often the case that some sensible estimate can be made in advance based on known cases. Even if it is not, however, there is a relatively easy route to determining useful values for the NAT, by running the algorithm several times on the data-set (see section 4).

3.2 The algorithm

The algorithm is designed to run continuously, and does not have (or require) a stopping criterion. This was a fundamental goal when designing the algorithm

and meant that the algorithm had to deal with the problem of *over-fitting* in some more intelligent way than by just halting the training process. The *culling* process combined with the selection of a suitable value for the *network affinity threshold* perform this rôle.

Network initialization Currently the network is initialized by taking a small set of arbitrarily chosen samples of the antigen set and creating ARBs which precisely recognise those antigens. The number of these ARBs in the initial network is made as small as possible. If too few ARBs are initially present then the algorithm culls all the ARBs in the network. This occurs when none of the ARBs are sufficiently connected or stimulated to form a stable memory structure. The process of determining the minimum number of ARBs that produces a non-empty network can be automated by simply taking one initial ARB and running the algorithm to see if the network dies off or not. If the network dies off then the algorithm is re-run with one more initial ARB and the process is repreated until the network survives. This process is not time consuming as the algorithm very quickly culls all the ARBs if there are not sufficient to maintain a population.

Stimulation The stimulation level of an ARB is calculated using components analogous to those proposed by Jerne. These components are based on: affinity to the current antigen (positive contribution), and affinity to connected neighbours.

These components can be summarised as follows:

1. Excitation, ps due to affinity to the current antigen:

$$ps = 1 - dis(p)$$

2. Excitation, ns due to affinity to neighbours:

$$ns = \sum_{x=0}^{n} dis(x)$$

In both equations the function $dis(a)$ returns the Euclidean distance between the current ARB and the item a; and n represents the number of neighbours at the current ARB. These components are summed. The second component is based on the neighbours of the ARB, and there is no limit to the number of neighbours an ARB can have. Due to the way in which growth of the network proceeds, it is no longer necessary (or desirable) to normalise these neighbour contributions as was done in (2). This is one of several simplifications to the algorithm which were undertaken, and a return to the function used in (4).

Cloning Cloning is now only undertaken when expansion of the repertoire is required. This is determined by the fact that the most highly stimulated ARB (as calculated above) is further from the antigen pattern (as measured by Euclidean distance) than the distance dictated by the NAT. Thus there is now an explicit recognition of the difference between primary and secondary immune response: cloning is only undertaken when a primary response is required to expand the

repertoire to cover an unrecognised antigen. Secondary response simply involves the stimulation of the relevant pre-existing parts of the network.

The cloning operation is also extremely simple. It consists of introducing into the network an ARB which precisely recognises the pattern which *caused* the primary response. There is currently no mutation operator used in the algorithm. Initial work included mutation, but subsequent experimentation showed that the networks produced without it were very similar to those produced with it. Whilst this is a significant deviation from the natural immune system which seems to rely on highly stochastic processes for repertoire expansion, it is not seen as central to the concept of a network memory mechanism for the immune system.

Resource allocation Since early in the development of the series of algorithms which have culminated in this work, the concept of limited resources has played a part. Initial work in (5) used a very brutal and simplistic approach which took little account of time and was reliant on an "epoch-based" learning regime. This algorithm uses a simple stimulation and decay mechanism for updating the resource level held by each ARB. Thus the level of resources held by each ARB is calculated on a rolling basis according to two mechanisms: first a simple geometric decay; second a boost to the resource level which is dependent on the stimulation level and resource level of the current ARB. It is worth pointing out that this means that all resource allocation and stimulation calculations are *local* to each ARB and do not require normalization or a central resource pool.

The decay mechanism can be expressed thus:

$$R_{decayed} = R_{current} \times decayrate$$

where *decayrate* is a scalar between zero and one. For this work a value of 0.99 was used in all cases; and $R_{current}$ is the resource level currently present at this ARB. The algorithm is robust to a wide range of values of the *decayrate* scalar. Further discussion can be found in section 3.3.

