STUDIA INFORMATICA
Nr 1/2(7) Systemy i technologie informacyjne 2006

A Look Inside The Artificial Immune Algorithm Inspired by Clonal Selection Principle

Krzysztof Trojanowski[1], Michał Grzegorzewski[2]
[1] Institute of Computer Science, Polish Academy of Sciences,
ul. Ordona 21, 01-237 Warsaw, Poland
[2] Warsaw School of Information Technology
ul. Newelska 6, 01-447 Warsaw, Poland

Abstract: Artificial Immune Systems inspired by clonal selection principle (called clonal selection algorithms) have already been successfully applied to pattern recognition tasks. In this paper we present our implementation of one of them, called CLONCLAS, and discuss its behavior in application to recognition of a set of binary patterns. The algorithm performs process of learning based on a set of training data including patterns which belong to ten previously unknown classes and finally generates a group of classifiers which are able to assign the testing input patterns to appropriate classes. Our experiments were performed for a set of commonly known similarity measures of binary strings to select the most efficient of them. We also observed a phenomenon of transformation of memory contents in subsequent phases of iterated process of the system learning.

Keywords: CLONALG, Clonal Selection Principle, CLONCLAS, Pattern Recognition

1 Introduction

Immune system of mammals includes a set of defence mechanisms, and among them a primary and a secondary immune responses are the most sophisticated. The primary one is responsible for recognition and destruction of intruders, i.e. pathogens recognized as dangerous for the mammal which appeared for the first time in the organism. The recognition is based on a quick adaptation of the system to a new type of patterns. In case of detection of pathogens of the previously unknown type, the system has to build new cells (i.e. white blood cells called lymphocytes) which will be able to eliminate intruders as soon as possible. The secondary response is based on memory and ability to remember previously recognized pathogens and this way it reacts much faster than the primary one. Both mechanisms complement each other making an efficient tool for recognition and elimination of the different microbes, viruses, etc.

The process of adaptation of cells of the immune system to the new patterns is called clonal selection, and it is based on the reaction of the lymphocytes which are generated in the bone marrow (thus called B-cells) to the matching pathogens [1]. The B-cell matches the pathogens with its receptors called antibodies. On the surface

of the B-cell there are hundred thousands of such receptors. When an intruder appears in the organism, the B-cells are activated by the system. Those cells which match the pathogens bind to them and eliminate them from the organism. The B-cells are cloned and undergo hypermutation to create new receptors able to match the pathogens better and this way they explore the space of pattern shapes. The number of clones is directly proportional to the B-cell matching specificity while the strength of mutation is inversely proportional. The B-cell, which closely matches the pathogen (which is called antigen since then because it is responsible for generation of new B-cells with new antibodies, simply it is antibody generating) generates large number of clones and becomes memory B-cell eventually. Lifetime of B-cells is rather short in comparison to the lifetime of their owner however the memory B-cells live much longer.

In our research we used a heuristic approach based on clonal selection principle to the pattern recognition problem. The problem can be stated in the following form: "given a collection of objects belonging to a predefined set of classes and a set of measurements on these objects, identify the class of membership of each of these objects by a suitable analysis of the measurements" [4]. For our experiments we prepared a set of binary patterns divided into ten classes. The set for each of classes is a subset of randomly selected patterns from a larger set delivered by UCI Repository [5].

The clonal selection process was an inspiration for the CLONCLAS system [6]. CLONCLAS originates from CLONALG (CLONal selection ALGorithm) [2,3]. The CLONCLAS extends features of CLONALG by taking into account multiple examples of a class of patterns to be learned instead of a rule that there is only one pattern for each of the classes. The CLONCLAS was reimplemented and adapted by us for our experiments. Comparisons described in [6] were made for classifiers built with measures based on the Hamming distance. In our research we tested the algorithm with Hamming distance as well as with some other affinity measures for binary patterns. Additionally a new transformation of binary patterns was proposed which modified properties of the affinity measures and significantly improved the results.

