WAIRS: Improving Classification Accuracy by Weighting Attributes in the AIRS Classifier

Andrew Secker, Alex A. Freitas

Abstract— **AIRS (Artificial Immune Recognition System) has shown itself to be a competitive classifier. It has also proved to be the most popular immune inspired classifier. However, rather than AIRS being a classifier in its own right as previously described, we see AIRS more as a pre-processor to a KNN classifier. It is our view that by not explicitly classing it as such development of this algorithm has been rather held back. Seeing it as a pre-processor allows inspiration to be taken from the machine learning literature where such pre-processors are not uncommon. With this in mind, this paper takes a core feature of many such pre-processors, that of attribute weighting, and applies it to AIRS. The resultant algorithm called WAIRS (Weighted AIRS) uses a weighted distance function during all affinity evaluations. WAIRS is tested on 9 benchmark datasets and is found to outperform AIRS in the majority of cases.**

I. Introduction

AIRS is a supervised immune-inspired classification system capable of assigning data items unseen during training to one of any number of classes based on previous training experience. AIRS is probably the best known AIS for classification, having been developed in 2001 [1]. It has undergone a number of revisions and refinements in order to increase efficiency and increase accuracy [2, 3] and has been shown to work competitively on benchmark tests using standard public domain datasets [4, 5]. Indeed, when AIRS was tested on the well known Iris dataset, Pima Indians dataset, ionosphere dataset and Sonar dataset, AIRS was comparable with the fifth to eighth most successful classifiers found in the literature for three out of the four datasets (where the Pima Indians dataset was the exception) [6].

In the literature, AIRS is referred to as a classification algorithm. However, we believe it would be useful if this attitude was changed. Rather than a classification algorithm in its own right, we believe it would be more useful to think of AIRS as a pre-processor to a Nearest Neighbour (NN) or K-Nearest Neighbour (KNN) algorithm. In this case AIRS can be seen to perform an instance construction task, a common task in the data mining literature [7]. Instance construction is a data reduction technique that aims to summarise a training dataset by creating a set of typical instances or prototypes that best generalise those data. In this way the amount of training data are reduced, increasing the efficiency of the induction algorithm, and a correctly functioning instance construction algorithm will remove outliers and other noisy instances from the training data, thus increasing classification accuracy over the test set. Re-positioning AIRS as an instance construction algorithm can allow users to take inspiration from the lazy learning literature. As the subject of data pre-processing techniques for nearest neighbour algorithms has been around for many years there is a vast wealth of information available on this subject. The use of such ideas applied to AIRS in turn may give insight into improvements for AIRS.

It is the aim of this paper to do such a thing. By identifying a weakness in the current AIRS implementation, solutions can be found in the data mining literature. The algorithm can then be augmented with these changes and the resulting algorithm tested against benchmark data to determine whether the changes have had a positive effect.

In the following section the AIRS algorithm is briefly introduced and some points are raised concerning the similarities between the high-level procedure of the AIRS algorithm and other data reduction strategies applied to classification. Section III is concerned with introducing the reader to a number of data mining concepts pertinent to the research, this section includes some technical details of the chosen attribute weighting strategy. Section IV details the changes made to the AIRS algorithm and these are tested and evaluated in Section V. The final section contains a summary and some concluding remarks.

II. AIRS

For technical details regarding the AIRS algorithm, the reader is referred to the literature such as [1, 2] although it is worth giving a quick overview of the algorithm. The AIRS algorithm has three main stages. First there is an initial seeding phase in which randomly chosen data vectors are used to form an initial population of memory cells. These cells will be of different classes as dictated by the underlying data. The second phase is a training phase. During this phase the affinity between cells and training data items is computed, this affinity is based on a notion of Euclidean distance. Cells are stimulated based on this degree of match combined with a match between the training data and the cell's class. Processes of cloning and mutation take place based on this stimulation value. While cloning would have the effect of increasing the population size the number of cells is controlled by a resource allocation mechanism where ARBs (Artificial Recognition Balls) compete for resources based on their stimulation level. This competition for resources applies strong selective pressure on the population, as any ARBs surviving this stage will go on to produce offspring. The aim of this stage is to create a set of memory cells that best summarise the training data that has been seen, in order to maximise the classification accuracy and

Manuscript received March 15th 2007.

