# A Comprehensive Benchmark of the Artificial Immune Recognition System (AIRS)

Lingjun Meng[a], Peter van der Putten[b][1], Haiyang Wang[a]

(a): Network Center, Shandong University, P.R.China
{mlj,why}@sdu.edu.cn
(b): Leiden Institute of Advanced Computer Science, Leiden University, the Netherlands
putten@liacs.nl

**Abstract**: Artificial Immune Systems are a new class of algorithms inspired by how the immune system recognizes, attacks and remembers intruders. This is a fascinating idea, but to be accepted for mainstream data mining applications, extensive benchmarking is needed to demonstrate the reliability and accuracy of these algorithms. In our research we focus on the AIRS classification algorithm. It has been claimed previously that AIRS consistently outperforms other algorithms. However, in these papers AIRS was compared to benchmark results from literature. To ensure consistent conditions we carried out benchmark tests on all algorithms using exactly the same set up. Our findings show that AIRS is a stable and robust classifier that produces around average results. This contrasts with earlier claims but shows AIRS is mature enough to be used for mainstream data mining.

## 1 Introduction

In recent years there has been a rapid growth in the interest in Artificial Immune Systems for applications in data mining and computational intelligence [4]. The immune system is sometimes called the ‘second brain’ for its abilities to recognize new intruders and remember past occurrences. Simulating the immune system or translating immune system mechanisms into machine learning is an interesting topic on its own. However, to be accepted as a candidate algorithm for data mining applications, the source of inspiration for these algorithms is not really an issue of interest. Rather empirical evidence is needed that these algorithms produce high quality, reliable results over a wide variety of problems compared to a range of other approaches, without the need of expert fine-tuning.

In this paper we present such benchmarking results. Given that we are interested in applicability of artificial immune systems for real world data mining, and that classification is one of the most important mining tasks, we focus on the Artificial Immune Recognition System (AIRS) algorithm. AIRS was introduced in 2001 as one of the first immune systems approaches to classification [14] and seemed to perform reasonably well on various classification problems. Until then, several papers have

[1] Corresponding author

been published dealing with AIRS benchmarking [5,6,10,15]. However, in our opinion these approaches were relatively limited, given that comparisons were made on a small number of data sets and algorithms, and that the benchmark results were sourced from literature rather than produced under exactly the same conditions as for the AIRS algorithm.

In contrast to the previous work mentioned, all our experiments have been run from scratch, to guarantee consistent experimental conditions. This includes applying AIRS on a wide range of representative real-world datasets with large differences in number of instances, attributes and classes.

The remainder of this paper is organized as follows. Section 2 provides some background on the AIRS algorithm. Section 3 reports on the set up of the benchmarking experiments. Section 4 discusses the results and identifies directions for future benchmarking research and section 5 concludes the paper.

## 2 The Artificial Immune Recognition System (AIRS)

The recognition and learning capabilities of the natural immune system have been an inspiration for researchers developing algorithms for a wide range of applications. This section introduces some basic immune system concepts and provides the history and background behind the AIRS algorithm for classification.

### 2.1 Natural immune systems

The natural immune system offers two lines of defense, the innate and adaptive immune system. The innate immune system consists of cells that can neutralize a predefined set of attackers, or ‘antigens’, without requiring previous exposure to them. The antigen can be an intruder or part of cells or molecules of the organism itself. In addition, higher animals like vertebrates possess an adaptive immune system that can learn to recognize, eliminate and remember specific new antigens. This is accomplished by a form of natural selection. The bone marrow and thymus continuously produce lymphocytes and each of these cells can counteract a specific type of antigen. Now if for example a B-cell lymphocyte encounters an antigen it codes for, it will produce antibody molecules that neutralize the antigen and in addition a large number of cloned B-cells are produced that code for the same antigen (‘clonal expansion’ or ‘clonal selection’). The immediate reaction of the innate and adaptive immune system cells is called the primary immune response. A selection of the activated lymphocytes is turned into sleeper memory cells that can be activated again if a new intrusion occurs of the same antigen, resulting in a quicker response. This is called the secondary immune response [4].

### 2.2 Artifical immune systems

Natural immune systems have inspired researchers to develop algorithms that exhibit adaptivity, associative memory, self – non self discrimination and other aspects of immune systems. These artificial immune system algorithms (also known as immuno-computing algorithms) have been applied to a wide range of problems such as biological modeling, computer network security & virus detection, robot navigation, job shop scheduling, clustering and classification [4].