This paper is organized as follows: in Section 2 our algorithm is described. Section 3 presents discussion on affinity measures applied in our research and Section 4 shows the solved problem, i.e. input data sets. Section 5 shows the algorithm in action, i.e. we present behaviour of the algorithm during the process of learning and its results. Section 6 concludes the paper.

2 Clonal Selection Algorithm

Our algorithm is based on the CLONCLAS however there are some modification in the schema presented in [6]. The goal is to generate a set of binary classifiers able to recognize more than one class of input patterns. Input data consist of a set of patterns, more precisely, binary strings of length n divided into classes. The number of classes is not given to the system. The system iteratively modifies and evaluates its set of classifiers using given affinity. The classifiers which are closer to the input patterns are evaluated as

better than the others. It is expected, that at the end of the process the set of classifiers will consists of different elements and each of them will be close to its input class. This expectation is a formal goal of the pattern recognition systems where a set of classifiers called modes is searched. It is obvious that the set of classifiers for the same set of input data can be different for different measures of affinity. It was interesting to study what set of classifiers they could generate and which affinity measure would be the most appropriate i.e. would give the highest accuracy.

The algorithm works with two sets of patterns: input patterns, called antigens (Ag) and system patterns (classifiers) called antibodies (Ab). In the model it is assumed that each B-cell is equipped with exactly one antibody. This assumption simplifies the model, without losing its adaptation features.

Ab is divided into two subsets:

$\mathbf{Ab} = \mathbf{Ab}_{\{r\}} \cup \mathbf{Ab}_{\{m\}}, \quad + \quad =$

where $Ab_{\{m\}}$ is a set of memory antibodies (i.e. memory B-cells) of size m, and $Ab_{\{r\}}$ is a set of remaining antibodies of size r. The main loop of the algorithm which is executed N_{gen} times consist of an internal loop executed for all M antigens. A pseudo-code of the algorithm is presented in Figure 2.1.

```
Input: Ab, Ag, N_gen, n, d, L, β
Output: Ab_m

for t = 1 to N_gen begin
 for j = 1 to M begin
   f̄_j      := affinity( Ab, ag_j );
   Ab^j_{n} := select( Ab, f̄_j , n);
   C^j      := clone(Ab^j_{n}, β, f̄_j );
   C^j*     := hypermut(C^j, f̄_j );
   f̄_j*     := affinity(C^j*, ag_j );
   ab_j*    := select(C^j*, f̄_j*, 1 );
   Ab^j_{m} := insert(Ab^j_{m}, ab_j* );
   Ab_{r}   := replace(Ab_{r}, C^j* , f );
   Ab_{d}   := generate( d, L );
   Ab_{r}   := replace(Ab_{r}, Ab_{d} , f );
 end;
end
```

Figure 2.1. A pseudo-code of the version of CLONCLAS algorithm applied in experiments.

The internal loop consists of the following steps: *affinity* method evaluates antibodies in *Ab***,** i.e. compares them with *j*-th antigen (ag_j) and writes the evaluated affinities in vector $\overline{f_j}$, *select* method selects the set of *n* the most fitted patterns to the *j*-th antigen ($Ab^j_{\{n\}}$), *clone* method generates set of clones (C^j) of size n_c of the selected patters, where:

$$n_c = \sum_{i=1}^{N} round\left(\frac{\beta \cdot N}{i}\right),$$

hypermut method mutates clones from C^j and this way creates a set of mutated clones C^j*, *affinity* method evaluates mutated clones from C^j* i.e. compares them with ag_j and writes the evaluated affinities in vector $\overline{f_j^*}$, *select* method selects the best mutated clone ab_j^* , *insert* method adds the best mutated clone ab_j^* to the memory ($Ab^j_{\{m\}}$) if $f(ab_j^*)>f(ab_m^j)$, *replace* method replaces patterns from the reference set $Ab_{\{r\}}$ by their clones from C^j* if the clone is better than its ancestor, *generate* method randomly generates set of *d* new solutions ($Ab_{\{d\}}$) and *replace* method replaces the least fitted patterns from the reference set $Ab_{\{r\}}$ by the new ones from $Ab_{\{d\}}$. Then the sequence of steps is finished and the flow-control returns to the beginning of the loop. The detailed description of the algorithm can be found in [2].