A. Secker and A. A. Freitas are affiliated with the Computing Laboratory at the University of Kent, Canterbury, Kent, UK CT2 7NF.
E-mail: a.d.secker@kent.ac.uk, a.a.freitas@kent.ac.uk

efficiency of the following stage. These memory cells will be derived from the best performing ARBs in this stage. The final stage of the AIRS algorithm is the classification stage, during which data items as yet unseen by the algorithm are assigned a class based on the result of the algorithm's training. This classification is performed by the well known nearest neighbour (NN) or K-nearest neighbour (KNN) procedure, the details of which are expanded in the following section.

Notice here how the AIRS algorithm pre-processes the data but the actual classification is left to a standard classification technique (NN or KNN). This follows the same template as other instance selection or instance reduction algorithms, such as IB2 [8] in which the raw training data set is reduced in size before the actual classification is performed by a standard lazy learning technique using those reduced data.

AIRS was compared in [1] against four standard datasets: iris, ionosphere, diabetes and sonar datasets from the UCI repository [9]. It was noted that AIRS was very competitive with other algorithms, achieving a higher classification accuracy than both C4.5 and a Bayesian approach over the diabetes dataset.

AIRS was updated to AIRS2 in [5] with five small changes being made, including slight updates to the mutation and cloning routines. AIRS2 was updated further to work as an efficient parallel system in [3, 4].

III. Data Mining Issues

Nearest neighbour classification algorithms use the available training set, or a subset of it, to classify unseen data. During training they do not infer any specific classification model, rather they defer any induction until a classification is requested. For this reason they are referred to as lazy algorithms rather than eager, such as decision tree building or rule induction. In essence, a simple nearest neighbour classifier will store all training data. When a new instance is to be classified it determines the most similar training instance and assign the new item the class of that instance. It is possible that the K most similar training instances will be retrieved and the new item assigned the most frequent class. In this case the algorithm is referred to as a K-nearest neighbour.

As every training instance is interrogated every time a classification is requested, the time taken to classify test instances can be intractable if the training set size is very large. Thus, a reduction in the size of the training set is commonly sought to increase efficiency. It is the aim of a data reduction algorithm to reduce the size of the training set, whilst still maintaining that dataset's characteristics and therefore its predictive ability. In the most simple case, instance selection [10], instances deemed redundant will be removed from the training set. The goal here is to remove as many training instances as possible whilst not significantly impacting the final classification accuracy. [11] describes how a concept (class) may be represented by a small number of typical instances of that concept. This approach is common in the literature, with a typical and easy to understand example being IB2 [8]. The IB2 algorithm selects only those instances from the training set that would have been misclassified when a KNN classifier is applied. This tends to leave only those instances close to class boundaries while redundant and noisy instances are removed.

Consider the example in Fig. 1 (A simplified version of one given in [8]). Two linearly separable classes are represented in two dimensions. The original training set is shown in Fig. 1 (A), while the reduced set on instances is shown in Fig. 1 (B). It can be seen that after the instance selection step, point X for example ,would still be classified as "-" based on its nearest neighbour, even though its neighbours are different between A and B. In this case the redundant instances have been removed. The goal of such an algorithm is, therefore, to remove as many instances as possible whist retaining the information about the class boundaries that was present in the original training set. I.e. there should be no difference in the predictive accuracy of an algorithm when it is using the original training data or the reduced set of instances for training.

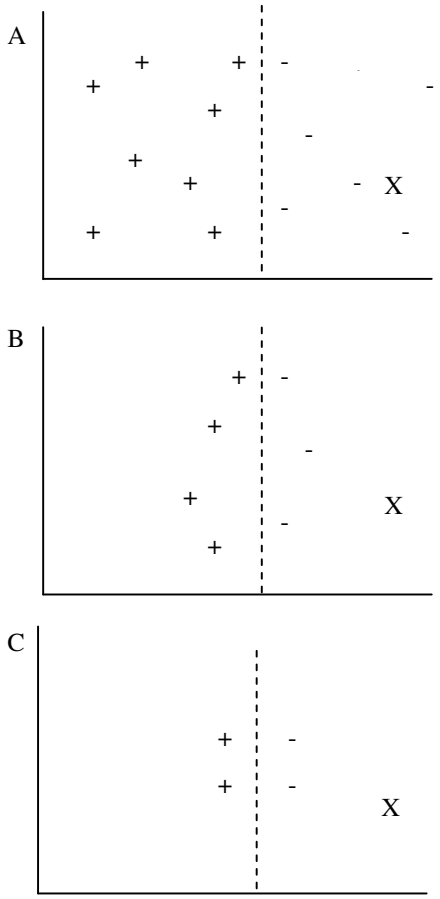

Fig. 1. Example of distribution of examples in a training set (A), the reduced set of examples after processing by an instance selection algorithm (B) and the reduced set of examples after processing by an instance construction algorithm (C).