The resource level after each data presentation can be expressed as follows:

$$R_{new} = R_{curent} + (k \times (maxres - R_{decayed}) \times S)$$

where R_{new} is the new resource level for the ARB, k is a scalar between zero and one (a value of 0.0005 was employed throughout this work. *maxres* is a maximum resource level which any ARB can claim. Throughout this work the value was set at 1000.0. The algorithm is robust to a wide variety of values for these two scalars. For more detailed discussion of sensitivity of the algorithm to both of these scalars see 3.3.

Culling ARBs are culled from the network when there resource level (as described above) falls below a threshold value. We call this threshold value *mortality* and throughout this work it was set at 1.0. The algorithm is robust to a wide variety of values of *mortality* (see section3.3).

3.3 Parameters and value selection

From the above, it can be seen that there are several scalars used to parameterise the algorithm. These are *decayrate*, $maxres$, *mortality* and k. All of these values are involved in the resource allocation process. Whilst a set of four parameters may seem daunting and dangerously "tweakable", they are in fact nowhere near as fiddly as they at first seem. In order to understand why this is so we must examine the resource allocation process in more detail, and the behaviour of typical networks of ARBs generated by the algorithm.

The resource decay mechanism is simple enough (see above). The resource boost mechanism is however a little more subtle. The level of boost given to an ARB's resource level is determined by a variety of factors:

Stimulation level (S): this ensures that ARBs which regularly recognise antigen patterns and/or are in a highly linked section of the network will accumulate more resources than ARBs which are not, and thus will survive longer.

Proximity to maximum resource level ($maxres - R_{decayed}$): this ensures that ARBs which are already rich in resources will not go on claiming more and more without limit. Their resource level will geometrically approach the value $maxres$.

A small scalar (k): a small constant to ensure that ARBs with high stimulation levels do not achieve extremely high resource allocations very rapidly. This is desirable as it allows large values of *decayrate* and thus long time lags between the creation of ARBs and their ultimate survival or demise.

Extensive experimentation with the variables k, $maxres$, *mortality* and *decayrate* has shown that the ranges of values for these variables shown in table 3.3 produce very similar networks in all cases.

Variable	Min	Network sizes	Max	Network sizes	Standard value
k	0.0001	39	0.001	33	0.0005
$maxres$	500	37	5000	32	1000
mortality	0.0	34	5.0	30	1.0
decayrate	0.90	34	0.999	35	0.99

Table 1. Ranges of variables shown to produce very similar networks. Statistics were generated using Fisher's iris data.

This belief can be further reinforced when the mechanisms employed and the behaviour that they promote are considered. The boosting of resource level using an amount that reduces dependent on the current resource level is clearly going to strike a balance at some level when combined with a geometric decay function. So long as this equilibrium falls somewhere reasonably far away from both of the limiting values for the majority of the ARBs then the precise values of these two limits will have little effect on the networks produced. These two values are provided by $maxres$ as an upper limit, and *mortality* as a lower limit.

In a similar manner, the values of k and *decay* are in the first instance simply going to affect the level at which this balance is to be struck. Once again so long as these values are arranged to ensure that the resource levels at which most ARBs stabilise falls well away from the upper and lower limits then little effect will be observed. This effect can be seen in figure 1, which shows the maximum and average resource levels in the network produced for the iris data with two different values for *decayrate*. Although the values of the parameters and the average, maximum and minimum resource levels are all significantly different, the size evolution of the network is almost identical in all cases (see figure 2).

Fig. 1. Maximum and average stimulation levels present in the network throughout the evolution of the networks.

Fig. 2. Size evolution of networks with decay rates of 0.9 and 0.99. The meta-stable state size of both networks is very similar.