The Artificial Immune System algorithm (AIRS) can be applied to classification problems, which is a very common real world data mining task. Most other artificial immune system research concerns unsupervised learning and clustering. The only other attempt to use immune systems for supervised learning is the work of Carter [2]. The AIRS design refers to many immune system metaphors including resource competition, clonal selection, affinity maturation, memory cell retention, and so on. AIRS builds on the concept of resource limited clustering [3,13].

According to the introductory paper, AIRS seems to perform well on various classification and machine learning problems [14]. Watkins claimed “the performance of AIRS is comparable, and in some cases superior, to the performance of other highly-regarded supervised learning techniques for these benchmarks”. Later on, Goodman, Boggess, and Watkins investigated the “source of power for AIRS” and its performance on multiple-class problems. They claim “AIRS is competitive with the top five to eight classifiers out of 10-30 best classifiers on those problems”, “it was surprisingly successful as a general purpose classifier” and it “performed consistently strong across large scope of classification problems” [5,6].

## 2.3 AIRS: the algorithm

From a data mining point of view, AIRS is a cluster-based approach to classification. It first learns the structure of the input space by mapping a codebook of cluster centers to it and then uses k-nearest neighbor on the cluster centers for classification. The attractive point of AIRS is its supervised procedure for discovering both the optimal number and position of the cluster centers.

In AIRS, there are two different populations, the Artificial Recognition Balls (ARBs) and the memory cells. If a training antigen is presented, ARBs (lymphocytes) matching the antigen are activated and awarded more resources. Through this process of stimulation, mutation and selection a candidate memory cell is selected which is inserted to the memory cell pool if it contributes enough information. This process is repeated for all training instances and finally classification takes place by performing a nearest neighbor search on the memory cell population.

To describe the AIRS algorithm in detail, let us assume we have a training data set $X$ containing $n$ labelled instances $ag_i = \{x_i, t_i\} \in \mathbb{R}^d \bullet \mathbb{Z}$ with $x_i$ an input with $d$ attributes and $t_i$ a one dimensional target class ($i=1,2,\ldots,n$). The algorithm goes through the following steps [15]:

*1. Initialization*

First all the data items will be normalized so that the affinity of every two training instances $ag_i$ and $ag_j$ is in the range [0,1]. In AIRS, the affinity is usually represented by Euclidean distance over the attributes. We assume the set $MC$ as the memory cell pool containing $m$ memory cells: $MC=\{mc_1,mc_2,\ldots,mc_m\}$, and set $AB$ as the ARB-population containing $r$ ARBs: $AB=\{ab_1,ab_2,\ldots,ab_r\}$, with $mc_j=\{x_j^{mc},t_j^{mc}\}$, ($j=1,2,\ldots,m$); $ab_k=\{x_k^{ab},t_k^{ab}\}$, ($k=1,2,\ldots,r$). Then the memory cells pool *MC* and the ARB population *AB* are seeded by randomly adding training instances.

*2. Memory cell identification and ARB generation*
From now on, antigens (training instances) will be presented to the algorithm one by one. If an antigen $ag_i=\{x_i,t_i\}$ is presented to the system, the algorithm will identify a memory cell $mc_{match}=\{x_{match}^{mc},t_{match}^{mc}\}$ which has the same class label ($t_{match}^{mc}=t_i$) and lowest distance to $ag_i$. If there is no $mc_{match}$ available at this moment, just let $ag_i$ act as the $mc_{match}$. This $mc_{match}$ will then be cloned to produce new *mc* clones. First the attributes of $mc_{match}$ will be mutated with a certain probability. If any mutations occurred for this particular clone, the class label will be mutated as well with the same probability

*3. Competition for Resources and Development of a Candidate Memory Cell*
At this moment, there are a set of ARBs including $mc_{match}$, mutations from $mc_{match}$, and others from previous training. AIRS mutates these memory cell clones to generate new ARBs. The number of ARBs allowed to produce is calculated by the product of the hyper clonal rate, clonal rate (both default 10), and the stimulation level (1-distance to $ag_i$). The newly generated ARBs will be combined with the existing ARBs.

AIRS then employs a mechanism of survival of the fittest individuals within the ARB population. First, each ARB will be examined with respect to its stimulation level when presented to the antigen. In AIRS, cells with high stimulation responses that are of the same class as the antigen and cells with low stimulation response that are not of the same class as the antigen are rewarded most and allocated with more resources. The losers in competing for resources will be removed from the system. Then the ARB population consists of only those ARBs that are most stimulated and are capable in competing for resources.