The above example concerns instance selection, but a reduction in the size of the training set may be accomplished in one of two ways, by instance selection or instance construction [7]. In the case of instance construction the

original training set is used to determine a new training set, presumably of smaller size than the original, so that the original training data are discarded and the algorithm trained on this abstracted set. The important distinction here is that instance selection algorithms use a subset of the original instances while instance construction algorithms construct new instances which are not present in the original data. Fig. 1 (C) shows how an instance construction algorithm may represent the class boundary as Fig. 1 (A) but this time the instances used to represent each class have been moved (i.e. they are not present in the original training set) yet point X will still be correctly classified. This example shows that instance construction can be more efficient than instance selection as the algorithm may place instances where they are most efficient, rather than where the original training set dictates. Indeed because this constraint is not present, the constructed instances have the potential to better represent a class boundary and therefore have the potential to increase the classification accuracy compared with an instance selection approach.

Examining the high-level outline of the AIRS algorithm on Section II, it can be seen that AIRS effectively performs instance construction, rather than instance selection. ARBs are constructed to best represent the original training set, which is discarded. The size of the ARB set is presumably much smaller than the original training set. Not only does this procedure have the effect of increasing the efficiency of the classifier, but it can also serve to enhance the generality of the training set and thus improve the classification accuracy over test data. Hereafter we focus on instance construction.

There exist a number of changes that could be made to the basic AIRS algorithm to try to improve classification accuracy. For example, a wealth of different distance measures may be used. As standard, AIRS uses Euclidean distance, but this distance measure may not be the best for a given problem. Different distance measures have different inductive biases making them suitable to different kinds of data sets (Freitas and Timmis, 2006). [12] contains a discussion of such distance metrics and presents a comparison of these over three data sets. However, the testing of different distance metrics is not the concern of this study.

Predictor attributes should be normalised so that one attribute with a large range or generally high values does not dominate another attributes with a smaller ranges or values. Attribute normalisation is performed by default in AIRS already.

A. Attribute weighting

An important improvement to the standard KNN procedure is that of weighting attributes. It stands to reason that, in most cases, not all predictor attributes will contribute equally to a correct classification. Weighting schemes may be introduced such that irrelevant attributes are given a low weighting while relevant attributes are given a large weighting. These weightings are used when the distance between a training instance and a classification instance is computed such that relevant attributes exert a disproportionately large influence on the final distance value. In effect this weighting lengthens the axes of relevant attributes in Euclidean data space, while shrinking the axes of irrelevant attributes. In extreme cases, attributes can be assigned a weight of 0, meaning they are deemed totally irrelevant and as such are disregarded in the distance calculation. A special case of this is feature selection, where the weights 1 or 0 are assigned to attributes. This can have the advantage of reducing the dimensionality of the data, resulting in an increase in classification speed. However, as a binary weight is assigned the algorithm is judging the attribute to be either completely relevant or completely irrelevant. This may not reflect the actual data where features are likely to have varying degrees of relevance.

There are numerous examples in the literature where attribute weighting has been shown to be beneficial on some problems [13-17] while there is also a small amount of theoretical work concerning the calculation of optimum feature weights [18]. In addition to this, the addition of attribute weighting to AIS algorithms has been encouraged. In [19], the case is made that unweighted distance measures are likely to be a suboptimal choice of affinity function for AIS used as classifiers. The implementation of weighted distance functions is one of ten research directions suggested in that article.

The use of AIRS in situations where noisy or irrelevant attributes have been present has been briefly investigated before. In [20], the authors tested AIRS on an artificial problem and found a reduction in predictive accuracy of around 5% but with a large increase in the number of output cells, which is undesirable.