4 Forming stable memories of data

In order to demonstrate the algorithm's behaviour, it has been tested extensively with a variety of data-sets. Results from three of these data-sets are presented here. The first of these is Fisher's iris data[1](6) (which provides a trustworthy benchmark), the second is a larger data-set of much higher dimensionality which contains statistical information about a number of gene sequences[2](7), and the third is the now well-explored Wisconsin breast cancer data set[3](8). The algorithm was used in all cases with all the standard parameter values from table 3.3. A wide range of values for the NAT were used in order to explore the structures of all three data-sets. In all cases the algorithm was run until the network ceased to grow. That is not to say that the network completely stabilized, just that a meta-stable state of approximately constant size was attained. Graphs showing final network sizes show the median value for network size after meta-stability has been achieved.

In the cases of Fisher's iris data and the Wisconsin breast cancer data the way in which the data is clustered is presented as confusion statistics. These statistics are generated by taking each Bcell and using the class of the majority of the data items claimed by that Bcell as the "correct" class for that Bcell; and then counting the number of data items which do not fall into the majority class at that Bcell. Where there is an equal number of data items of the different classes, *all* of the data items are labelled as misclassified. The complete statistics are shown for a variety of networks at different NAT values.

Such data is not presented for the statistical sequence data due to the poor performance of the algorithm and the large number of classes (see section 4.2).

[1] this data-set contains 150 data items in four dimensions
[2] this data-set contains 1693 data items in 435 dimensions
[3] this data-set contains 699 data items in 9 dimensions

4.1 The iris data

The iris data has provided a useful benchmark data-set in previous closely related work (2; 5). The data-set has the advantage of being relatively small and contains simple, but interesting and non-trivial structure. See figure 6 for a principal component plot which captures the majority of the structure of the data-set. The algorithm produces networks which separate the data into two distinct clusters. This is as expected and as has been observed in previous versions of the algorithm. One of the networks produced by the algorithm presented here is shown in figure 5. The network was generated with a NAT value of 0.7 and clearly shows the separation of the data into well-defined clusters. The confusion statistics are presented in table 4. The network can be seen to correctly classify 97.3% of the data items at this NAT value. A better performance figure is acheived at a NAT value of 1.5, however at this point the network produced is less informative in structure, and captures much less detail of the structure inherent in the data. From the perspective of the exploration of data in a data-mining context the network produced at a NAT value of 0.7 is more appealing. In a similar manner to the other parameters examined above it is reassuring to note that the precise value of the NAT does not dramatically affect the performance of the algorithm.

As observed in (9), it is interesting to note that one of the data classes tends to dominate the structures produced by the algorithm. For the iris data it is the Setosa class which is *always* correctly classified, and as the NAT increases the Virginica class dominates at the expense of the Versicolor class. This effect is not so damaging for this version of the algorithm however, as the network does stabilise with all the components of the network present over a large range of NAT values. This behaviour was not seen in previous versions of the algorithm in which all but one region of the network eventually died.

Thus for the iris data we can conclude that the algorithm performs very well in terms of classification, and in terms of generating meaningful networks over a good range of NAT values.

4.2 The statistical gene sequence data

This data set was chosen as it is quite large, and contains relatively little useful structure (see figure 9). The data set contains statistical information about the make-up of genetic sequences and labels which represent the functional class of the gene. There are 17 different functional classes represented in this data (although there are subdivisions within these also). The data was examined in

Fig. 3. Classification errors produced at various NAT values for the iris data.

Fig. 4. Meta-stable network sizes reached at various NAT values for the iris data.

Fig. 5. Network produced by the algorithm at a NAT value of 0.7. We believe that qualitatively similar patterns can be discerned in the PCA plot and the network.