The affinity of the set of antibodies to the *K*-th class of antigens of size *k* is evaluated with the formula:

$$TotalAffinity(Ab, Ag^K) = \sum_{j=1}^{k}\sum_{i=1}^{N} ab_i \oplus ag_j^K$$

where:

ab_i – *i*-th antibody form *Ab*,

ag_j^K – *j*-th antigen from class *K*,

and $\oplus$ represents affinity measure (e.g. Hamming distance, or any other).

3 Affinity Measures

In our study a set of affinity measures was taken into account [Nadler93]:

1. Russel and Rao,
2. Jaccard and Needham,
3. Kulzinski,
4. Sokal and Michener,
5. Rogers and Tanimoto,
6. Yule,

and compared with a seventh measure which was a Hamming distance.

A measure is a mapping ρ which for every two elements of the domain returns a non-negative real valued number (R_*):

$\rho : \mathrm{X} \times \mathrm{X} \to R_*$

The relation is called a measure if it is defined with the following rules:

$\underline{x}^\mu \in X \ (\mu = 1, 2, \ldots)$

$\rho\left(\underline{x}^\mu, \underline{x}^\nu\right) = 0 \Leftrightarrow \underline{x}^\mu \equiv \underline{x}^\nu,$

$\rho\left(\underline{x}^\mu, \underline{x}^\nu\right) = \rho\left(\underline{x}^\nu, \underline{x}^\mu\right),$ (symmetry)

$\rho\left(\underline{x}^\mu, \underline{x}^\nu\right) < \rho\left(\underline{x}^\nu, \underline{x}^\eta\right) + \rho\left(\underline{x}^\eta, \underline{x}^\nu\right)$

Some of the selected affinity measures do not satisfy the formal conditions of the measure mentioned above. In spite of this in further text we shall still call them measures, even if probably the more appropriate name would be *affinity function*.

All the measures applied in our tests are described below. For the formal description we shall use the following definition of the binary strings:

$X, Y \in \{0,1\}^N$,

and the following reference variables:

$$a = \sum_{i=1}^{n} \varsigma_i, \quad \varsigma_i = \begin{cases} 1, & X_i = Y_i = 1 \\ 0, & otherwise. \end{cases} \qquad b = \sum_{i=1}^{n} \xi_i, \quad \xi_i = \begin{cases} 1, & X_i = 1, Y_i = 0 \\ 0, & otherwise. \end{cases}$$

$$c = \sum_{i=1}^{n} \gamma_i, \quad \gamma_i = \begin{cases} 1, & X_i = 0, Y_i = 1 \\ 0, & otherwise. \end{cases} \qquad d = \sum_{i=1}^{n} \psi_i, \quad \psi_i = \begin{cases} 1, & X_i = Y_i = 0 \\ 0, & otherwise. \end{cases}$$

Note, that the total: $a+b+c+d$ is a constant value and equals n, i.e. the length of the binary string.

a) Russel and Rao:

$$f = \frac{a}{a+b+c+d} = \frac{a}{n}$$

b) Jaccard and Needham:

$$f = \frac{a}{a+b+c}$$

c) Kulzinski:

$$f = \frac{a}{b+c+1}$$

d) Sokal and Michener:

$$f = \frac{a+d}{a+b+c+d} = \frac{a+d}{n}$$

e) Rogers and Tanimoto:

$$f = \frac{a+d}{a+d+2(b+c)}$$

f) Yule:

$$f = \frac{ad-bc}{ad+bc}$$

The last of the discussed measures was a Hamming distance d_H:

$$d_H = \sum_{i=1}^{N} \overline{(X_i \oplus Y_i)},$$

which could be denoted in terms of a, b, c, d as

$d_H = b + c$

Properties of the selected measures were subject of our special interest. We especially considered the problem of equal opportunities of the binary strings matching with these measures. For example, in case of Hamming distance it is clear that $d_H \in \langle 0,..,n \rangle$ for every two strings of length n. However it is not obvious for the other measures. Therefore, we checked if the number of different affinity values is constant for all binary strings of the same length for the six affinity measures. We did a set of evaluation with 16-bit strings. Simply we evaluated how many different affinity values can be obtained for each of the measures for each of the strings matched with all of the 16-bit strings. Thus for each of the 65535 string a number of 65535 matches were performed. The results are presented in the histograms (Figure 3.1).