Computing the optimum weight for an attribute is a complex problem. There are two general methods for the discovery of weights; wrapper and filter (sometimes called performance bias and preset bias respectively). In the case of the wrapper approach, the predictive accuracy of the classification algorithm is used as a feedback to guide the weighting algorithm. The classification algorithm that will use the attribute set should provide a better estimate of accuracy than a separate measure that may have an entirely different inductive bias. However, the major disadvantage is the computational cost which results from calling the classification algorithm to evaluate each set of attribute weights [21]. In the investigation undertaken here, the expected computational costs are too high to use this approach and so a filter approach is preferred. A filter approach will use the available training data to adjust the attribute weights without using the classification algorithm for feedback. Thus filter methods are usually independent of the classification algorithm.

A great many attribute weighting algorithms exist in the literature. Considering only filter approaches, methods such as Continual Probabilities [22] and Class Projection [23] are found in the literature.

An empirical comparison of weighting algorithms is performed in [14] in which four common filter-based weighting algorithms are compared. The mutual information (MI) approach is seen to perform the best on average. The MI of two variables is the reduction in uncertainty of one

variable's value given the knowledge of the other [24]. In this case, the weighting changes as the mutual information between a predictor attribute and the instance's class change. Thus if an attribute provides absolutely no information about the class, its weight will be 0. The weight *w* of feature (attribute) *f* is computed as shown in Equation 1.

$$w(f) = \sum_{v \in V_f} \sum_{c_j \in J} p(c_j, x_f = v) \times \log\left(\frac{p(c_j, x_f = v)}{p(c_j) \times p(x_f = v)} \right) \quad (1)$$

There exists a problem with this metric in that one must determine the frequency with which an attribute value predicts a given class. This is meaningless over continuous attributes as there are an infinite number of values a predictor attribute may take. Instead, some kind of attribute discretization must be undertaken to assign each continuous attribute value to one discrete value (an interval of continuous values). In this way the probability of a discrete value predicting a class may be computed as shown in Equation 1.

B. Attribute Discretization

Discretization is the process of converting a numerical attribute into a symbolic attribute by partitioning the attribute domain [25]. Discretization of continuous attributes is fundamental to many decision tree algorithms and is therefore a well researched area in data mining [26]. Many decision tree algorithms such as ID3, C4.5 and CART all require binary splits at decision nodes [27]. The value at which this split occurs is usually determined by a discretization algorithm, although the difference here is that this discretization will occur in a dynamic manner as the tree is built, rather than occurring as a pre-processing step as occurs in nearest neighbour algorithms. As such there are commonalities with algorithms such as naïve Bayes in which discretization often occurs before the classification algorithm is run. In these cases the discrete values resulting may not be binary, rather, numerous partitions may be created if the data requires and it is this multiple splitting that is the concern of this section.

TABLE I
EXAMPLE OF NAÏVE DISCRETIZATION

Predictor attribute value	Class	Symbolic value
1	-	A
2	-	
3	-	B
4	+	
5	+	C
6	+	

The most straightforward method for discretization is to assign data instances to a partition based on user defined partition boundaries. However, this strategy causes two issues that may negatively impact the performance of the algorithm. Firstly the number of partitions may not naturally fit the data and secondly the partition boundaries that result may not naturally fit the data. Consider a case in which the data can be classified into two classes. A "low" value of a predictor attribute indicates one class while a "high" value indicates another. Forcing this data into three subsets may, incorrectly, force the position at which the boundary point occurs into one of the subsets. Notice in the example, Table I, how the data naturally partitions into two classes while forcing it into three symbolic, discrete values would lead to substandard performance as discrete value B contains data from two classes.

This example illustrates that it is generally beneficial for a discretization algorithm to choose the number of partitions and the partition boundaries in a data driven manner, i.e. discretization algorithms should be supervised. Many well-known discretization algorithms are supervised and as such fulfil the criteria above. Suitable choices include ChiMerge [28] and Vector Quantization [29] or Recursive Minimal Entropy Partitioning [30]. An empirical comparison of these was undertaken in [26]. The strategy of Fayyad and Irani is used in this paper as it is found to be common throughout the literature [31] and was seen to fare well in the above empirical evaluation on a number of standard public domain datasets. The chosen method uses the calculated entropy of each class to select the most suitable partition boundary, where the class information entropy can be calculated as shown in Equation 2.

$$E(A, T; S) = \frac{|S_1|}{S} \times \mathrm{Ent}(S_1) + \frac{|S_2|}{S} \times \mathrm{Ent}(S_2) \quad (2)$$

In Equation 2, *S* is a set of instances, *A* is a given feature (attribute) and *T* is a partition boundary. For feature *A*, the boundary T_{min} that minimises the entropy over all possible boundaries is selected [27]. The application of this will therefore result in a binary split, and the method can be applied recursively until a stopping criterion is met, in this case, a criterion based on the Minimum Description Length Principle.

