Control and Cybernetics

vol. **39** (2010) No. 2

Evolving ensembles of linear classifiers by means of clonal selection algorithm[*]

by

Michał Bereta[1] and Tadeusz Burczyński[1,2]

[1]Institute of Computer Science, Cracow University of Technology
Cracow, Poland
[2]Department for Strength of Materials and Computational Mechanics
Silesian University of Technology, Gliwice, Poland

Abstract: Artificial immune systems (AIS) have become popular among researchers and have been applied to a variety of tasks. Developing supervised learning algorithms based on metaphors from the immune system is still an area in which there is much to explore. In this paper a novel supervised immune algorithm based on clonal selection framework is proposed. It evolves a population of linear classifiers used to construct a set of classification rules. Aggregating strategies, such as bagging and boosting, are shown to work well with the proposed algorithm as the base classifier.

Keywords: artificial immune systems, clonal selection, linear classifiers, bagging, boosting.

1. Introduction

This work presents a novel immune algorithm, which is an example of applying the general framework of clonal selection with suppression based on usefulness (a novel measure of artificial immune cell quality) to create new computational techniques. Clonal selection is one of the most important paradigms of artificial immune systems (AIS) (Dasgupta, 1998). The general schema of clonal selection algorithm is presented in Fig. 1. The algorithm is inspired by the evolutionary process of B–lymphocytes in the immune system of higher vertebrates. In the presence of antigens the B–cells become stimulated. The intensity of stimulation depends on how close the B–cell matches the antigen (on the molecular level). The most stimulated cells are cloned and mutated, the number of clones depends on the stimulation. The useless cells are removed from the system. The hypermutation process, i.e., the mutation at very high rates, leads potentially to a population of lymphocytes that recognise and react against the antigens

[*]Submitted: November 2008; Accepted: September 2009.

more efficiently. In AIS the problems (e.g., the training samples) are treated as antigens, whereas possible solutions are represented as lymphocytes. Although the process of developing the immune response is a complex biological process, not fully understood yet, the main concept is the basis for creating the clonal selection algorithms. The general paradigm and its main steps, common for all clonal selection algorithms, are presented in Fig. 1. The algorithm proposed in this work is based on this paradigm. However, novel ideas are introduced.

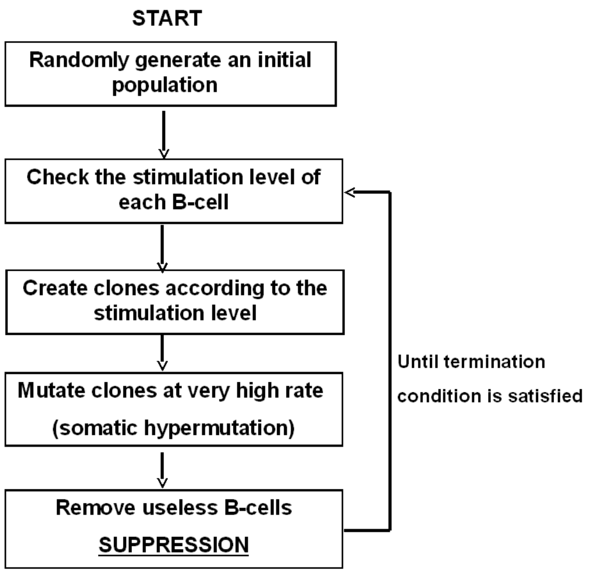
START
Randomly generate an initial population
Check the stimulation level of each B-cell
Create clones according to the stimulation level
Mutate clones at very high rate (somatic hypermutation)
Remove useless B-cells SUPPRESSION
Until termination condition is satisfied

Figure 1. The general schema of Clonal Selection Algorithm.

There are several features in the proposed algorithm that contribute to its novelty and originality. It assumes hyperplanes as the form of lymphocyte representation. It evolves a population of hyperplanes, solving a multiclass classification task. It is a supervised learning algorithm, as it makes use of class information in the form of the sample class labels during training. In its basic form it is formulated for the two–classes case, then it is extended to multiclass case. The final classification is based on voting among rules, which use weighted attributes and are created by means of the evolved linear classifiers. Since the proposed approach is formulated as a clonal selection algorithm, all the steps of the general clonal selection schema are defined with respect to the assumed representation of lymphocytes as linear classifiers. Linear classifiers (artificial lymphocytes) are trained to solve the problem not only by competition, but first of all by cooperation, which is due to the proper formulation of the suppression step. The idea of suppression based on usefulness is utilised. Usefulness is this context has the meaning of artificial immune cell quality. This concept was proposed in Bereta and Burczyński (2007) and further developed in Bereta and Burczyński (2006). Proper definition of the *usefulness* of a given lymphocyte for the whole population is crucial for the evolution of the population with desired properties. This approach is different from the commonly used suppression based on similarity, in which if there are too similar lymphocytes, the worst of them are removed. As it was the case in the cited works, also in this paper it is shown that suppression based on *usefulness* of the lymphocytes allows to develop populations of different, interesting properties.

The proposed algorithm is very unstable (i.e., very sensitive to the change in the training set) and for that reason it is especially suitable for the usage in construction of aggregated classifiers, which are very often better in solving difficult classification problems than single classifiers. Aggregated classifiers, also referred to as ensemble of classifiers, are discussed in Section 3.6.

