Artificial Immune Systems

Lee Hall

Fri 07 Dec 2012 01:39:09 AM EST

1 Abstract

An Artificial Immune System (AIS) is a novel approach to guided classification that combines traditional genetic algorithms with a model of biological immune systems. Our approach is based on the work of von Zuben and de Castro[1], which was in turn based on the earlier work of D. Dasgupta[2]. We attempted to apply these techniques to hyperspectral image classification, by generalizing an existing classifier that recognized vectors in $\mathbb{N}^3$ to work instead with vectors in $\mathbb{R}^n$. This generalization revealed flaws in the underlying codebase that gave inconclusive results on the original project goals, but did provide some interesting information on the utility of various parts of the AIS algorithm.

2 Introduction

One of the more common means of building a classifier is a Genetic Algorithm (GA). A GA initially generates a random set of states. After applying a fitness heuristic to select the most fit states, it combines these states and makes random perturbations before repeating the process in an attempt to converge on an ideal state[3, 126]. The AIS system is a relative of the GA, also biologically inspired, which uses simulated annealing to converge on a match for each element of the training set. The matching set, or "antibody set", is then produced by combining the matches with the highest fitness, and pruning elements that are too closely related to each other. Once the algorithm ends (either due to detecting convergence or iteration limits), it is used to classify the data set.

3 ImagePimp

The starting point for our research was an existing application known as ImagePimp, which was built to do basic image manipulation and testing. There was an AINet module in this program, which would load a training set from a text file and attempt to classify the existing image based on that training set. After some work splitting the AINet code into an independent class, an attempt was made to generalize the codebase. While doing so, it became apparent that the code was extremely fragile and was implementing an algorithm much closer to a random walk than the AINet technique we desired.

4 Benchmarking Harness

When confronted by the AINet code that existed, it immediately became clear that a testing harness must be implemented as quickly as possible so that the performance of successive iterations could be measured and regressions could be easily detected. As we had been provided with several sets of classified data in text file format, scripts were written to select a randomize portion of the classified data as a training set, and to generate an unclassified copy of the original data set. A database was built which could track these benchmarks by recording the revision from the source control system as well as relevant parameters, like the size of the Antibody set, and the number of iterations allowed for convergence of the Antibody set. The benchmarking utility used 4 data sets of classified and decoded picture data: 2 "ground_training" sets of 3 dimensional data, the "iris" set with 4 dimensions, and the "wine" set with 16 dimensions. Later a 2-d set

Name	Iterations	Round	Percent Correct	Stddev	Trials
ground_training1	5	0.0042	79.56	15.95	14
ground_training2	5	0.0027	69.81	7.08	14
iris	5	0.7792	4.67	11.43	14
wine	5	0.5982	0.60	0.50	14

Table 1: Initial Results

was generated based the SPIR problem[1]. Finally, the AINet class was given a driver function that could be executed independent of the GUI and allow scriptable selection of input and output files, as well as some simulation parameters. The initial results were not

5 Differences from AIS

As noted, the existing AINet code was much closer to a simulated annealing algorithm than the AIS code described in [1]. The training set and the data set were read into arrays of antigens, at which point the following algorithm was used:

1. Initialization: A random set of Antigens (AbBase) of size AbScale is generated.
2. Classification: Every Antigen is classified against the training set by finding the element that is closest, with distance computed by the taxicab metric:

$$D = 1 \sum_{i=0}^{d} |Ab_i - Ag_i|$$

   where d represents the number of dimensions, and Ag_i and Ab_i represent the i^{th} element of the Antigen and Antibody in question. An affinity is calculated as $\frac{1}{D}$.
3. Clonal Expansion: Copies of all elements were made based on the affinity, where elements with high affinities were more likely to reproduce, and placed in the clonal_population.
4. Affinity Maturation: These copies are modified by adding the affinity to each element of the cloned Antibody.
5. Clonal Suppression: All elements below a certain distance are removed from the clonal_population.
6. Apoptosis/Metadynamics: All elements greater than a certain distance are removed, as well.
7. Network Reconstruction: The AbBase and clonal_population lists are concatenated and reclassified.
8. Network Supression: This is the same as Clonal Supression
9. Introduce Diversity: Random Antibodies are added to the clonal_population.
10. A semi-random selection of BaseScale number of Antibodies is removed and saved in AbBase.
11. Repeat until you run out of iterations or manage a successful convergence.