IV. A NEW, ATTRIBUTE WEIGHTED VERSION OF THE AIRS2 ALGORITHM

Attribute weighting was added to the AIRS algorithm as a pre-processing stage. The Mutual Information weighting scheme was implemented according to Equation 1. This used the Fayyad and Irani's discretization technique as described in Section III.B and Equation 2. The code used to perform the discretization was taken from the open source "Bayesian Network Classifiers Toolbox (JBNC)" [32].

The data used to determine the weight of each attribute was the same training data used by AIRS. Once initialised, the weights do not change during the run of AIRS.

The standard Euclidean distance measure is changed to include the weight *w* of a feature *f* such that the distance *d* between two data instances, *x* and *y*, is determined as follows [16]:

$$d(x,y) = \sqrt{\sum_{f=1}^{n} w_f \times \left(x_f - y_f\right)^2} \quad (3)$$

While, in this paper, the concept of attribute weights was introduced in the context of the KNN classifer, it should be noted that the inclusion of attribute weights will affect the whole AIRS algorithm, not just the final KNN classification routine. This modified Euclidean distance measure is used whenever an affintiy is evaluated during the training stage.

It should be noted that while this investigation only uses Euclidean distance, other distance measures are available for AIRS [6]. Not only do these new distance measures inclue measures for continuous attributes (i.e. manhattan distance), they also include measures for symbolic data (Value Distance Metric) and mixed data (Hetrogeneous Euclidean-Overlap Metric). The intrioduction of these metrics was not only found to make AIRS more flexible as it could be used for more datasets, in some cases the use of a non-Euclidean distance measure was found to increase the accuracy of the classifier.

V. Computational Results

Experiments were carried out in order to determine how an AIRS with weighted features performed compared to a standard AIRS implementation. Java source code was obtained from the original author of the AIRS algorithm [33]. This source code was for the revised AIRS algorithm, AIRS2, as described in [2]. Both the WAIRS (Weighted-feature AIRS) and standard AIRS algorithms were run with the default parameters found in the code, i.e. no optimisation of parameters was performed. This makes the comparison between the two algorithms as fair as possible. The values of the parameters can be found in Table II.

A number of datasets were retrieved from the well-known UCI machine learning repository [9]. Due to the inability of AIRS to handle datasets in which continuous and discrete attributes are present, the chosen datasets used continuous attributes only. The datasets were temporarily discretized, as explained earlier, just for the sake of computing feature weights in a pre-processing phase. Once those weights have been determined, they are fixed throughout the run of WAIRS – i.e., the algorithm learns a classification model from the originally continuous features. The "waveform 40" dataset is as used in [14] and [16] to evaluate the quality of weighting methods, as this includes 19 artificially irrelevant attributes. Along similar lines the "Iris+4" dataset is the standard iris dataset with an additional 4 irrelevant attributes. The remaining datasets are left as standard.

TABLE II
Algorithm Parameters

Parameter	Value
Clonal rate	10
Mutation rate	0.7
Affinity threshold	0.2
Stimulation threshold	0.95
Resources	200
Hypermutation rate	10
K value in KNN classifier	3

A 10-fold cross validation approach was taken to estimate the predictive accuracy of the algorithms. In this approach, data instances are randomly assigned to one of 10 approximately equal size subsets. At each iteration, all but one of these sets are merged to form the training set while the classification accuracy of the algorithm is measured on the remaining subset. This process is repeated 10 times, choosing a different subset as the test set each time until all data instances have been used 9 times for training and once for testing. The weighting algorithm is run once per fold using the same training data as the main AIRS algorithm. The final predictive accuracy is computed over all folds in the usual manner but dividing the number of correct classifications taken over all folds by the number of data instances in all folds. As there is a certain amount of non-determinism involved both in the random partitioning of the data and the running of AIRS, at each of the 10 iterations of the cross-validation procedure both WAIRS and AIRS2 were run 10 times, varying the random seed used to create the initial population in each run.