1.1. Related work

Evolutionary computation methods have been applied to evolve a population of optimal classifiers. The approaches used differ by the type of the evolutionary algorithm used (Genetic Algorithm (GA), Genetic Programming (GP), etc.) and the classifier evolved (linear classifiers (LC), Neural Networks (NN), etc.). An approach that resembles the method presented in this paper is the one proposed in Pal, Bandyopadhyay, and Murthy (1998) and later further developed in Bandyopadhyay, Pal, and Aruna (2004). The similarity is that in Pal, Bandyopadhyay, and Murthy (1998) a population of linear classifiers is evolved (however, by means of genetic algorithm instead of clonal selection algorithm). The final classification of the unknown object is done based on the decisions of all the linear classifiers. However, there are not many similarities between the algorithm from Pal, Bandyopadhyay, and Murthy (1998) and the one proposed here. The main difference is that in the cited work each individual in the population encodes all the linear classifiers in the final aggregated classifier. In Immune Hyperplane Algorithm (IHA), each artificial lymphocyte is a single linear classifier and the whole population cooperates to solve the problem, so that the solution is the aggregation of all the artificial lymphocytes, the number of which differs from task to task, due to the suppression step. The proper definition of usefulness in the suppression step makes the population size in IHA adjusted – for simple problems, there are only a few lymphocytes needed. What should be stressed is that this population size adjustment is done already during the evolutionary process, starting from the very first iteration. Following this, it not necessary for the user to set the number of the linear classifiers for the problem. This is not the case in Pal, Bandyopadhyay, and Murthy (1998). Additionally, the usage of GA means that the individuals in the population compete first of all. There is also a need to set the maximum number of linear classifiers that a single individual can encode. The procedure of removing some of the linear classifiers is performed after the evolutionary process finishes. This step resembles in some sense the suppression step of IHA, however, it is performed at the end, which means that IHA, which removes useless LCs starting from the first iteration, will probably find the proper population size more effectively. The advantage comes from the fact that IHA uses the Michigan approach to solution encoding instead of the Pittsburgh approach, which is somewhat more popular in evolutionary algorithms. In Bandyopadhyay, Par, and Aruna (2004) a solution for the problem of estimating the proper population size is solved by means of the multi–objective evolutionary algorithm. The algorithm tries to minimize the number of misclassified samples and to minimize the number of linear classifiers. Several multi–objective techniques have been compared in Bandyopadhyay, Pal, and Aruna (2004) yielding better results than for the version of the algorithm with a constant number of LCs encoded by each individual. However, multi–objective optimization problems are hard to solve. For instance, weighting several objectives and combining them into one fitness function, eligible for use in GA, is very sensitive to the values of the weights and it is almost impossible to select a universal set of weights, optimal for each problem. The proposed IHA avoids the issue of solving multi–objective optimization problem by proposing a novel approach based on clonal selection paradigm and properly defined usefulness.

The ideas of using evolutionary computation to create ensembles of classifiers appear in many other works. In Wang and Wang (2006) a genetic algorithm is proposed to perform a search for appropriate weights. The weights are assigned to each of the training samples, which are then used to create an ensemble of classifiers. Good results in the problem of face detection are reported. In Kim and Oh (2008) a hybrid genetic algorithm is proposed for classifier ensemble selection. The proposed approach includes local search operations to improve the offspring. The proposed algorithm focuses not only on optimization of the decision function of the aggregated classifier (i.e., the combination algorithm) but also makes an effort to optimize the selection step of the candidate classifiers from the pool of available classifiers. The problem of selecting optimal classifiers to construct the final combined classification system is addressed also in Ruta and Gabrys (2005). In Jing and Zhang (2003) face recognition problem is solved by a combination of linear classifiers. The rational weights for the classifies are generated by means of a genetic algorithm with a fitness function based on a novel criterion, the maximum complementariness criterion. Other works that report successful applications of combined classifiers and novel evolutionary computation methodologies to create such ensembles are Gabrys and Ruta (2006), Zhang and Bhattacharyya (2004) or Kim, Street, and Menczer (2006).

The rest of this paper is structured as follows. First, the idea of a linear classifier is briefly reminded. Next, the proposed algorithm is described in detail. The generalization ability of the proposed method is discussed and possible ways of improvement are proposed, such as committee voting, bagging and boosting. Very promising results for several benchmark databases are presented in Section 4 and finally, conclusions are drawn in Section 5.

2. Linear classifiers

Let us assume that the training data consist of N–dimensional samples and the class labels are given in the form of $T = \{(\mathbf{x}_i, y_i), i = 1,.., M\}$, where $\mathbf{x}_i = (x_{i1}, x_{i2}, ..., x_{iN})$ is the vector of real valued features (attributes), M is the number of samples in the training set, and $y_i \in \{1, -1\}$ is the label of the class, to which x_i belongs. Linear classifiers are natural discriminators in the case when the samples are linearly separable, i.e. there exists a hyperplane $\mathbf{w} = \{w_0, w_1, ..., w_N\}$ such that $\mathbf{w} \cdot \mathbf{x}_i + w_0 > 0$ and $\mathbf{w} \cdot \mathbf{x}_i + w_0 < 0$ for all $\mathbf{x}_i$ from class 1 and -1, respectively. The element w_0 of $\mathbf{w}$ is often refereed to as *bias.* As it is always possible to extend the training input vector $\mathbf{x}_i$ to $\mathbf{x}'_i = (1, \mathbf{x}_i) = (1, x_1, x_2, ..., x_N)$, the following notation can be assumed, without losing the generality of the discussion: $\mathbf{w} \cdot \mathbf{x}_i > 0$ for all $\mathbf{x}_i$ from class 1, and $\mathbf{w} \cdot \mathbf{x}_i < 0$ for all $\mathbf{x}_i$ from class -1. In the case of data that are not linearly separable, one single plane separating all training samples cannot be found. The most common approach to deal with nonlinear problems is to use a nonlinear transformation of the original problem to a new feature space, most likely of higher dimensionality, in which the problem is linearly separable. The transformation can be achieved, for example, by means of appropriate kernel function (e.g., in Support Vector Machines - SVM, Vapnik, 1995) or multiple layers in neural networks (e.g., in Multi–Layer Perceptrons (MLP) or in Radial Basis Functions (RBF), Bishop, 1995). The problem with this approach is the choice of the appropriate transformation, i.e. the proper kernel function, the proper number of hidden neurons, etc. There exists no easy solution for these problems.