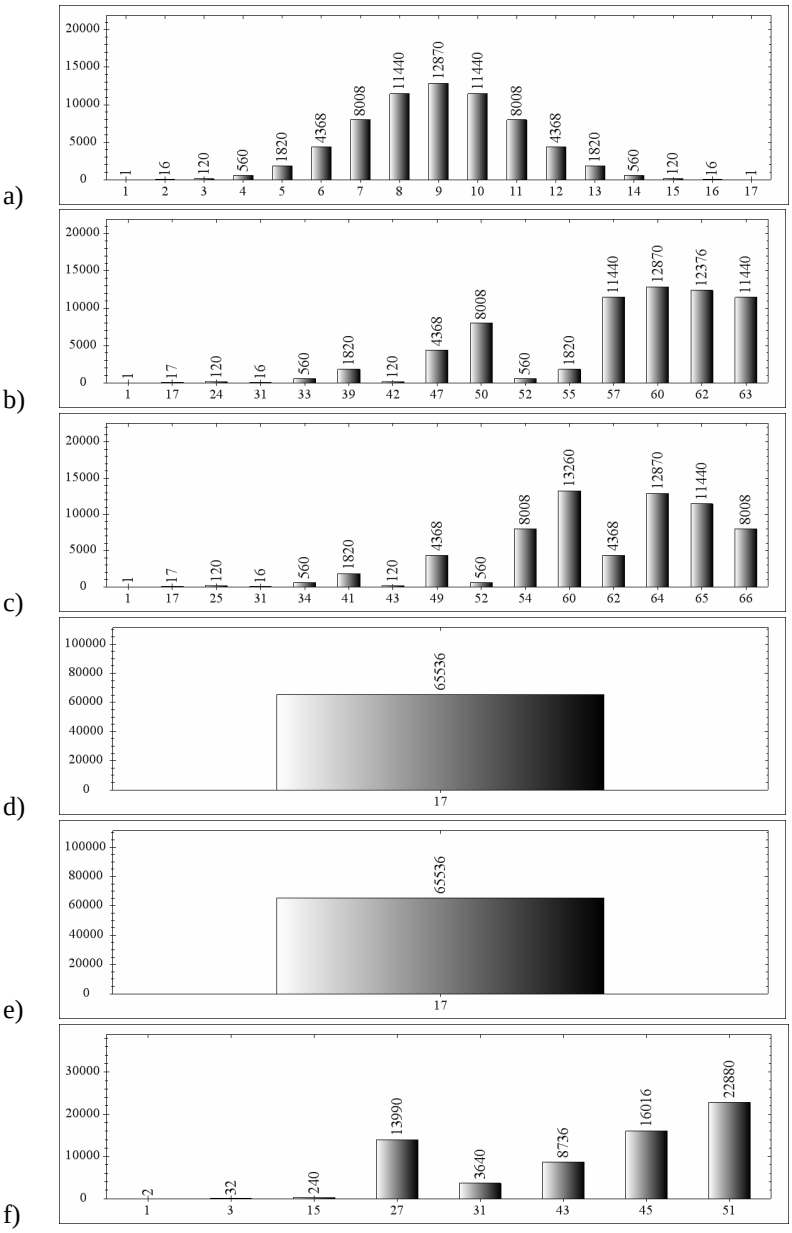

Figure 3.1. Histograms of numbers of different distances for affinity measures: a) Russel and Rao, b) Jaccard and Needham, c) Kulzinsky, d) Sokal and Michener, e) Rogers and Tanimoto, f) Yule

In Figure 3.1 each bar in the histograms presents a number of strings with respective number of different matching values (the number of different matching values is on the X axis). For example, in the Figure 3.1(a) it can be seen that there is one string, where the number of different distances to all the remaining 65535 strings equals 1, there are 16 strings where the number of different distances equals 2, 120 strings where the number of different distances equals 3, etc. The histogram shows that the measure does not treat the strings equally, but its properties change respectively to the measured points.