Table III shows the mean classification accuracy obtained when running AIRS on the selection of publicly available datasets. The AIRS and WAIRS columns show the mean predictive accuracy of the respective algorithm. The 'significance' column shows the probability of the accuracy obtained for AIRS2 and WAIRS do not differ. A value in this column of <0.05 is deemed significant while a value of <0.01 is highly significant. This figure was obtained by comparing the results of each algorithm using a two-tailed unpaired Student's t-test [34, 35]. Significant results are shown in bold type.

TABLE III
Comparison of Classifiaction Accuracy

Dataset	AIRS2	WAIRS	Significance
Iris+4	88.07%	94.73%	**6.2226E-07**
Waveform 40	75.92%	81.91%	**3.5782E-17**
Iris	95.00%	94.53%	0.17303348
Waveform	80.24%	81.59%	**3.5147E-06**
Wine	95.74%	97.47%	**0.00010374**
Sonar	77.10%	79.34%	**0.02448068**
Ionosphere	88.40%	87.81%	0.12493906
Glass	60.77%	60.03%	0.31628134
Diabetes	71.52%	71.29%	0.32420265

The results show that WAIRS achieves significantly higher predictive accuracy in both datasets where irrelevant attributes have been artificially injected into the data. The increase in accuracy is really quite striking in both cases. Out of the remaining 7 datasets, WAIRS achieves a significantly higher predictive accuracy over 3 of the 7 datasets. Of the remaining 4 datasets, any difference in the accuracies reported was not found to be statistically significant. On no occasion did WAIRS result in a significantly lower classification accuracy than AIRS2.

The main test of this algorithm was whether the classification accuracy would increase over AIRS2 when run on data with attributes known to be irrelevant. Thus it can be concluded that the weighting procedure is working as expected. The results shown here contrast with those found

in [20], in which the accuracy was found to decrease when AIRS was tested using a dataset with known irrelivet attributes.

Over the remaining datasets, those taken directly from the UCI repository produced mixed, although no negative, results. While these public datasets are comprised of real-world data, they are often pre-processed, including the selection of attributes particularly suited to the target data mining task. As such it is likely that no or few irrelevant features are included in the data, thus it was not expected that any large increase in accuracy would be seen. The fact that there was no significant difference in the classification accuracy over these datasets was pleasing and suggests the chosen strategy may be robust.

It should be noted that in previous publications where datasets have been used to evaluate AIRS2, the reported accuracies differ from the figures in Table III. This is due to differences in the testing strategy and in the parameters used. In [1, 36] a 5-fold cross validation approach was taken, while in [4] different datasets were tested differently with 5, 10 or 13 folds used. However in the case of the Ionosphere dataset, no cross validation was applied and a single test/training set was used. In all cases the tests were repeated three tines and the averages taken. In addition to this,. the parameters were optimised for performance on each dataset separately. In the case of [2], no details of the testing strategy or the parameter values are given, but taking information from [4] it is believed that test procedures and parameters were the same as the previous papers. Thus is likely that these two factors conspire to result in the accuracies reported here that are slightly lower than those previously reported. the interested reader wishing to assess the quality of AIRS classification accuracy in the wider context of machine learning should therefore consult previously published papers such as [5] which contain tables comparing AIRS with other state-of-the-art classifier systems in a fairer context.

A. *Data reduction*

As discussed in Section III, when performing classification with a KNN type algorithm one major goal is a reduction in training data while maintaining accuracy. We have shown in the previous subsection that predictive accuracy is either improved or not significantly impacted but it is of interest to observe the number of final memory cells produced after training to determine whether WAIRS produces fewer memory cells than AIRS2. The results are shown in Table IV. All results obtained were statistically highly significant (probability less than 1%) and so a significance column would be redundant.

From Table IV, it can be seen that in 5 of the 9 datasets WAIRS resulted in a decrease in the number of data items used to perform the KNN classification. There does not, however, seem to be much consistency in these results, for example, an increase in classification accuracy (Table III) does not necessarily predict an increase/decrease in B-cells produced. Bold values show a reduction in memory cells. Rather it is expected that this may be dataset dependent. Notice how the numbers of cells for both the Iris datasets have reduced while the numbers of cells for both the Waveform datasets has increased. This, as yet, cannot be explained. What is important is that the increase in the number of cells is not exclusively for the datasets that resulted in the better classification accuracy. Significantly, however, the massive increase in memory cells reported in [20] was not repeated when datasets containing noisy attributes are compared to the same dataset which does not contain noisy attributes (Iris vs. Iris+4 and Waveform vs. Waveform 40).