Then the stop criterion is evaluated. The stop criterion is reached if the average stimulation value of every class subset of AB is not less than the stimulation threshold (default 0.8). Then the candidate memory cell $mc_{candidate}$ is chosen which is the most stimulated ARB of the same class as the training antigen $ag_i$. Regardless whether the stop criterion was met the algorithm proceeds by allowing the ARBs the opportunity to proliferate with more mutated offspring. This mutation process is similar to the mutation of phase 2, with a small exception: the amount of offspring than can be to produced is calculated by the product of stimulation level and the clonal rate only. If the evaluation criterion was not met in the last test, the process will start again with the stimulation activation and resource allocation step. Otherwise the algorithm will stop.

*4. Memory Cell Introduction*
Now if $mc_{candidate}$ is more stimulated by the antigen than $mc_{match}$, it will be added into the memory cell pool. In addition, if the affinity value between and $mc_{candidate}$ and $mc_{match}$ is also less than the product of the affinity threshold (average affinity between all training items) and the affinity threshold scalar (a parameter used to provide a cut-

off value, default 0.8), which means $mc_{candidate}$ is very similar to $mc_{match}$, $mc_{candidate}$ will replace $mc_{match}$ in the set of memory cells. By this mechanism, better classifying memory cells can replace existing memory cells so that the data reduction capabilities of the algorithm are improved. Training is completed now for this training instance $ag_i$. and the process is repeated from step 2 for the next instance.

5. Classification

With the training completed, the evolved memory cell population $MC=\{mc_1, mc_2\ldots,mc_m\}$ $(m<n)$ will be used for classification using *k*-nearest neighbor. The classification for a test instance will be determined by the majority vote of the *k* most stimulated memory cells.

## 3 Benchmark Experiments

The goal of the benchmark experiments is to evaluate the predictive performance of AIRS in a real world application setting. We assume that our users are non data mining experts, e.g., business users, who may lack knowledge or time to fine-tune models. To ensure consistency, the experiments for all classifiers were carried under exactly the same conditions, in contrast to some earlier published work on AIRS.

We selected data sets with varying number of attributes, instances and classes, from simple toy data sets to difficult real world learning problems, from the UCI Machine Learning and KDD repositories [1]. The TIC data sets are derived from the standard TIC training set by downsampling the negative outcomes to get an even distribution of the target. In addition, TIC5050S only contains the most relevant according attributes according to a subset feature selection method [11,12].

In the experiments, we selected some representative, well known classifiers as challengers. These classifiers include naive Bayes, logistic regression, decision tables, decision trees (C45/J48), conjunctive rules, bagged decision trees, multi layer perceptrons (MLP), 1-nearest neighbor (1-NN) and 7-nearest neighbor (7-NN). This set of algorithms was chosen because they cover most of the algorithms used in business data mining and correspond to a variety of classifier types and representations - instance based learning, clustering, regression type learning, trees and rules, and so on. Furthermore we added classifiers that provide lower bound benchmark figures: majority class simply predicts the majority class and decision stumps are decision trees with one split only. For AIRS we chose the 1 and 7 nearest neighbor versions of the algorithm. We used the Java version of AIRS by Janna Hamaker [7] and the WEKA toolbox for the benchmark algorithms [9].

All experiments are carried out using 10-fold stratified cross validation. The data is divided randomly into ten parts, in each of which the target class is represented in approximately the same proportions as in the full dataset. Each part is held out in turn and the classifier is trained on the remaining nine-tenths; then the classification accuracy rate is calculated on the holdout validation set. Finally, the ten classificaton accuracy rates on the validation sets are averaged to yield an overall accuracy with standard deviation. To test the robustness of classifiers under real world conditions, all classifiers were run with default settings, without any manual fine-tuning

