Clustering Moving Data with a Modified Immune Algorithm

Emma Hart and Peter Ross

School of Computing, Napier University
Edinburgh EH14 1DJ, Scotland
{emmah,peter}@dcs.napier.ac.uk

Abstract. In this paper we present a prototype of a new model for performing clustering in large, non-static databases. Although many machine learning algorithms for data clustering have been proposed, none appear to specifically address the task of clustering moving data. The model we describe combines features of two existing computational models — that of Artificial Immune Systems (AIS) and Sparse Distributed Memories (SDM). The model is evolved using a coevolutionary genetic algorithm that runs continuously in order to dynamically track clusters in the data. Although the system is very much in its infancy, the experiments conducted so far show that the system is capable of tracking moving clusters in artificial data sets, and also incorporates some memory of past clusters. The results suggest many possible directions for future research.

1 Introduction

As the ability to collect and store vast quantities of data increases, having some facility to intelligently and efficiently analyse that data in order to detect clusters, patterns and meaningful correlations becomes essential. Many algorithms have been proposed to perform some or all of these tasks, however it seems clear that a successful algorithm must address the following key features of larger databases if it is to prove useful in the real-world:

- The data-base is likely to be non-static; data is continually added and deleted
- Trends in the data change over time
- The data may be distributed across several servers
- The data may contain a lot of 'noise'
- A significant proportion of the data may contain missing fields or records

Recently, a growing body of work has shown that the biological immune contains many desirable features which allow it to be used to address some of the characteristics listed above. In (one) extremely simplified view, the immune system (IS) can be considered to be a decentralised self-organising system which operates by producing antibodies which recognise potentially harmful invaders and eliminate them from the body. The recognition takes place via some kind of

E.J.W. Boers et al. (Eds.) EvoWorkshop 2001, LNCS 2037, pp. 394–403, 2001.

sophisticated pattern matching mechanism which allows it to access a content addressable memory of past invaders. The matching mechanism is imprecise — an antibody is stimulated by an antigen if the strength or affinity of the match between the two exceeds some threshold. Any antibody which stimulates an antigen is said to be within the ball of stimulation of the antigen. Moreover, the IS is able to dynamically learn about new substances when it encounters them and add them to its memory. At the same time, any little used information is deleted from the memory, therefore the memory is continually changing. Several of these features have been modelled in a number of very different implementations of artificial immune systems and applied to the problem of clustering data. For example, Potter *et al.* describe a model of an AIS that uses a coevolutionary genetic algorithm (GA) to evolve antibodies to cluster artificial data sets [1] and Congress voting records [2]; Forrest *et. al* [3] describe a GA that uses emergent fitness sharing to find patterns; Hunt *et al.* [4] describe a system named *Jisys* which was used to cluster data for use in mortgage fraud detection and Timmis [5] has adapted this system to successfully cluster the well known but very small benchmark data set containing iris petal sizes. Both the Timmis and Hunt work used a model based on connected networks of antibodies, in which nodes which are connected recognise similar patterns. So far, none of these methods have addressed the question of clustering data in time-varying databases. Although there is no intrinsic barrier to extending either the coevolutionary or network models to deal with non-stationary data, both methods present obstacles. In the network model, there are high computational overheads associated with re-organising large networks as the data changes, which increase as the size of the database increases also. It is also unclear whether the coevolutionary method of evolving clusters is able to cope with extremely large databases, particularly as the antibodies compete to exclusively recognise data, whereas in reality clusters may overlap.

In other work, Smith *et. al* [6] have shown that the immune system can be considered to be representative of the same class of memories as Kanerva's Sparse Distributed memory, [7]. The SDM is a content-addressable memory which was originally proposed as an efficient method for storing a very large number of large binary data patterns using a very small number of physical data addresses, in a manner which allows accurate recall of all the data. An SDM is composed of a set of physical or hard locations, each of which recognises data within a specified distance of itself — this distance is known as the recognition radius of the location. Each location also has an associated set of counters, one for each bit in its length, which it uses to 'vote' on whether a bit recalled from the memory should be set to 1 or 0. An item of data is stored in the memory by distributing it to every location which recognises it — if recognition occurs, then the counters at the recognising locations are updated by either incrementing the counter by 1 if the bit being stored is 1, or decrementing the counter by 1 if the bit being stored is 0. To recall data from the memory, all locations which recognise an address from which recall is being attempted vote by summing their counters at each bit position; a positive sum results in the recalled bit being set to 1, a negative sum