TABLE IV
COMPARISON OF THE NUMBER OF MEMORY CELLS LEFT AFTER TRAINING

	Final memory cells	
Dataset	AIRS2	WAIRS
Iris+4	124.56	**37.57**
Waveform 40	3086.3	3513.9
Iris	49.61	**37.22**
Waveform	3330.4	3472.6
Wine	137.38	**133.16**
Sonar	127.41	153.32
Ionosphere	144.79	159.86
Glass	89.49	**74.95**
Diabetes	494.05	**422.41**

VI. CONCLUSION

In this paper we have highlighted that the well known AIS-based classification algorithm, AIRS, can be thought of as a pre-processor to a KNN algorithm functioning as a powerful instance construction algorithm. The case was made that inspiration could be taken from the data mining literature and the performance of AIRS could be enhanced by determining individual weights for data attributes. An enhanced affinity function was utilised to make use of the attribute weights, with the same function being used to compute the distances during KNN classification. The results of a comparison between this new algorithm, WAIRS, and the previously published AIRS2 showed that WAIRS outperformed AIRS2 significantly in 5 out of 9 datasets. Of the remaining 4 datasets no significant difference between the results of both algorithms was found. In cases where a dataset contains irrelevant attributes, WAIRS is likely to outperform AIRS, while in cases where no irrelevant attributes are present both algorithms work equally well.

It was noted in the background section that filter type attribute weighting schemes such as that implemented here can not take account for attribute interaction as it is independent of the classification algorithm used. An obvious improvement would be to use a wrapper approach to further improve attribute weighting.

As part of the aim of this paper was to illustrate that AIRS may be better classified as an instance construction algorithm rather than a classifier in itself, the next logical progression of this work would be to compare AIRS2/WAIRS to other instance construction algorithms in terms of both data compression and the ability of each to maintain or improve classification accuracy.

REFERENCES

[1] A. Watkins, "AIRS: A Resource Limited Artificial Immune Classifier", Masters Dissertation, Department of Computer Science, Mississippi State University, MS. USA, 2001.

[2] A. Watkins and J. Timmis, "Artificial Immune Recognition System (AIRS): Revisions and Refinements" in *1st International Conference on Artificial Immune Systems (ICARIS 2002)*, Canterbury, UK, 2002, pp.173-181.

[3] A. Watkins and J. Timmis, "Exploiting Parallelism Inherent in AIRS" in *3rd International Conference on Artificial Immune Systems (ICARIS 2004)*, Catania, Sicily, 2004, pp.427-438.

[4] A. Watkins, "Exploiting Immunological Metaphors in the Development of Serial, Parallel, and Distributed Learning Algorithms", PhD. Thesis, Computer Science, University of Kent, Canterbury, England, 2005.

[5] A. Watkins, et al., "Artificial Immune Recognition System (AIRS): An Immune-Inspired Supervised Learning Algorithm", *Genetic Programming and Evolvable Machines*, 5(3), pp. 291-317, 2004.

[6] J. Hamaker and L. Boggess, "Non-Euclidean Distance Measures in AIRS, an Artificial Immune Classification System" in *2004 Congress on Evolutionary Computation (CEC 2004)*, 2004, pp.1067-1073.

[7] H. Liu and H. Motoda, *Instance Selection and Construction for Data Mining*. Kluwer Academic Publishers, 2001.

[8] D. W. Aha, et al., "Instance-Based Learning Algorithms", *Machine Learning*, 6(1), pp. 37-66, 1991.

[9] D. J. Newman, et al., "UCI Repository of machine learning databases", Retrieved September 2006 from: http://www.ics.uci.edu/~mlearn/MLRepository.html. 1998.

[10] H. Liu and H. Motoda, "On Issues of Instance Selection", *Data Mining and Knowledge Discovery*, 6(2), pp. 115-130, 2002.

[11] J. Zhang, "Selecting Typical Instances in Instance-Based Learning" in *9th International Conference on Machine Learning (ICML 2000)*, Aberdeen, Scotland, UK, 1992, pp.470-479.

[12] S. Saltzberg, "Distance Metrics for Instance-Based Learning" in *6th International Symposium on Methodologies for Intelligent Systems (ISMIS-91)*, Charlotte, USA, 1991, pp.339-408.