The approach proposed is different. The goal is to find a set (a population) of linear classifiers, among which none can solve the problem separately, but only in cooperation with other classifiers. Fig. 2 shows the idea. The data presented in Fig. 2 are not linearly separable, i.e., there exists no single line which separates the two classes. However, without any nonlinear transformation, it is possible to separate the two classes with two lines, $\mathbf{w}_1$ and $\mathbf{w}_2$. The final classification has to take into account on which side of *each* line a given subsample is. It can be viewed as performing two **linear** transformations, and constructing a set of rules of the following type: **IF** $\mathbf{w}_1 \cdot \mathbf{x}_i > 0$ **AND** $\mathbf{w}_2 \cdot \mathbf{x}_i < 0$ **THEN** $y_i = -1$. The output of the algorithm proposed is a population of linear classifiers, which can serve as a base for creation of such classification rules. The proposed algorithm is well suited to the general framework of clonal selection with the suppression based on properly defined *usefulness.* As in all previous algorithms, based on this framework (Bereta and Burczyński, 2006, 2007), an important characteristic of the proposed algorithm is the ability to determine the population size dynamically during evolutionary process, i.e., for simple problems small populations are expected. In classification problems, in the case of linearly separable data, a population consisting of only one lymphocyte (one hyperplane) is expected.

Clonal selection paradigm has been relatively seldom used to create supervised learning algorithms for multiclass classification problems. The most known, successful and influential ones are Watkins, Timmis and Boggess (2004), Watkins (2005) or Carter (2000). The approach proposed in this work proves that the creation of such algorithms is not only possible, but also can bring very satisfying results. Even more rarely linear classifiers have been used in artificial immune systems as artificial lymphocytes. The work presented in Ando and Iba (2003) seems to be an exception. It is, however, completely different from that presented here.

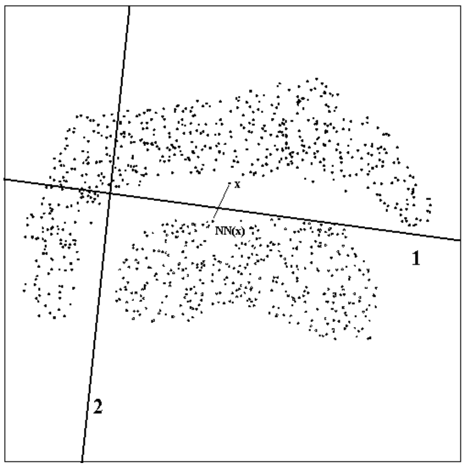

Figure 2. Line 1 is better than line 2 (it yields less misclassifications), however, both lines have to be considered together to separate the data. Example **x** is misclassified by line 2, but the pair (**x**, $NN^*(\mathbf{x})$) is not misclassified and does not decrease the usefulness of line 2.

3. Immune hyperplane algorithm

3.1. Definition of usefulness

In order to solve wide range of classification problems, also nonlinear, the proposed algorithm, Immune Hyperplane Algorithm (IHA), evolves a population of linear classifiers, which solve the problem through cooperation, i.e., none of them is globally optimal. They are locally optimal, and cooperate globally. This means that suppression should remove classifiers, which do not contribute to the global solution. This general definition of usefulness is, however, not so obvious to formulate in detail. In the case of linear classifiers, none of them has class label appointed to it, as none of them can solely decide about the final classification. The problem is that the influence of a single linear classifier is valid for the whole space, i.e., it divides the whole space into two subspaces. Thus, as each classifier has to be locally tuned, checking the number of training samples, which are correctly classified by a particular classifier does not work. As an example, consider the situation presented in Fig. 2. Line 1 is definitely better than line 2, in the sense that it yields less misclassifications globally. On the other hand, both lines are equally important from the point of view of cooperative solution of the problem. This example explains why suppression, which takes into account the total number of misclassifications of a given hyperplane, is not a good choice. Favoring hyperplanes, which make few errors, produces globally optimal planes, which try to solve the problem independently, and this is not the goal. In this place it is worth mentioning that during evolution of the population of hyperplanes, no classification rules (as presented earlier) are created. The rules are created after the evolutionary process, based on the already existing linear classifiers.

Fortunately, there is a straightforward way to define the usefulness of a given hyperplane lymphocyte. To achieve that, the algorithm does not work on original data, but rather on samples matched in pairs. Assuming two–class classification case, in each pair one sample is from the 1 class and the other is from the -1 class. The original set of training samples $T = \{(\mathbf{x}_i, y_i), i = 1..M\}$ is used to create a new training set in the following form:

$$T_{pairs} = \{\{(\mathbf{x}_i, y_i), (\mathbf{x}_j, y_j)\} : i = 1..M, j \in \{1..M\}, \\ NN(\mathbf{x}_i) = \mathbf{x}_j, y_i \neq y_j\} \tag{1}$$

where $NN(x_i)$ is the nearest neighbor of x_i. It means that for each training sample, its nearest neighbor from the other class is found and the pair constructed from these samples is added to T_{pairs}.