Because of inequality in treating the strings by the four of six selected measures, a new transformation T operator was proposed [7]. All the measures except for Hamming were applied to the transformed pairs of binary strings. Before the evaluation one of the patterns in the pair was modified by a transformation operator T working as follow.

For every two patterns A, B $\in \{0, 1\}^n$:

$$\underset{i\in\{0,1,..,n\}}{\forall} A[i]=0 \Rightarrow ((A[i]=1)\wedge(B[i]=1-B[i]))$$

The operator reduces the search space, i.e. for a set of 65536 pairs of 8-bit binary strings we obtained 256 different transformed pairs. After transformation one of the strings is always turned into a sequence of digits "1", while the other includes information about differences between the input strings. The operator is simple and of low computational cost, however significantly modifies properties of the measure and improves their sensitivity.

The operator is applied just before matching. Every matched pair of strings is first turned into a new pair with the T operator and then the measure is applied to the new pair of strings. However, the returned value is assigned to the original pair of strings, i.e. the pair before transformation. For all the measures applied with T transformation (except for Hamming) and for all the 65536 tested strings the number of different distances to all the other strings was the same and equal 17. Thus, it can be said that for the measure with T transformation all the strings are treated equally.

When we use T transformation one of the resulting strings is always a sequence of digits "1". It means that X_i never equals 0, so the values c and d are equal zero for all pairs of transformed binary strings. The definitions of the measures changed:

a) Russel and RaoT:

$$f=\frac{a}{a+b}$$

b) Jaccard and NeedhamT:

$$f=\frac{a}{a+b}$$

c) KulzinskiT:

$$f=\frac{a}{b+1}$$

d) Sokal and MichenerT:

$$f=\frac{a}{a+b}$$

e) Rogers and TanimotoT:

$$f=\frac{a}{a+2b}$$

f) YuleT: does not exist.

This way we finally obtained three affinity measures for tests:

T1) $f = \frac{a}{a+b}$ T2) $f = \frac{a}{b+1}$ T3) $f = \frac{a}{a+2b}$ T4) Hamming distance.

4 Testing Data Sets

For our experiments we selected a set of patterns from the UCI repository [5]. It was a set of letters represented by 10x12 binary matrices. The set consisted of 1000 letters divided into 10 classes, each represented by 100 patterns. The set was divided into two disjoint subsets: a training set and a testing set where each of them consisted of 50 patterns randomly selected from each of 10 classes. A set of patterns from the class 'A' of the training set is presented in Figure 4.1.

The expected result of the algorithm is a set of ten classifiers where each of the classifiers represents one of the classes i.e. "recognizes positively" most of the patterns from this class and "negatively" all the others. For a randomly generated set of patterns the probability of proper recognition should be close to 10%.

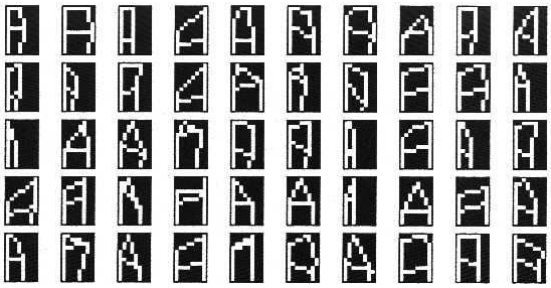

Figure 4.1. A set of patterns from the class A (the letter A) of the training set.

5 Experiments

A subject of our interest was the process of building the memory content during the optimization process for different affinity measures. The algorithm was equipped with 10-cells memory buffer, each cell for each of the classes of input data. At the very beginning of each of experiments the memory buffer was filled in with randomly generated patterns. During subsequent iteration the content changed, randomly generated patterns were exchanged by others generated with the algorithm. We observed the content of the memory during a set of experiments with the selected measures of affinity, each of the experiments with the same input data.