This algorithm differs substantially from the intended one, but the process of translating between one and the other does bring some insight into the originally desired algorithm. Currently, the code has absorbed the following modifications to the algorithm:

1. Initialization: A random set of Antigens (AbBase) is generated. The number of antigens is a user specified fraction of the size of the training set. This change allows us to control for varying data sets in a more reasonable manner.

2. Classification: Every Antigen is classified against the training set by finding the element that is closest, where the Euclidean distance is computed with:

$$D = |Ab - Ag| = \sqrt{\sum_{i=0}^{d} (Ab_i - Ag_i)^2}$$

   An affinity is calculated as $\frac{1}{D}$. This distance metric is much more stable over increasing numbers of dimensions, as it grows more slowly, and it produces substantially better results.

3. Clonal Expansion: This is unchanged. Future places to look for improvements would be to strengthen the guided randomness of the process here.

4. Affinity Maturation: These copies are modified by adding a random number between $\pm\frac{\text{Affinity}}{2}$ to each element of the array. Unlike the previous function, this does not result in vectors that are monotonically increasing, so it is possible for them to converge. This non-convergence is most likely the reason that early versions of the program showed substantially worse results after 100 iterations than after 5.

5. Clonal Suppression: This has been temporarily removed, as it will just be repeated shortly with a slightly larger set.

6. Apoptosis/Metadynamics: All elements greater than a certain distance are removed, as well. This also seems correct, though we may wish to have the Apoptosis threshold scale based on the range of the input set to get better coverage.

7. Network Reconstruction: The AbBase and clonal_population lists are concatenated and reclassified.

8. Network Supression: All elements below a certain distance are removed from the clonal_population. This is the same.

9. Introduce Diversity: Random Antibodies are added to the clonal_population to provide up to $\frac{1}{\zeta}$ Antibodies, where ζ is a user adjustable parameter representing the percentage of antibodies that need to be pruned in each pass. This allows us to maintain a relatively constant antibody pool, but adjustments in the Clonal Expansion process may allow us to allow more dynamism here.

10. Prune the ζ Antibodies with the lowest Affinity.

11. Repeat until you run out of iterations or manage a successful convergence.

This approach brings substantially better results with the 3-dimensional data, and slightly better results with the 4-dimensional data. Larger elements are still outside of our grasp at the moment. With 2-d data, the approach converges quickly, even with very low compression factors (See Table 2.)

Name	Iterations	Compression	Percent Correct	Standard Dev	Trials
ground_training1	50	0.00	99.98	0.02	5
ground_training2	50	0.00	99.62	0.00	1
spir	50	0.20	50.00	0.00	5
wine	50	0.00	0.79	0.28	5
ground_training1	5	0.0040	99.97	0.02	5
ground_training2	5	0.0040	94.36	9.68	5
iris	5	0.0040	33.33	0.00	5
wine	5	0.0040	0.90	0.45	5

Table 2: Final Results

6 Future Extensions

The primary difference between the algorithm as specified and the algorithm implemented here is that this algorithm is using only a single Clonal memory store, while the target algorithm keeps a separate one for each Antigen. This is why the Network Suppression and Clonal Suppression steps are not redundant there as they were here. This seems counterintuitive at first, as it transforms the training set into a larger dataset, not a smaller one, but this is only temporary, as the size of the final antigen set will still be controlled by the compression ratio specified by the user. This seems likely to improve the classification results seen on the SPIR problem, where all of the generated Antibodies end up with a single classification, though it possible that there are undiscovered bugs in the suppression or expansion steps that are causing this.

7 Conclusion

While the goal of classifying hyperspectral imagery has not been realized, there has been substantial improvement in the classification performance by the algorithm. In addition, a substantial infrastructure of tools for generating and manipulating the test corpora as well as automating benchmarks has been produced, and the codebase itself has been bashed into a maintainable shape that also allows some sane debugging to occur.

References

[1] L. N. de Castro and F. J. V. Zuben, *Data Mining: a Heuristic Approach*, ch. aiNet: An Artificial Immune Network for Data Analysis. Idea Group Publishing, 2001.

[2] D. Dasgupta, *Artificial immune systems and their applications*. Springer, 1999.

[3] S. J. Russell and P. Norvig, *Artificial Intelligence: A Modern Approach*. Pearson Education, 2003.