Fig. 6. PCA plot of first two principal components for Fisher's iris data. Squares represent setosa examples, circles represent virginica examples and triangles represent versicolor examples.

order to test the performance of the algorithm, both for large data sets and for data sets which do not contain easily separable clusters. The performance of the algorithm with respect to the functional classes of the genes represented was, as expected, quite poor. The only functional class (class 29) which was reasonably separable can be seen as a more diffuse cloud of points above and to the left of the main clump in figure 9. This was reflected to an acceptable degree in the networks produced, and an example network generated at a NAT of 1.45 is shown in figure 8, the diamond-shaped structure attached to the upper-right of the network represents those data items belonging to class 29. It is worth saying that this relatively poor performance is not surprising when the PCA plot is examined: there is very little inherent structure for an unsupervised algorithm to capitalise on. In addition, when examined with standard single linkage cluster analysis (results not shown), it was very clear that there were no clear clumps of data that could be easily separated. When supervised algorithms such as C4.5 were applied to the data, much better classifications were possible. However it is not really sensible to compare performance with such supervised algorithms. Suffice it to say that there *is* sufficient information within the data to separate the classes reasonably effectively using supervised machine learning algorithms. This is stated simply to make clear that the data is worthy of examination, and that the algorithm presented here is restricted in just the same way as other unsupervised algorithms.

The data set provided a good test for the time complexity of the algorithm with a large data-set of high-dimensionality. The most useful network (with NAT of 1.45) was stable after 10000 data presentations (about 6 passes through the data set) and took 8 minutes to run on a 1.1GHz Celeron lap-top. For a data set of this size and a relatively modest computer system this is acceptable performance. The data set is available from (10).

Class	NAT	Correct	Incorrect	% Class Correct	% Total Correct
Setosa	0.7	50	0	100	
Virginica	0.7	49	1	98	
Versicolor	0.7	47	3	94	97.3
Setosa	1.0	50	0	100	
Virginica	1.0	45	5	90	
Versicolor	1.0	46	4	92	94
Setosa	1.5	50	0	100	
Virginica	1.5	50	0	100	
Versicolor	1.5	48	2	96	98.7
Setosa	3.0	50	0	100	
Virginica	3.0	50	0	100	
Versicolor	3.0	0	50	0	66.6

Table 2. Classification performance at various NAT values for Fisher's iris data.

4.3 The Wisconsin breast cancer data

The Wisconsin breast cancer data is now well explored, and is a relatively easy data-set to classify to a fairly high accuracy. It is intermediate in size between the previous two data-sets and provides a further test of the suitability of the algorithm for separating relatively well-defined clusters of data. Unlike previous versions of the immune network algorithms, this algorithm retains segments representing both classes of data when it reaches its meta-stable state(9; 11). The final network represents the data in a way which reflects the structure seen in the PCA plot. This behaviour is also seen for the iris data (see above).

The network which was chosen as the most useful is that which was generated with a NAT value of 1.0. This network is shown in figure 12. For this data, this network is the best at classifying the data (see figure 11. In this case it is also the most appealing network to examine in the vizualization tool with a simple spring-embedder graph-layout algorithm (12).

Class	NAT	Correct	Incorrect	% Class Correct	% Total Correct
Benign	0.05	432	12	97.3	
Malignant	0.05	226	13	94.6	96.3
Benign	0.7	428	16	96.4	
Malignant	0.7	232	7	97.1	96.6
Benign	1.5	430	14	96.9	
Malignant	1.5	230	9	96.2	96.6
Benign	2.0	432	12	97.3	
Malignant	2.0	216	23	91.4	94.9
Benign	4.0	442	2	99.5	
Malignant	4.0	166	73	69.5	89.0

Table 3. Classification performance at various NAT values for Breast Cancer data.

5 Discussion

From the experiments that have been undertaken with various data sets it seems that the networks generated using this algorithm and displayed with a spring-embedder graph-layout algorithm have the following properties:

– There do not seem to be drastic changes in the networks produced for similar values of *any* of the parameters to the algorithm (including the NAT). The algorithm is quite insensitive to these parameters, although gross changes in the NAT will produce noticeably different networks.

Fig. 7. Steady-state size of network generated at various NAT values for the statistical sequence data set.

Fig. 8. Network produced with NAT of 1.45 for the statistical sequence data. Qualitatively similar patterns can be seen in fig. 9.

Fig. 9. PCA plot of first two principal components for the statistical sequence data. Compare with 8.