After few preliminary tests it was observed that all the applied measures except for the Hamming measure converge to a stable set of system patterns within at most 1000 iterations. For the Hamming the system needed more iterations – 3000 and more.

A sample set of system patterns stored in the memory buffer for ten classes of input patterns at the iterations: 1, 10, and 1000 obtained during experiments with T2 measure is presented in Figure 5.1.

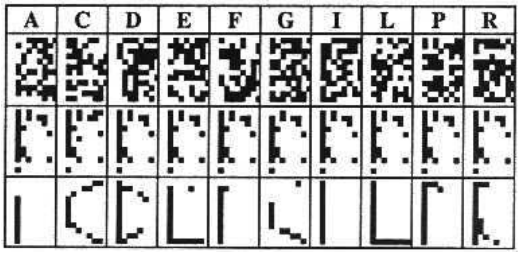

Figure 5.1. A set of classifiers for ten classes of input patterns at the iterations: 1 (first row), 10 (second row), and 1000 (last row). The classifiers were obtained with T2 measure

Note that at the 10-th iteration the set of patterns consisted of nine identical patterns and one very close to them. It means that at the beginning of the process the algorithm tries to find just one solution which will be common for all classes. However, at the end of the optimization process the set consists of ten different patterns like the ones in the last row in Figure 5.1, where each of them is created for recognizing patterns for the different class. The results obtained with the remaining affinity measures (except for the Hamming) looked similar, i.e. the number of different patterns in the set was small at the beginning, and increased with the progress of the process. This behaviour is presented in Figure 5.2.

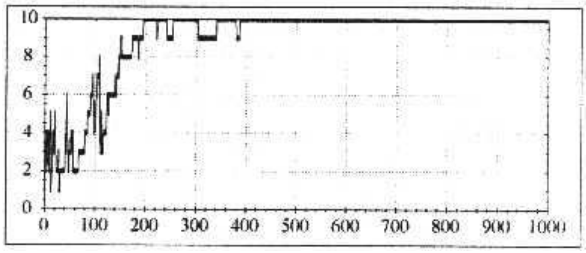

Figure 5.2. Number of different patterns in the memory buffer with T2 measure applied. X axis represents number of iteration

In case of Hamming distance the algorithm's behaviour is different that for the other measures. The algorithm needs more iterations to converge, and the number of different patterns varies in time. The set of system patterns for ten classes of input patterns at the iterations: 1, 10, and 1000 obtained during experiments with Hamming distance is presented in Figure 5.3.

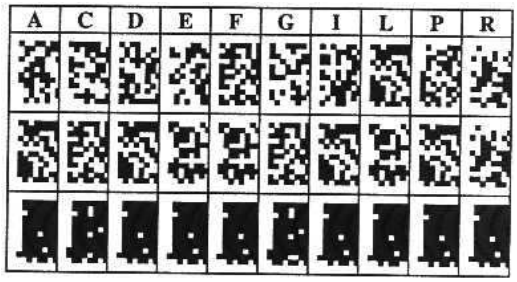

Figure 5.3. A set of system patterns for ten classes of input patterns at the iterations: 1 (first row), 10 (second row), and 1000 (last row). The classifiers were obtained with the Hamming distance.

The initial set of randomly generated 10 different patterns decreased to the number of 4 different patterns after first ten iterations. Then the number went down further and gained the level of one or two patterns. The number of different patterns in the memory observed during a single run of the process is presented in Figure 5.4.

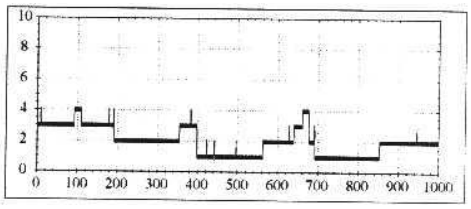

Figure 5.4. Number of different patterns in the memory buffer with Hamming distance applied. X axis represents number of iteration

It is definitely not a state of stagnation because the number of different patterns slightly varies in subsequent iterations which means that the algorithm is still searching. On the other hand, in spite of the searching effort, the algorithm is not able to find the appropriate set of patterns.