in the bit being set to 0. This results in a memory which is particularly robust to noisy data due to its distributed nature and inexact method of storing data. These properties make it an ideal candidate for addressing clustering problems in large databases; For example, we can consider each physical location along with its recognition radius to define a cluster of data; the location itself can be considered to be a concise representation or description of that cluster, and the recognition radius specifies the size of the cluster. Clusters can overlap — indeed, it is this precisely this property which allows all data to be recognised with high precision whilst maintaining a relatively low number of clusters. If no overlap is allowed, then a large number of locations are required to cluster the data, the system becomes overly specific, and hence general trends in the data are lost. In the form described, the SDM is also static and inflexible, however given its powerful and efficient storage and recognition capacities, it is fruitful to adapt it to operate in a dynamic environment. Therefore, the model we now describe combines features of SDM and the type of AIS describe by Potter to produce a system that is dynamic, adaptable and capable of tracking changes in large volumes of data. For simplicity, during the remainder of this paper we use immunological terminology — an antigen is equivalent to a piece of data, an antibody to a description of a cluster, and the ball of stimulation of the antibody defines the size of the cluster.

2 Description of the Proposed Model

The proposed model is shown in figure 1. The basic proposition is to use a coevolutionary GA, running continuously, to find quickly the set of antibodies (and their corresponding balls of stimulation) that best cluster the data currently visible to the system. An *antigen* is represented by a bit string of length L. An *antibody* is also represented by a bit string of length L, and also defines the recognition radius R of the antibody. Each antibody has an associated set of counters, one for each bit, which are used to 'vote' on whether the bit should be set to 1 or 0 as described in the previous section. The accuracy of the SDM formed by the set of antibodies can be determined by attempting to recall each data item stored and comparing the results to the actual data in the database. The coevolutionary GA controls the evolution of k populations of antibodies — each population is attempting to evolve the location and radius of one of the antibodies defining the memory (and therefore the clusters). At any time t, the best antibodies in each population cooperate to form an SDM in which all data visible to the system at this time can be stored and ideally accurately recalled (and hence clustered). The mechanism by which the evolution proceeds is detailed in the next section.

3 Experimental Details

In order to calculate the fitness of an antibody in any population (which only represents a partial solution to the problem), the antibody is added to a serum

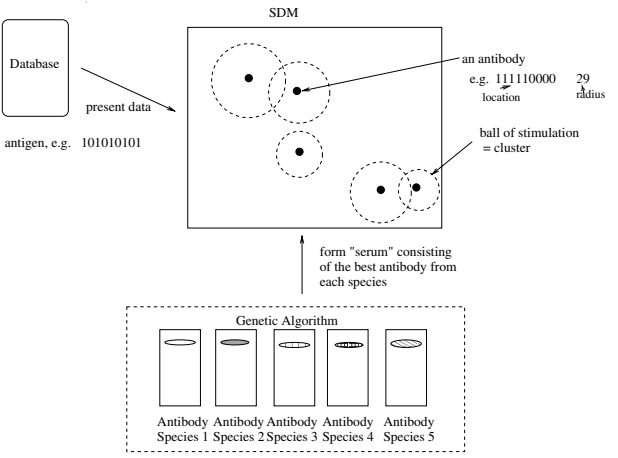

Fig. 1. The proposed model, combining features from an SDM, AIS and coevolutionary genetic algorithm

consisting of itself and the best members of the other populations.[1] The counters of each antibody in the serum are set to 0. All the antigens in the database are then stored in the SDM defined by this serum, and then recall attempted of each antigen. Antigens not recognised by any of the antibody species are allocated to a default cluster which is defined by the antibody '0000...000' with a recognition radius of L. Otherwise, the antigen is assigned a match-score equivalent to the number of correctly recalled bits. The fitness of the antibody is then set to the average value of M. Antibodies which cooperate with other antibodies to more accurately represent the dataset are thus more highly rewarded. Note that antibodies are *not* exclusively competing for antigen — several hard locations in the serum may recognise an antigen and thus collaborate in order produce the recalled data, which should result in a higher recall accuracy than in the system described in [1]. This bears a close analogy to the real immune system in which a cross-reaction between antibodies can occur.