| | Sonar | W. Breast Cancer | Wave form | Iris | Iono sphe re | Pima diabet es | Ger man credi t | TIC 5050 | TIC5 050s |
|---|---|---|---|---|---|---|---|---|---|
| **Majority Class** | 53.4 ± 1.7 | 65.5 ± 0.5 | 33.8 ± 0.1 | 33.3 ± 0.0 | 64.1 ± 1.4 | 65.1 ± 0.4 | 70.0 ± 0.0 | 49.7 ± 0.4 | 49.7 ± 0.4 |
| **1-NN** | 86.6 ±7.0 | 95.3 ±3.4 | 73.6 ± 1.3 | 95.3 ± 5.5 | 86.3 ± 4.6 | 70.2 ± 4.7 | 72.0 ± 3.1 | 55.9 ± 7.8 | 59.9 ± 5.1 |
| **7-NN** | 80.8 ± 7.8 | 96.6 ± 2.2 | 80.1 ± 1.1 | 96.7 ± 3.5 | 85.2 ± 4.3 | 74.7 ± 5.0 | 74 ± 4.1 | 61.1 ± 3.2 | 65.4 ± 8.4 |
| **Decision Stump** | 73.1 ± 8.3 | 92.4 ± 4.4 | 56.8 ± 1.5 | 66.7 ± 0.0 | 82.6 ± 4.8 | 71.9 ± 5.1 | 70 ± 0.0 | 68.5 ± 4.7 | 68.5 ± 4.7 |
| **C45/J48** | 71.2 ± 7.1 | 94.6 ± 3.6 | 75.1 ± 1.3 | 96 ± 5.6 | 91.5 ± 3.3 | 73.8 ± 5.7 | 70.5 ± 3.6 | 68.1 ± 5.5 | 69.1 ± 4.4 |
| **Naive Bayes** | 67.9 ± 9.3 | 96.0 ± 1.6 | 80.0 ± 2.0 | 96.0 ± 4.7 | 82.6 ± 5.5 | 76.3 ± 5.5 | 75.4 ± 4.3 | 62.8 ± 6.4 | 68 ± 3.3 |
| **Conj. Rules** | 65.9 ± 8.7 | 91.7 ± 4.5 | 57.3 ± 1.3 | 66.7 ± 0 | 81.5 ± 5.4 | 68.8 ± 8.67 | 70.0 ± 0 | 67.4 ± 3.7 | 68.3 ± 4.5 |
| **Bagging** | 77.4 ± 0.1 | 95.6 ± 3.1 | 81.8 ± 1.4 | 94 ± 5.8 | 90.9 ± 4.4 | 74.6 ± 3.6 | 74.4 ± 4.9 | 59.9 ± 5.8 | 68.4 ± 4.1 |
| **Logistic** | 73.1 ± 13.4 | 96.6 ± 2.2 | 86.6 ± 2.3 | 96 ± 5.6 | 88.9 ± 4.9 | 77.2 ± 4.6 | 75.2 ± 3.4 | 62.7 ± 4.6 | 66.5 ± 3.4 |
| **MLP** | 82.3 ± 10.7 | 95.3 ± 2.6 | 83.6 ± 1.7 | 97.3 ± 3.4 | 91.2 ± 2.8 | 75.4 ± 4.7 | 71.6 ± 3.0 | 60.7 ± 4.3 | 65.4 ± 4.7 |
| **Decision Table** | 74.5 ± 8.2 | 95.4 ± 2.7 | 73.8 ± 1.6 | 92.7 ± 5.8 | 89.5 ± 4.5 | 73.3 ± 3.6 | 72.2 ± 4.1 | 61.9 ± 4.5 | 69.1 ± 5.7 |
| **AIRS-1** | 84.1 ± 7.4 | 96.1 ± 1.8 | 75.2 ± 1.7 | 96 ± 5.6 | 86.9 ± 3.1 | 67.4 ± 4.6 | 68 ± 5.1 | 56.8 ± 4.4 | 55 ± 6.5 |
| **AIRS-7** | 76.5 ± 8.4 | 96.2 ± 1.9 | 79.6 ± 2 .2 | 95.3 ± 5.5 | 88.6 ± 5.0 | 73.6 ± 3.5 | 71.4 ± 3.1 | 57.8 ± 5.5 | 59.1 ± 6.1 |

**Table 1.** Benchmark comparison of average and standard deviation of validation set accuracy for 10 folds.

## 4 Results and Discussion

The results of the experiments can be found in Table 1. With respect to the worst case classifiers we highlight some interesting patterns. Almost all classifiers outperform majority vote. The comparison with decision stumps is more striking. For example, for all data sets with the exception of the waveform data set the conjunctive rules classifier does not perform better than decision stumps. Other examples are the TIC data sets: none of the classifiers other than C45 on TICTRAIN5050s perform better than decision stumps. This demonstrates the power of a very simple decision rule in a real world black box modeling environment (see also [8]).

To get a better picture on the relative performance of AIRS we compare it to the average classifier performance (excluding decision stump and majority vote). AIRS-1 performs better than average on 3 of these 9 datasets. AIRS-7 performs better than average on 6 of these 9 datasets. This conflicts with the claims made in earlier studies that were cited in section 2.2. We also made some comparisons to the IB-k algorithms, because these may be closest to a trained AIRS classifier. AIRS-1 improves on IB-1 more often than the other way around; this is probably due to the fact that AIRS-1 provides some useful generalization. However IB-7 performs better than AIRS-7 on all of the data sets. AIRS-7 performs better than AIRS-1on 7 out of 9 data sets. Using more clusters may give better results but not to the extent that IB-7 can be beaten (basically as many cluster centers as data points).