– Networks which perform well are often intuitively appealing and display structure that is inherent to the data as clusters in the network.
– Networks produced by the algorithm seem to pay more than a passing resemblance to the principal component plots of the first two most significant components. Obviously this will not necessarily be a direct mapping in two dimensions, but clusters which are apparent in the PCA plot will be distinguishable in the network layout and vice-versa. We make no claim for a theoretical basis for this observation.

These properties along with the self-organizing nature of the algorithm mentioned in the introduction indicate that we have obtained behaviour that reflects the ability to generate stable memory structures in the artificial immune networks generated. Furthermore these networks pick out structures inherent in the data and can be laid out in two dimensions for interactive examination when seeking to analyse data. Thus the algorithm forms a useful addition to the toolbox of the data-miner: the networks are intuitive to manipulate, easy to vizualize, retain relationships within the data, and perform a drastic dimensionality reduction. We are now satisfied that this algorithm represents a version with which we can usefully attempt some real data-mining problems, and the software (whilst still most definitely a prototype) is sufficiently developed to allow this.

6 Conclusion

A development of an unsupervised algorithm for generating artificial immune networks was presented. Evidence of its ability to extract useful structure from complex data-sets was presented. These results are now of a quality and reliablity that will allow the algorithm to be used in anger on some real data-mining problems. The software develolped is available on request from the author.

Fig. 10. Steady-state size of network generated at various NAT values for the Wisconsin Breast Cancer data set.

Fig. 11. Number of errors made in classifying the two classes for the Wisconsin Breast Cancer data set with increasing NAT value.

Fig. 12. Network generated with NAT of 1.0 for Wisconsin Breast Cancer data set. The Malignant examples appear in the sparse cluster at the top of the image. Compare with fig. 13.

Fig. 13. PCA plot for Wisconscin Breast Cancer data. $\times$ represents benign examples and $+$ represents malignant examples. Note denser concentration of benign data points on the left of plot.

Bibliography

[1] Jerne, N.K.: Towards a Network Theory of the Immune System. Ann. Immunol. (Inst. Pasteur) **C** (1979) 373–389

[2] Neal, M.: An Artificial Immune System for Continuous Analysis of Time-Varying Data. In Timmis, J., Bentley, P., eds.: Proceedings of the First International Conference on Artificial Immune Systems (ICARIS), Canterbury, UK, UKC (2002) 76–85

[3] Perelson, A.S.: Immune Network Theory. Imm. Rev. (1989) 5–36

[4] Timmis, J.: Artificial Immune Systems: a novel data analysis technique inspired by the immune system (PhD Thesis). University of Wales, Aberystwyth (2000)

[5] Timmis, J., Neal, M.: A Resource Limited Artificial Immune System. Knowledge Based Systems **14** (2001) 121–130

[6] Fisher, R.A.: The use of multiple measurements in taxonomic problems. Annual Eugenics **II** (1936) 179–188

[7] Clare, A.: Machine learning and data mining for yeast functional genomics (PhD Thesis). University of Wales, Aberystwyth (2003)

[8] Mangasarian, O.L., Wolberg, W.H.: Cancer diagnosis via linear programming. SIAM News **23** (1990) 1,18

[9] Knight, T., Timmis, J.: AINE: An immunological approach to data mining. In: Proc. of the IEEE International Conference on Data Mining. (2001) 297–304

[10] Clare, A.: http://users.aber.ac.uk/compsci/Research/bio/dss/yeastdata/. University of Wales, Aberystwyth (2003)

[11] Wierzchon, S., Kuzelewska, U.: Stable Clusters Formation in an Artificial Immune System. In Timmis, J., Bentley, P.J., eds.: Proceedings of the 1st International Conference on Artificial Immune Systems (ICARIS), University of Kent at Canterbury, University of Kent at Canterbury Printing Unit (2002) 68–75

[12] Timmis, J.: aivis - artificial immune network visualisation. In: EuroGraphics UK 2001 Conference Proceedings, Univerisity College London., Eurographics (2001) 61–69