3.1 Control of Number of Species

The number of species is dynamic, that is species are added and deleted from the algorithm as becomes necessary. The rate at which this happens is controlled by 4 parameters; the extinction threshold, e, the learning phase, l, the stagnation phase length, sp and the stagnation level st. If the fitness of the serum composed of the best member of each species does not increase by at least st over spgenerations, then a new species is added to the system, with randomly generated members. Similarly, if the best member of a species does not recognise at

[1] In the initial generation of the algorithm, antibodies are chosen at random from populations that have not yet been evaluated when forming the serum

least e antigens from the current antigen population, and the species has been in existence for at least l generations then that species is removed from the system, with the caveat that if the species recognises an antigen that is *not* recognised by any other antibody then the species is allowed to remain. A limit of M species is imposed on the system to prevent it growing too large (and therefore too specialised).

4 Experiments

Two series of experiments were performed in order to investigate the capability of the system. The first series of experiments investigated the ability of the system to track clusters which vary in a random manner with time. The second series was concerned with investigating the performance of the system in an environment in which data appears in cycles, and is designed to test the ability of the system to react more quickly to clusters of data which it has previously encountered.

Generating Data. In all experiments, the system is continually exposed to a set of 100 antigen. The antigen are generated from s schemas. Each schema consists of a string of 64 bits, in which d contiguous bits are set to 1, with the start position of the d bits randomly chosen. All remaining bit positions contain wild-cards. Antigen are generated in equal proportion from each schema by randomly replacing wild-cards with either 0 or 1. In order to generate non-stationary data, the following procedure is followed. 100 antigens are generated at time $t = 0$ from s schema. Every U time-steps, g schemas are chosen at random and replaced by g new randomly generated schema. New antigens are generated from the new schema and replace those antigens generated from the schema being replaced.

Tolerization Period. In each experiment, the system is allowed to undergo a *tolerization* period of T iterations in order to learn the data present at $t = 0$. This is necessary in order to accurately measure the response of the system to data changes from a state in which it has accurately clustered the current data. If this was not present, the system may still be learning the original data when changes occur, and hence we are not measuring the ability of the system to *adapt* to new data from an already stable state. This can be considered similar to the *neonatal* period in humans in which the body is thought to become tolerant of 'normal' proteins, [8]. In all experiments described, T is set to 200.

5 Simple Pattern Tracking

In the first series of experiments, a number of tests were performed for an update rate $U = 50$, varying the parameters s, d, and g. In this paper, due to space constraints, we report the results from experiments with $s = 5$,$d \in (8, 16, 32)$ and $g \in (1, 2, 3, 4, 5)$. Each experiment was repeated 5 times, and the resulting fitness at each time averaged. In each experiment, the size of each population

was set to 50. Initially, 2 populations are created, and a maximum limit of 10 populations is imposed. The populations are evolved using fitness proportionate selection, 2-point crossover, and bit-flip mutation at a rate of $1/L$ per gene. The parameters controlling the dynamics of the evolution were set to $e = 5, l = 10, sp = 5, st = 0.5$.