The next question was whether it is a typical behaviour of the algorithm with Hamming distance or it is just a matter of the algorithm's parameters settings. Further research showed that the algorithm was able to converge however with another set of parameters. In both cases described above, the algorithm converged to a set of ten different patterns identifying classes of input data with the following values: mutation probability =0.25, β=2 and d=21. Figures presented above were taken with different parameters: mutation probability =0.05, β=0.5 and d=2.

Another observation concerned with comparison of measures: T2 and Hamming shows that the final set of patterns obtained at the end of experiments is quite similar to each other except for the fact that the patterns from one set are the negative version of the others (Figure 5.5).

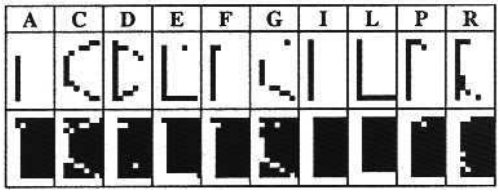

Figure 5.5. A final set of patterns obtained at the end of experiments for two affinity measures: the T2 measure (first row) and the Hamming distance (second row)

We also observed the accuracy of the classifiers learned with different affinity measures. Detailed description of accuracy obtained with Hamming distance is presented in Table 5.1.

The numbers on the diagonal of the Table 5.1 represent correct recognitions of the classifiers, while all the remaining numbers – incorrect ones. Looking at the found patterns it can be seen that some of them are not very similar to the letters they have to identify. It should be stressed that the patterns have been generated respectively to the letters present in the input data. Some of them, e.g. patterns for "C" or "D", are easy to identify. Others, like patterns for "A" or "G", do not resemble their letters at all. For the input data where patterns from different classes differ from each other, the system patterns are also different and easy to identify. In our case, the bit map of size 10 by 12 is a very simple pattern and therefore the presence of similarities between the letters is obvious and natural.

Table 5.1. Hits of the classifiers generated with Hamming distance: rows represent different classes of input data and columns: different classifiers.

	A	**C**	**D**	**E**	**F**	**G**	**I**	**L**	**P**	**R**
A	**21**	0	2	0	6	0	9	0	11	1
C	2	**25**	0	1	0	22	0	0	0	0
D	0	0	**38**	0	3	0	0	0	9	0
E	2	3	2	**20**	2	0	2	14	5	0
F	2	0	0	0	**26**	0	0	0	22	0
G	1	22	0	1	0	**26**	0	0	0	0
I	10	0	0	0	0	0	**40**	0	0	0
L	2	0	0	2	0	0	8	**38**	0	0
P	0	0	2	0	0	0	3	0	**45**	0
R	0	0	2	0	1	0	2	1	23	**21**

Graphical presentation of the accuracy of the classifiers learned with all the tested measures are presented in Figure 5.6.

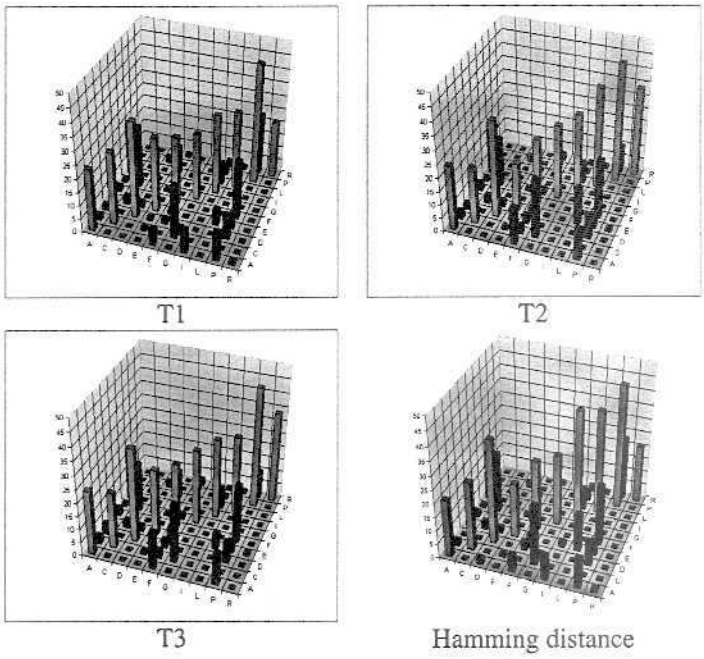

Figure 5.6. Accuracy of the classifiers learned with the tested measures

Below there is a comparison of hit ratios for the final set of system patterns stored in memory buffer for the selected measures evaluated with the testing set of input data (Table 5.2).