That said, with the exception of AIRS-1 on German credit data, the AIRS algorithms produce at least around average results. This suggests that AIRS is a mature classifier that delivers reasonable results and that it can safely be used for real world classifications tasks.

In our future work we want to make a more extensive investigation into what claims can be made at the various significance levels. In addition, assuming that there will be no classifier that outperforms all other classifiers across all problem domains, we want to investigate on what kind of data sets AIRS performs well, for example by relating properties such as data set size to performance of AIRS relative to other algorithms. Furthermore, we intend to study the relation between AIRS and other algorithms by looking at patterns of performance across algorithms.

## 6. Conclusions

In this paper we have presented benchmark results for the AIRS immuno-computing algorithm and provided directions for interpretation of these results. We are interested in immuno-computing because it is one of the newest directions in biologically inspired machine learning and focused on AIRS because it can be used for classification, which is one of the most common data mining tasks.

To our knowledge this the first benchmark of AIRS that compares AIRS across a wide variety of data sets and algorithms, using a completely consistent experimental set up rather than referring to benchmark results from literature. In contrast to earlier claims, we find no evidence that AIRS consistently outperforms other algorithms. However, AIRS provides stable, near average results so it can safely be added to the data miner’s toolbox. Whether the relative complexity of the algorithm is justified by demonstrating outstanding performance in specific, identifiable problem domains remains a question for further research.

## Acknowledgements

We would like thank Janna Hamaker for providing the AIRS source code and Andrew Watkins for his assistance with understanding the details of the AIRS algorithm.

## References

[1] Blake, C. and C. Merz. ‘UCI Repository of machine learning databases’, http://www.ics.uci.edu/~mlearn/ MLRepository.html . 1998

[2] Carter, J. H. The immune systems as a model for pattern recognition and classification. Journal of the American Medical Informatics Association 7(1), 28-41, 2000

[3] De Castro, L. N. and F. von Zuben. The clonal selection algorithm with engineering applications. In: D. Whitley, D. Goldberg, E. CantuPaz, L. Spector, I. Parmee, and H. Beyer (eds.): Proceedings of Genetic and Evolutionary Computation. San Francisco, CA, Morgan Kaufman. Pp. 36-37, 2000

[4] De Castro, L.N. and J. Timmis. Artificial Immune Systems: a New Computational Intelligence Approach. Springer Verlag, 2002

[5] Goodman, D. E. , L. Boggess, and A. Watkins. Artificial Immune System Classification of Multiple-Class Problems. In Artificial Neural Networks in Engineering (ANNIE), 2002

[6] Goodman, D. E. , L. Boggess, and A. Watkin. An Investigation into the Source of Power for AIRS, an Artificial Immune Classification System. In Proceedings of the International Joint Conference on Neural Networks (IJCNN), 2003

[7] Hamaker, J. and A. Watkins. Artificial Immune Recognition System (AIRS), Java source code, 2003

[8] Holte, R.: 1993, ‘Very Simple Classification Rules Perform Well on Most Commonly Used Datasets’. Machine Learning 11, 63–91.

[9] Ian H. Witten and Eibe Frank Data Mining. Practical machine learning tools with Java implementation. Morgan Kaufmann, San Francisco, 2000

[10] Marwah G. and L. Boggess. Artidicial immune systems for classification: Some issues. In 1st International Conference on Artificial Immune Systems, pp. 149-153, 2002

[11] van der Putten, P. and M. van Someren (eds) . CoIL Challenge 2000: The Insurance Company Case. Published by Sentient Machine Research, Amsterdam. Also a Leiden Institute of Advanced Computer Science Technical Report 2000-09. June 22, 2000.

[12] van der Putten, P. and M. van Someren. A Bias-Variance Analysis of a Real World Learning Problem: The CoIL Challenge 2000. Machine Learning, vol. 57, iss. 1-2, pp. 177-195, Kluwer Academic Publishers, October 2004,

[13] Timmis, J. and M. Neal. A Resource Limited Artificial Immune System. Knowledge Based Systems 14(3/4), 121-130, 2001

[14] Watkins, A., AIRS: A Resource Limited Artificial Immune Classifier. M.S. thesis, Department of Computer Science. Mississippi State University, 2001

[15] Watkins, A., J. Timmis, and L. Boggess Artificial Immune Recognition System (AIRS): An Immune-Inspired Supervised Learning Algorithm. Genetic Programming and Evolvable Machines, 5 (3): 291-317, 2004