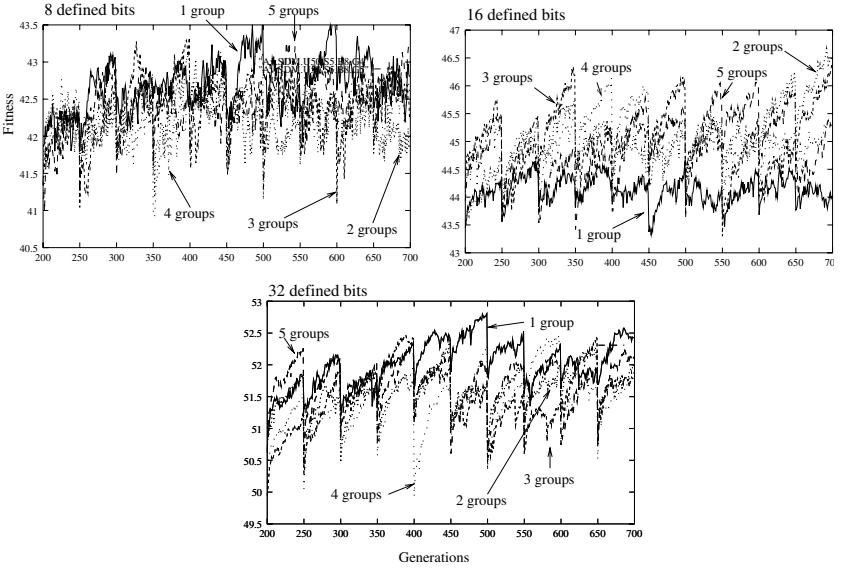

Fig. 2. The figure shows the performance of the proposed system for 3 experiments in which groups of $1 \rightarrow n$ schema are replaced at each update. The vertical axis shows the fitness of the best SDM found, where fitness is equal to the average number of correctly recalled bits across the entire data set.

5.1 Results

Figure 2 shows the results of the experiments for each combination of d and g, i.e. number of defined bits and number of schemas replaced. The results are shown from the end of the tolerization period only so that trends can be more clearly observed. In order to analyse the trends more thoroughly, the magnitude of the average drop in fitness in the system whenever a change in antigen occurs is plotted against the number of schemas replaced. This is shown in figure 3. In all experiments, a single antibody with all bits set to 1 and a radius of L should provide the most general clustering of the system, as this would match all of the possible sequences of d defined bits comprising the data set. This antibody would produce a recall score for the entire data set of 36,40 or 48, for the cases when $d = 8$, $d = 16$ and $d = 32$ respectively. This is calculated by

considering that exactly d bits would match perfectly and on average $(l-d)/2$ of the remaining bits would match. This gives a baseline against which to compare the performance of the system.

The first observation to note from Figure 2 is that in all experiments, the fitness of the system immediately before any antigen change always exceeds the baseline fitness described above, therefore the system is clearly performing some clustering. The success of the system varies as the number of antigen replaced at each update changes, and also as the number of bits defining the schema change. The greatest effect on performance of the number of antigens replaced is observed when $d = 32$. This is as expected — if the number of defined bits is large compared with the length of the antigen, then the number of bits that must change in order to recognise different schema may be very large (for example, in the extreme case, if schema consisting of `'11111.....#####'` is replaced by one described by `'#####....11111'` then $L/2$ bits may need to change.) Hence we expect to see a large variation in performance as the number of antigen replaced increases. At the other end of the scale, when $d = 8$, and therefore small compared with the length of string, it is likely that common patterns exist in the schema other than those described by the section of defined bits, and therefore introducing new antigen produces much less overall change in the composition of the entire antigen data set, as these spurious patterns can always be generated by chance. In the case where $d = 16$, slightly anomalous results are observed, as the worst fitness values are obtained when $g = 1$, i.e. when only 20% of the antigen population is updated. This requires further investigation, particularly as figure 3 indicates that the drop in fitness between updates increases approximately linearly as the number of schemas replaced increases for $d = 16$ but that the drop is higher than in the case when $d = 32$. Figure 3 shows that although the drop in fitness increases with number of schema replaced, the drop is actually small in proportion to the relative fitness of the system; the largest drop obtained is approximately 1.3, whereas the fitness of the system is generally higher than 41.