[13] D. W. Aha, "Tolerating Noisy, Irrelevant and Novel Attributes in Instance-Based Learning Algorithms", *International Journal of Man-Machine Studies*, 6(1), pp. 267-287, 1992.

[14] D. Wettschereck, et al., "A Review and Empirical Evaluation of Feature Weighting Methods for a Class of Lazy Learning Algorithms", *Artificial Intelligence Review*, 11, pp. 273-314, 1997.

[15] G. Demiroz and H. A. Güvenir, "Genetic Algorithms to Learn Feature Weights for the Nearest Neighbor Algorithm" in *BENELEARN-96*, 1996, pp.117-126.

[16] D. Wettschereck and D. W. Aha, "Weighting Features" in *First International Conference on Case Based Reasoning Research and Development*, 1995, pp.347-358.

[17] D. Wettschereck and T. G. Dietterich, "An Experimental Comparison of the Nearest-Neighbor and Nearest-Hyperrectangle Algorithms", *Machine Learning*, 19(1), pp. 5-27, 1995.

[18] C. X. Ling and H. Wang, "Computing Optimal Attribute Weight Settings for Nearest Neighbour Algorithms", *Artificial Intelligence Review*, 11, pp. 255-272, 1997.

[19] A. A. Freitas and J. Timmis, "Re-visiting the Foundations of Artificial Immune Systems for Data Mining", *To appear in IEEE Transactions on Evolutionary Computation*, pp. 2006.

[20] L. Boggess and J. S. Hamaker, "The Effect of Irrelevant Features on AIRS, an Artifcial Immune-Based Classifier" in *Intelligent Engineering Systems through Artifcial Neural Networks (AINNIE)*, 2003, pp.219-224.

[21] P. Langley, "Selection of Relevant features in machine Learning" in *AAAI Fall Symposium on Relevance*, New Orleans, 1994, pp.1-5.

[22] R. H. Creecy, et al., "Trading MIPS and Memory for Knowledge Engineering", *Communications of the ACM*, 35, pp. 48-64, 1992.

[23] C. Stanfill and D. Waltz, "Toward Memory-based reasoning", *Communications of the ACM*, 29(12), pp. 1213-1228, 1986.

[24] T. M. Cover and J. Thomas, *Elements of Information Theory*. Wiley, 1991.

[25] J. W. Grzymala-Busse, "Data reduction: discretization of numerical attributes", in *Handbook of Data Mining and Knowledge Discovery*, W. Klösgen and J. M. Zytkow Ed. Oxford University Press, 2002, pp. 218 - 225.

[26] J. Dougherty, et al., "Supervised and Unsupervised Discretization of Continuous Features" in *12th International Conference on Machine Learning*, San Francisco, USA, 1995,

[27] P. Perner and S. Trautzsch, "Multi-Interval Discretization Methods for Decision Tree Learning" in *SSPR/SPR*, 1998, pp.475-482.

[28] R. Kerber, "ChiMerge: Discretization of Numeric Attributes" in *10th National Conference on Artificial Intelligence*, San Jose, USA, 1992, pp.123-128.

[29] T. Kohonen, *Self-Organization and Associative Memory*. Springer-Verlag, 1989.

[30] U. M. Fayyad and K. B. Irani, "Multi-interval Discretization of Continuous-Valued Attributes for Classification Learning" in *13th international joint Conference on Artificial intelligence (IJCAI '93)*, 1993, pp.1022-1027.

[31] K. M. Ting, "Discretisation in Lazy Learning Algorithms", *Artificial Intelligence Review*, 11, pp. 157-174, 1997.

[32] J. Sacha, "JBNC - Bayesian Network Classifiers Toolbox", Retrieved September 2006 from: http://sourceforge.net/projects/jbnc/. 2004.

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

[34] H. L. Alder and E. B. Roessler, *Introduction to Probability and Statistics*. W. H. Freeman, 1968.

[35] J. H. Creighton, *A First Course in Probability Models and Statistical Inference*. Springer-Verlag, 1994.

[36] A. Watkins and L. Boggess, "A New Classifier Based on Resource Limited Artificial Immune Systems" in *Congress on Evolutionary Computation (CEC 2002)*, Honolulu, USA, 2002, pp.1546-1551.