The form of representation of the training data as T_{pairs} has very useful implications. It is very easy to observe that by counting the number of correctly classified pairs, it is possible to assess the usefulness of a given linear classifier in a way which leads the population to the desired configuration. A given pair $\{(\mathbf{x}_i, y_i), (\mathbf{x}_j, y_j)\} \in T_{pairs}$ is correctly classified by a given linear classifier $\mathbf{w}$ if $signum(\mathbf{w} \cdot \mathbf{x}_i) = y_i$ and $signum(\mathbf{w} \cdot \mathbf{x}_j) = y_j$, where $signum(u)$ returns 1 if $u > 0$ and -1 if $u \leq 0$. Counting the number of correctly classified pairs for each plane does not have some of the disadvantages of counting the total number of misclassifications. For example, line 1 in Fig. 2 intersects several segments corresponding to pairs in T_{pairs}. Training sample $\mathbf{x}$ is misclassified by line 2, however, the segment related to $\mathbf{x}$ and its NN^* (nearest neighbour from the other class) is not intersected by line 2, thus, this case is considered neither as error nor as correct classification with respect to line 2, and it does not decrease the usefulness of line 2.

The concept of T_{pairs} construction is in a sense similar to the idea of dipoles — pairs of feature vectors. An example of the usage of dipoles in the construction of an ensemble of Dipolar Neural Networks can be found in Kretowska (2008). However, the idea of T_{pairs} used in the approach proposed in this work is much simpler.

To summarize, a given lymphocyte (hyperplane) is considered as useful for the whole population, if there exists at least one pair from T_{pairs}, which is correctly classified only by this lymphocyte. If there exists no such pair, the linear classifier is considered as useless. It means that if the subtask of the overall classification task performed by the given lymphocyte can be distributed among other individuals in the population, the given classifier can be removed without any loss to the performance of the whole system. On the other hand, a given classifier can be considered as useful even if it is a very poor global classifier, i.e., it yields more misclassifications than correct classifications.

3.2. The two class case

In this section, basic formulation of the proposed algorithm is given. The training examples $\mathbf{x}_i$ from the set T are considered to be vectors of real valued features (attributes). The basic formulation of the IHA is for the two–classes case. The proposed algorithm goes as follows:

```
IMMUNE HYPERPLANES ALGORITHM
$T$: A set of training samples from two classes, 1 and $-1$.
$Pop$: Population of lymphocytes, i.e., linear classifiers.
$LC_i$: i–th linear classifier in $Pop$.
$startPopSize$: The starting size of $Pop$.
$mutRange$: The parameter of mutation.
$clonesNum$: The number of clones of each $LC_i$.
$newNum$: The number of newly generated linear classifiers in each iteration.
START
  1. Create the $T_{pairs}$ set.
  2. Generate the starting population $Pop$ of the size $startPopSize$.
  3. For each $LC_i$ in $Pop$, $i = 1, .., |Pop|$, check which pairs from $T_{pairs}$ are
     correctly classified by $LC_i$.
  4. Sort $Pop$ in the descending order according to the number of correctly
     classified pairs from $T_{pairs}$.
  5. Create $clonesNum$ clones for each $LC_i$ in $Pop$, $i = 1, .., |Pop|$.
  6. Mutate each clone created in step 5 and check correctly classified pairs
     from $T_{pairs}$ for each mutated clone.
  7. Add $newNum$ newly generated linear classifiers to $Pop$ and check correctly
     classified pairs from $T_{pairs}$ for each newly generated classifier.
  8. Sort $Pop$ in the descending order according to the number of correctly
     classified pairs from $T_{pairs}$.
```

9. Perform suppression. Starting from the worst (i.e., the one with the smallest number of correctly classified pairs from T_{pairs}) LC_i in sorted Pop, for each LC_i temporarily remove it from Pop. If there exist no loss in the total number of correctly classified pairs by all classifiers in Pop, remove LC_i definitely.
10. Repeat from step 5 until termination condition is satisfied.

END

After creation of the T_{pairs} set, a random initial population is generated. To alleviate the learning process, the generation process assures that the generated linear classifier separates at least two learning samples, one from each class. This is achieved by randomly selecting two training samples, $\mathbf{x}_{-1}$ and $\mathbf{x}_1$, from class -1 and 1, respectively. Next, the new weight vector $\mathbf{w}_i$ of the i–th generated linear classifier, LC_i, is constructed as a hyper plane perpendicular to the direction of vector $\mathbf{x}_1 - \mathbf{x}_{-1}$, and intersecting the middle point between $\mathbf{x}_{-1}$ and $\mathbf{x}_1$. From that, $\mathbf{w}_i = \mathbf{x}_1 - \mathbf{x}_{-1}$ and the bias of $\mathbf{w}_i$, w_{i0} is determined as $w_{i0} = -\sum_{j=1}^{N} w_{ij} * (x_{-1j} + x_{1j})/2$. The new lymphocytes generated in step 7 are constructed in the same way. In all experiments presented in this paper, it is assumed that every LC_i produces the same number of clones, regardless of the number of correctly classified pairs from T_{pairs}, which could be, in fact, interpreted as stimulation level and used to promote better classifiers by allowing them to produce a bigger number of clones. It is not used here, however, as the algorithm works well without this property. The mutation is performed in a very simple manner. Parameter $mutRange$ describes the maximum allowed decrease or increase of the value of each w_{ij} of a given $\mathbf{w}_i$, i.e. $w_{ij} = w_{ij} + mutRange * (2 * random(0,1) - 1)$, where $random(0,1)$ is a random value, uniformly sampled from the range $[0,1]$, $j = 0,..,N$ (bias is also mutated).

The results of applying the proposed algorithm to artificial 2D data sets are presented in Fig. 3. Each example is a two–class classification problem. The algorithm is able to find the appropriate number of lines separating the classes cooperatively.