6 Investigating the Memory Retention

The second series of experiments aimed to investigate whether the system could retain some memory of past clusters so that if a cluster reappears the system responds to it more rapidly. In this series of experiments, antigen sets were again generated from sets of schema in the following manner:

$c * s$ schema are initially generated in the manner previously described. At any time t, only s of these schema are used to generate the antigen population. A sliding window of size s defines which schemas are used; this window is moves w schemas along the schema list every U generations. The schema list is treated as cyclic and wraps around when the window reaches the end. Thus, if $c = 2$ and $s = 4$, then 8 schemas are initially generated. If w is equal to c, then all antigens are replaced at each update; thus at time $t = 0$, antigens $\{0, 1, 2, 3\}$ define the data set. At time U, antigens $\{4, 5, 6, 7\}$ define the data, at time $2U$, antigens

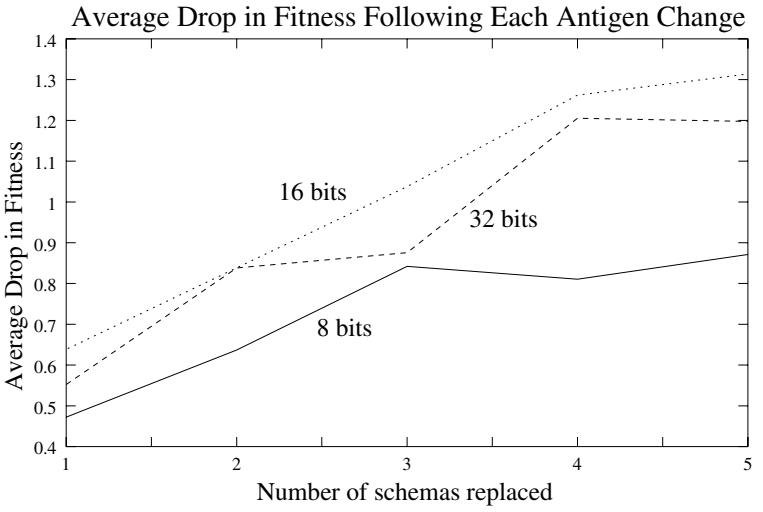

Fig. 3. The figure shows the drop in fitness experienced following a change in antigen for three experiments using schema containing 8,16, or 32 defined bits.

$\{0, 1, 2, 3\}$ again define the data etc. A more incremental update is achieved by setting $w < s$.

The experiments reported have $s = w$ where $s \in (2, 5, 10)$. The update rate U is set to 50 generations, and c is set equal to 2. Again, a tolerization period of T generations is allowed, in which the system learns the first s schemas. Thereafter, the schema set alternates between the two possible schemas (referred to as set A and B in the following discussion) every 50 generations. Figure 4 shows the results of these experiments. In each case, the experiments are compared to an equivalent experiment in which the antigen set is updated from randomly generated schema at each update. Clearly, the experiments which replace antigen with previously encountered antigen outperform the random set, showing that the system must be displaying some kind of memory. Again, the results are only shown from the end of the tolerization period.

To investigate the 'period' of this memory, we analyse the best fitness found for schema set A each time it appears. The experiment described above is repeated for values of c equal to 3, and 4, so that $3 * s$ schemas are generated in the former case, and $4 * s$ in the latter. In each case, the experiments are run for sufficient generations that the schema set A appears 5 times. The best fitness found on each occasion is averaged over the entire experiment and the results are shown in table 1. t-tests applied to each pair of results shows that the only significant difference in values is found between cases $(c = 2, s = 2)$ and $(c = 3, s = 2)$, $(c = 3, s = 2)$ and $(c = 4, s = 2)$, and finally between cases $(c = 3, s = 10)$ and $(c = 4, s = 10)$. Therefore, the system appears relatively robust to the parameter c which controls the period of the memory.