Table 5.2. Hit ratios of the system patterns generated with the tested affinity measures

Measure	Avg. value	Best value	Worst value
T1	50.44	56.8	36.4
T2	53.14	57.4	39.2
T3	46.76	55.2	30.2
Hamming	39.82	56.6	11.0

The last comparison shows distances of the classifiers to its classes as well as the distances of the best pattern in the class to the rest of patterns. In this case, the best pattern is the pattern closest to the classifier. The distance between the pattern (or classifier) and its class is evaluated as the total of distances from the pattern to all the patterns of this class. In Table 5.3 distances for Hamming distance are presented. The tables for the other measures are similar.

Table 5.3. Distances of the classifiers and selected patterns to their classes for Hamming distance

Class	**A**	**C**	**D**	**E**	**F**	**G**	**I**	**L**	**P**	**R**
classif.	1000	662	649	810	570	876	801	122	352	637
b. patt.	1280	718	713	910	668	1006	1003	122	374	767

For every classifier the distance to its class is closer or at least the same as for the best pattern of the class. For the letter "L" the distances are equal which means that the classifier is identical to the best pattern. However, for most cases the distance of the classifier is about 20% better than the distance from the best pattern which is a quite significant difference. It can also be seen that the distances are not the same i.e. the highest distance is for the letter "A" while the lowest is for the letter "L". This difference shows that the classes of letters are not equal to each other, simply some of them can be easier to recognize than the others.

6 Conclusions

The best results were obtained for the T2 affinity measure, while the worst ones – for the Hamming distance. This confirms our previous observations about difficulties with convergence of the algorithm when the Hamming distance is applied as the affinity measure. The hit ratio for the Hamming distance is significantly worse than the ratios of the other measures (with T transformation included). The results are not too impressive: a level of more than 50% of correct recognitions is of course a better result than the statistical level of 10% in case of randomly generated patterns. The reason for this could be found in the properties of the input data, especially in the similarity of some of the classes of patterns to each other and therefore difficult to discriminate between classes (e.g. 'F' and 'P', or 'G' and 'C'). On the other side, the results showed that the artificial immune system is able to cope even with difficult input data, and that the decision of selection of appropriate affinity measure is very significant for the quality of the results.

References

1. Burnet F.M., (1959). *The Clonal Selection Theory of Acquired Immunity*. Cambridge University Press.
2. de Castro L.N., Von Zuben F.J., (2001). Learning and optimization using the clonal selection principle. IEEE Transactions on Evolutionary Computation, Special Issue on Artificial Immune Systems, 6(3): 239-251.
3. de Castro L.N., Von Zuben F.J., (2000). The clonal selection algorithm with engineering applications. *GECCO'00, Workshop on Artificial Immune Systems and their Applications*, 36-37.
4. Nadler M., Smith E.P., (1993). *Pattern Recognition Engineering*, John Wiley and Sons, New York.

5. Newman D.J., and Hettich S., and Blake C.L., and Merz C.J., (1998). UCI Repository of machine learning database [http://www.ics.uci.edu/~mlearn/MLRepository.html]. Irvine, CA: University of California, Department of Information and Computer Science.
6. White J.A., Garret S.M., (2003). Improved pattern recognition with artificial clonal selection? *ICARIS-2003*, Springer-Verlag Lecture Notes in Computer Science No.2787, 181-193.
7. Trojanowski K., Jankowiak M., (2004). Właściwości miar podobieństwa w modelu sieci idiotypowej z binarna reprezentacją wzorców, *IPI PAN Reports*, Nr 997, Warsaw, Poland (in Polish).