3.3. Dealing with noise

The formulation of the algorithm from Section 3.2 is very optimistic. The naive assumption was that even in the case of nonlinear boundaries between classes, the samples from different classes are not mixed, i.e., the class distributions do not overlap and there is no noise or incorrect labels in the training set. If the distributions do overlap or some incorrect labels exist in the training set, the construction of the set T_{pairs} becomes erroneous, in the sense that there are pairs from T_{pairs} entirely included inside one of the classes. This leads to populations, in which many classifiers are adapted only to these noisy pairs. This undesirable effect is illustrated in Fig. 4, in which two classes with artificially added noise

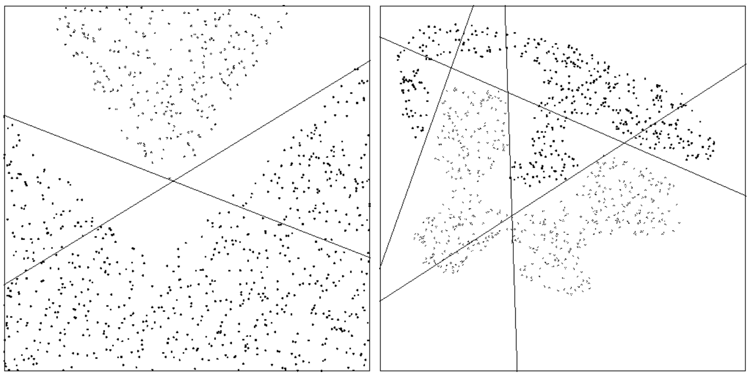

Figure 3. Example results of Immune Hyperplane Algorithm applied to 2D artificial data (classification problems with two classes). The proposed method is able to find the proper population size for the given problem.

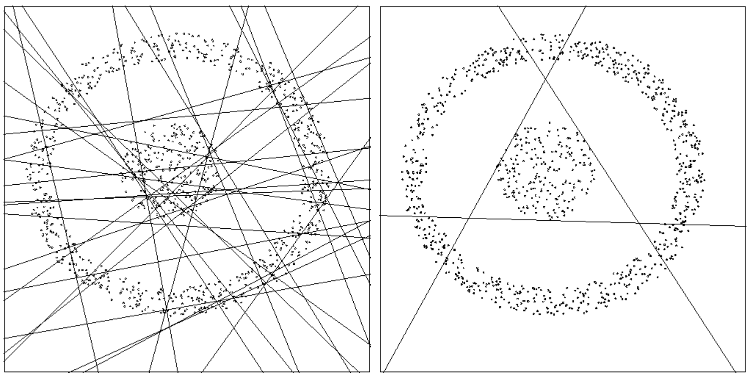

Figure 4. The population evolved on data with added 5% of noise (left). The population evolved on noisy data with modified method of T_{pairs} construction (right).

are presented, together with a population of evolved linear classifiers. Note that there are many unnecessary lines, due to the existence of misleading pairs.

Fortunately, there is a simple modification that can be introduced into the algorithm, allowing for the limitation of the undesirable effect of the misleading pairs. Such pairs can be in many cases discovered during the first step of the algorithm, in which T_{pairs} set is constructed. This procedure is modified as follows. For each $\mathbf{x}_i$ its nearest neighbor among samples from the other class is found, as previously. Additionally, the nearest neighbor from both classes, i.e., from the whole training set T, is found, too. If both nearest neighbors are in fact the same training sample, $\mathbf{x}_i$ does not create **any** pair which could be used during training and no new element is added to T_{pairs}. Although this simple procedure does not eliminate all undesirable pairs, it definitely makes IHA very resistant to noisy samples. Fig. 4 presents an example result of applying modified IHA to noisy data. Note that the modification has no effect if there is no noise present in the data.

3.4. Classification rules

As it was already mentioned, a final classification of an unknown sample by means of a set of hyperplanes requires checking on which side of every hyperplane a given sample is. Assuming there are K hyperplanes in the final population in IHA, an Immune Hyperplane Classifier (IHC) can be constructed, which is represented as a set of rules of a type

$$\textbf{IF } \mathbf{w}_1 \cdot \mathbf{x}_i > 0 \textbf{ AND } \mathbf{w}_2 \cdot \mathbf{x}_i < 0 \textbf{ AND } \ldots\ \mathbf{w}_K \cdot \mathbf{x}_i > 0 \textbf{ THEN } y_i = -1. \quad (2)$$

Having the population of hyperplanes already evolved, it is relatively easy to create a set of such rules. The procedure is straightforward. Each training sample $\mathbf{x}_i \in T,\ i = 1, .., M$ is checked against each linear classifier $LC_j,\ j = 1, .., K$, whether it satisfies a condition $\mathbf{x}_i \cdot \mathbf{w}_j > 0$. Thus, a premise of an l–th rule is in the form of a vector of boolean values $\mathbf{Premise}_l = (p_{l1}, p_{l2}, ..., p_{lK})$, where $p_{lj} = true$ if $\mathbf{x}_i \cdot \mathbf{w}_j > 0$ and $p_{lj} = false$ otherwise, $j = 1, ..., K$, and the conclusion of the l–th rule, $Conl_l$ is the class label, i.e., either 1 or -1. Additionally, each rule has an additional field, $Perc_l$, in which a percentage of samples correctly classified training by this rule is stored. After the training sample $\mathbf{x}_i$ is checked against all LC_j, a vector *tempPremise* of binary values is formed. If it is equal to a premise of the already existing, l–th, rule and $y_i = Concl_l$, i.e., $\mathbf{x}_i$ is correctly classified by the l–th rule, then $Perc_l$ is increased properly. If the class labels do not match or if there is no rule with a premise equal to *tempPremise*, a new rule is constructed with the premise *tempPremise* and class label y_i. It means that there can exist rules with the same premises but different conclusions (different class labels). This is not a problem, as each rule has its *Perc* value, allowing for deciding, which rule should make the decision about the final classification, in the case of a test sample satisfying both rules. Fig. 5 presents class boundaries found by means of the set of rules constructed as described.