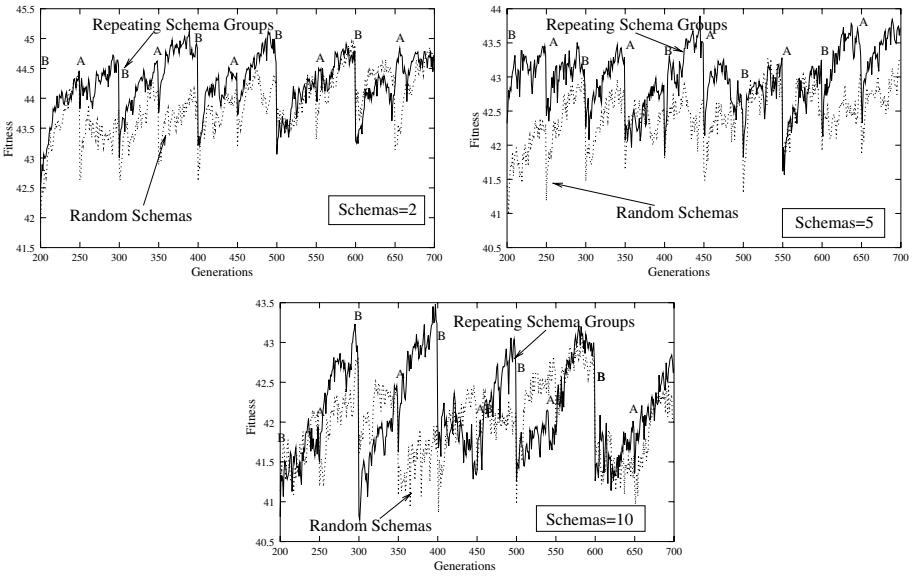

Fig. 4. Comparison of experiments in which new antigen are generated from new, randomly generated, schema to those in which antigen are generated from schema that the system has previously been exposed to

Table 1. The mean and standard deviation (shown in brackets) of the maximum fitness found for schema set A averaged across 5 occurrences of the set

Number of Schemas	Mutltiplier for Number of Schema c		
s	2	3	4
2	44.63 (0.186)	44.37 (0.274)	44.77 (0.322)
3	43.06 (0.382)	44.1 (0.342)	43.01 (0.405)
4	42.72 (0.266)	42.78 (0.474)	42.79 (0.281)

7 Conclusion

So far, only a very preliminary investigation of the capabilities of the proposed new system have been investigated. However, in light of those experiments reported here, and others to be reported in future publications, we make the following observations;

- The model appears capable of clustering data sets; this has been tested with up to 10 clusters.
- The model satisfactorily copes with moving data; the experiments show that the model tracks both incremental and large changes in data, but performance degrades as the amount of data changing increases.

– The model exhibits a basic form of memory; when re-exposed to familiar antigen, it reacts more rapidly than to previously unseen antigen.

Clearly, there is much more work to do to fully assess the performance of the model, however the results found so far are promising. The model described seems to provide a sensible method of addressing the difficulties concerned with clustering data in non-stationary databases. Issues that must be addressed in future however include investigating the scalability of the system, the robustness of the system to noise in the data and to the distance between clusters. Although the system produced satisfactory results with arbitrarily chosen parameters, we wish to investigate the sensitivity of the parameter choices. Finally, we intend to compare the model to other methods which potentially could be employed to track non-stationary data.

References

1. M.A. Potter and K.A De Jong. Cooperative coevolution: An architecture for evolving coadapted subcomponents. *Evolutionary Computation*, 8(1):1–29, 2000.
2. M.A Potter and K.A De Jong. The coevolution of antibodies for concept learning. In *Parallel Problem Solving From Nature - PPSN V*, pages 530–540. Springer-Verlag, 1998.
3. S Forrest, B Javornik, R.E Smith, and A.S Perelson. Using genetic algorithms to explore pattern recognition in the immune syste. *Evolutionary Computation*, 1(3):191–211, 1993.
4. D Dasgupta, editor. *Artificial Immune Systems and Their Applications*, chapter Jisys: The Development on An Immune System for Real World Applications, pages 157–184. Springer-Verlag, 1999.
5. J. Timmis and M. Neal. A resource limited artifical immune system for data analysis. In *Expert Systems 2000: International Conference on Knowledge BAsed Systems and Applied Artificial Intelligence*. Springer-Verlag.
6. D.J Smith, S Forrest, and A.S Perelson. *Artificial Immune Systems and Their Applications*, chapter Immunological Memory is Associative, pages 105–112. Springer-Verlag, 1999.
7. P Kanerva. *Sparse Distributed Memory*. MIT Press,Cambridge,MA, 1988.
8. R.E Billingham, L. Brent, and P.B. Medawar. Actively acquired tolerance of foreign cells. *Nature*, 172:603–606, 1953.