It can be observed even in these 2D examples that during testing of an unknown sample $\mathbf{x}_{test}$ with the set of rules, it is possible that *tempPremise* constructed by checking $\mathbf{x}_{test}$ against each LC_j, does not match a premise of any known rule. Areas of such test samples are presented in darker colours in Fig. 5. Obviously, it is possible to use the idea of partial matching, popular in many artificial immune algorithms. If *tempPremise* of $\mathbf{x}_{test}$ is interpreted as an antigen and $Premise_l$ as antibody receptor, it is possible to check how many

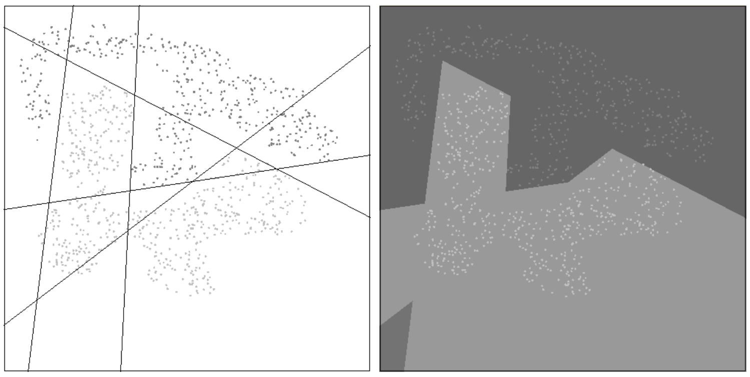

Figure 5. The results of applying classification rules with partial matching, constructed by means of evolved populations of hyperplanes (lines in 2D). Darker color depicts regions for which rules with only partial matching exist.

single conditions in $Premise_l$ and $tempPremise$ match. The final classification procedure is defined as follows:

1. For a given $\mathbf{x}_{test}$ construct $tempPremise$ by checking $\mathbf{x}_{test}$ against all $LC_j,\ j = 1, ..., K$.
2. Find a rule, whose premise exactly matches $tempPremise$. If there are at least two such rules, choose that with the highest value of $Perc$. Return $Concl$ of the chosen rule. If there is no exactly matching rule, go to step 3.
3. Find a rule that best matches $tempPremise$, i.e., the number of single matching conditions in the premises, is the biggest. If there are at least two such rules that match $tempPremise$ equally well, choose that with the highest value of $Perc$. Return $Concl$ of the chosen rule.

This approach allows also for signaling that a given test sample $\mathbf{x}_{test}$ is classified, but with a very low confidence level, i.e., it is classified by a rule with a very weak match $tempPremise$ generated by $\mathbf{x}_{test}$. A threshold can be set for the matching level, in order to signal the impossibility of classification rather than a classification with a very low confidence. Fig. 5 presents the class boundaries found by the aforementioned approach with partial matching. The areas with darker colors are those classified by partial matching. As it can be observed, although matching is only partial, the results are consistent with those given by exact matching.

3.5. The multiclass case

The aforementioned formulation of the Immune Hyperplane Algorithm deals with classification problems with only two classes. It can be, however, easily extended to the multiclass case. Let us consider a classification problem in which a training set T consists of samples belonging to C different classes, i.e.,

$T = \{(\mathbf{x}_i, y_i), i = 1, ..., M\}$, where $y_i \in G$. The set $G = \{1, 2, ..., C\}$ is a set of possible class labels. The idea of solving the classification problem with C possible classes by means of the proposed IHA is to create one classifier for each class, treated as class 1, and all samples from the rest of the classes as representing class -1. This will result in a set of C two–class IHC. During the classification of an unknown sample x_{test}, each j–th basic IHC, $j = 1, .., C$, tries to classify x_{test} as 1 or as -1, as it was described in the previous section. First, only the exact matches are considered. The answer 1 of the j–th classifier means that j–th IHA classifier classifies x_{test} into j–th class. The answer -1 means that the j–th IHA classifier classifies x_{test} as *not belonging to the j–th class*. If there are more basic classifiers that give answer 1, the final decision is made based on the $Perc$ values, i.e., the one with the highest $Perc$ wins. In the case of no exact matches, partial matches are considered. Note, however, that each basic IHC can have rules with premises of different lengths. This comes from the fact that each of the original classes can be separated from all others by a different number of hyperplanes and x_{test} has to be checked, in general, against different number of hyperplanes. For example, if one of the C classes can be linearly separated from all others, its rules will have premises of length 1 (assuming that IHA correctly finds a single separating hyperplane), while in the case of nonlinear class boundaries, more than one hyperplane is needed to separate classes, thus, the premise part of the rules has length greater than 1. Because of that, if there is no exact matching and partial matching has to be used, the matching level of different IHC cannot be compared directly, but rather a percentage of the exact match should be used. For example, if one of the IHCs has premise of length 4 and in one of its rules there are 2 conditions that match x_{test}, and in the case of other IHC with premise of length 9 there are 3 matches, it is better to choose the first case as there is 50% of the ideal match, and only 33% in the second case.

3.6. Generalization issues

In IHA, for the case where there is no correctly matching rule for a new sample, a rule with the highest, but incomplete matching, is selected. This can be viewed as a limited generalization ability, although it does not solve the problem. Even if the modified procedure of constructing the T_{pairs} set helps with dealing with noise in IHA, it is clear that the algorithm is constructed to maximize the overall number of correctly classified pairs. No evolutionary pressure is put upon the future generalization ability on test data.

Classification trees (Breiman et al., 1984) are **unstable** classifiers, i.e. slightly different training sets lead to different classifiers. This is similar for the proposed IHA. Differences in training sets result in different T_{pairs} to which IHA adapts. There exists yet another source of instability. The evolution is stochastic and after each run, different populations, and thus different sets of rules, are achieved. In general, although after each run one can separate the training data even ideally, the potential future classification error on test data is expected to have great variability, i.e., it is unstable. The instability of classifiers is not necessarily their drawback. It turned out that such classifiers are especially suited for constructing families of classifiers, through techniques such as bagging or boosting. The easiest way to limit the variability of a single classifier from IHA is to combine them in family **F** and use simple majority voting to make the final decision. If there are two classes with the same number of votes, a sum of all $Perc$ values of IHC classifiers voting for a given class are used, and the class with the highest value wins.

The techniques of bagging and boosting come from the attempts to answer the question whether weak classifiers (so called weak learners) can be used to form a better classifier through the aggregation of the weak ones. Weak classifier is a classifier that gives not much better results than random classification. However, when there are C **statistically independent** weak classifiers, the expected performance can be improved. Having enough classifiers one can count on correct classification by means of plurality voting. The problem is that having one training set T it is impossible to produce a set of independent classifiers. Yet, creating multiple training sets by randomly sampling the original training set and using them to train weak classifiers can bring significant improvements, although the classifiers are not independent. **Bagging** states for *bootstrap aggregating*, a method for generating multiple versions of classifiers and aggregating them into an aggregated predictor, was proposed by Breiman (1996). Bagging can improve accuracy if there is a big variability in the base classifier, i.e., if perturbing the training set can cause significant changes in the predictor constructed. For that reason, commonly used as base classifiers in bagging, are classification trees, due to their instability, which is their advantage in this case. Each replicate training data set is formed by sampling the original training set at random (with uniform probability), but with *replicates*. The bagging procedure is independent of the base classifier. This enables the use of the proposed IHC as the base classifier. This seems to be justified, as IHC are also unstable. The expected result would be decreased variability of IHC, even more than due to the procedure of constructing a simple family of IHC. This was empirically verified and the results are presented in Section 4. Boosting is also a general method for improving any type of classifier. The most striking similarity to bagging is that the base classifier is constructed several times by means of a different training set generated from the original training set. Unlike in bagging, in boosting the probability of each training sample to be selected to the consequent training sets is not constant. The probability of being selected is increased after each call to base classifier training, for samples that were hard to learn for the previous base classifiers. One of the first and most popular boosting algorithms is **AdaBoost** (Schapire, 2001), which was used int the experiments presented in this work. Random Forests (RF) were proposed by Breiman (2001) as aggregated classifiers, especially designed for the classification trees as the base (weak) classifiers. They are able to obtain very good generalization results. For that reason RF are used in the next section for comparison.

4. Results

In this section preliminary results obtained by the proposed immune algorithm are presented and compared with the results obtained by the Random Forest classifier. The proposed IHC was used as a base classifier in different strategies for constructing aggregated classifiers. These were: simple committee with majority voting, bagging, boosting (all with 50 base classifiers), bagging and boosting of committees of IHC (both with 50 committees, each consisting of 5 IHC). Several benchmark databases from UCI Repository were used for testing. The classifiers were tested by means of 5-fold crossvalidation. The tests were repeated 20 times and result were averaged (with an exception of Diabetes database, which was averaged over 10 runs). Standard deviations were also calculated. Table 1 summarises the results.

Table 1. Results of 5-fold cross-validation tests of the proposed aggregated classifiers. The classifiers are: single IHC (1), committee of IHCs (2), bagging with IHC (3),boosting with IHC (4), bagging with committees of IHCs (5), boosting with committees of IHCs (6), Random Forest (7). The results are presented in the form of *MeanTestError/Std.Dev.OfTestError*. (Ionos. and Diab. stands for Ionosphere and Diabetes, respectively.)

DB	1	2	3	4	5	6	7
Iris	6.1/1.5	4.8/0.8	**4.3/0.7**	4.4/0.8	**4.3/1.0**	**4.3/1.0**	4.9/0.9
Sonar	27.0/2.3	18.2/1.8	19.8/1.9	16.8/2.1	19.2/1.6	**15.9/2.0**	18.3/2.2
Ionos.	17.1/1.5	11.7/1.0	11.5/0.8	8.0/0.9	11.9/0.7	7.3/0.9	**6.9/0.4**
Glass	46.1/2.8	33.0/2.0	32.5/1.4	30.5/1.8	32.3/1.5	29.8/1.9	**22.4/1.8**
Diab.	28.1/1.7	24.2/0.9	**23.7/1.0**	26.1/0.8	23.9/0.5	25.8/1.3	23.8/0.7
Bupa	37.9/2.8	30.0/1.8	29.1/1.8	32.2/2.3	28.7/1.5	32.4/2.1	**28.2/1.7**
Vowel	62.8/4.3	36.5/1.7	38.2/1.6	33.2/1.9	38.8/1.1	**31.1/1.7**	46.3/1.7

The results are very promising. The proposed immune algorithm is not only able to perform comparably with the well established Random Forest (RF) algorithm, but it also outperforms RF on some of the databases. Even if the Random Forest algorithm turns out to perform better, the differences are not significant. Different aggregating strategies seem to be better suited for different databases. From the results obtained it is obvious that all aggregating strategies tested work with the proposed immune algorithms very well. This is observed through significantly better results of all aggregating classifiers when compared to a single IHC, and, secondly, through the decrease of variation of the single IHC due to the aggregation (which was expected). The important results prove that it is possible to construct supervised learning algorithms based on immune metaphors (clonal selection in this case). Being competitive even with Random Forests classifiers, they can become an important alternative in many real world classification problems.

5. Conclusions

In this paper a novel evolutionary algorithm is proposed. It utilises the concept of clonal selection from artificial immune systems in order to evolve the population of linear classifiers. The most significant novelty of this algorithm is that the evolution is based not only on competition but most of all on cooperation. The degree to which a given artificial lymphocyte contributes to the overall performance of the whole population is referred to as *usefulness*. It has been presented that *usefulness* in this context can be treated as a measure of a lymphocyte quality. By means of incorporating this novel concept into the clonal selection framework, the resulting algorithm is able to adjust the population size to the difficulty level of the classification problem. The proposed algorithm is very unstable (i.e., very sensitive to the change in the training set) and for that reason it is especially suitable for use in construction of aggregated classifiers. The results presented show that the proposed method is competitive with the well established classifier, such as Random Forests.

Among the increasing number of novel methods for evolutionary construction of combined classifiers it can be stated that the proposed Immune Hyperplane Algorithm can be assigned to the growing family of modern and robust classification techniques. Future work on the IHA algorithm should include the analysis of the statistical significance of the results obtained by IHA when compared to other classifiers. More sophisticated filtering process in construction of the T_{pairs} set should also contribute to the better results of IHA.

References

Ando, S. and Iba, H. (2003) Artificial Immune System for Classification of Gene Expression Data. In: E. Cantú-Paz et al., eds., *GECCO 2003*, **LNCS 2724**, Springer, 1926–1937.

Bandyopadhyay, S., Pal, S.K. and Aruna, B. (2004) Multiobjective GAs, quantitative indices, and pattern classification. *IEEE Trans. SMC B*, **34**, 2088–2099.

Bereta, M. and Burczyński, T. (2006) Immune K-means: A novel immune algorithm for data clustering and multiple-class discrimination. In: *Evolutionary Computation and Global Optimization 2006. Prace Naukowe, Elektronika.* Warsaw Univ. of Technology Publishing House Warszawa, 49–60.

Bereta, M. and Burczyński, T. (2007) Comparing binary and real-valued coding in hybrid immune algorithm for feature selection and classification of ECG signals. *Eng. Appl. Artif. Intell.* **20** (5), 571–585.

Bishop, C.M. (1995) *Neural Networks for Pattern Recognition.* Oxford University Press.

Breiman, L. (1996) Bagging predictors. *Mach. Learn.* **24** (2), 123–140.

Breiman, L. (2001) Random Forests. *Mach. Learn.* **45** (1), 5–32.

Breiman, L., Friedman, J.H., Olshen, R.A. and Stone, C.J. (1984) *Classification and Regression Trees.* Statistics/Probability Series, Wadsworth Publishing Company, Belmont, California, U.S.A.

Carter, J.H. (2000) The Immune System as a Model for Pattern Recognition and Classification. *Journal of the American Medical Informatics Assocation* **7** (1), 28–41.

Dasgupta, D. (1998) *Artficial Immune Systems and Their Applications.* Springer-Verlag New York, Inc., Secaucus, NJ, USA.

Gabrys, B. and Ruta, D. (2006) Genetic algorithms in classifier fusion. *Applied Soft Computing* **6**, 337–347.

Jing, X. and Zhang, D. (2003) Face recognition based on linear classifiers combination. *Neurocomputing* **50**, 485–488.

Kim, Y.S., Street, W.N. and Menczer, F. (2006) Optimal ensemble construction via meta-evolutionary ensembles. *Expert Systems with Applications* **30**, 705–714.

Kim, Y.W. and Oh, I.-S. (2008) Classifier ensemble selection using hybrid genetic algorithms. *Pattern Recogn. Lett.* **29** (6), 796–802, doi:http://dx.doi.org/10.1016/j.patrec.2007.12.013.

Kretowska, M. (2008) Ensemble of Dipolar Neural Networks in Application to Survival Data. In: *ICAISC '08: Proceedings of the 9th international conference on Artificial Intelligence and Soft Computing.* Springer-Verlag, Berlin, Heidelberg, 78–88.

Pal, S.K., Bandyopadhyay, S. and Murthy, C.A. (1998) Genetic algorithms for generation of class boundaries. *IEEE Trans. SMC B* **28**, 816–828.

Ruta, D. and Gabrys, B. (2005) Classifier selection for majority voting. *Information Fusion* **6**, 63–81.

Schapire, R. (2001) The boosting approach to machine learning: An overview. URL http://citeseer.ist.psu.edu/schapire02boosting.html.

Vapnik, V.N. (1995) *The Nature of Statistical Learning Theory*, Springer-Verlag New York, Inc., New York, NY, USA.

Wang, X. and Wang, H. (2006) Classification by evolutionary ensembles. *Pattern Recogn.*, **39**(4), 595–607, doi:http://dx.doi.org/10.1016/j.patcog.2005.09.016.

Watkins, A. (2005) Exploiting Immunological Metaphors in the Development of Serial, Parallel, and Distributed Learning Algorithms. Ph.D. thesis, University of Kent.

WATKINS, A., TIMMIS, J. and BOGGESS, L. (2004) Artificial Immune Recognition System (AIRS): An Immune-Inspired Supervised Machine Learning Algorithm. *Genetic Programming and Evolvable Machines* **5** (3), 291–317, URL citeseer.ist.psu.edu/watkins03artificial.html.

ZHANG, Y. and BHATTACHARYYA, S. (2004) Genetic programming in classifying large-scale data: an ensemble method. *Inf. Sci.* **163** (1-3), 85–101, doi:http://dx.doi.org/10.1016/j.ins.2003.03.028.