Genetic Classifier System as a heuristic weighting method for a Case-Based Classifier System

Elisabet Golobardes i Ribé Xavier Llorà i Fàbrega
Josep Maria Garrell i Guiu David Vernet i Bellet
Jaume Bacardit i Peñarroya

Departament d'Informàtica
Enginyeria i Arquitectura La Salle
Universitat Ramon Llull (URL)
Passeig Bonanova 8 - 08022 Barcelona
{elisabet,xevil,josepmg,dave,is04517}@salleURL.edu

Abstract

This paper proposes how to use a Genetic Classifier System as a heuristic weighting method for a Case-Based Classifier System. The preliminary results are performed into a specific domain: the prediction of the student's qualification for a WWW Distance Learning Platform.

Keywords: Case-Based Reasoning, Genetic Algorithms, Computer Assisted Learning, Machine Learning.

1 Introduction

The main goal of this paper is to propose the possibility of using a Genetic Classifier System as a weighting method for a Case-Based Classifier System. This idea comes up when we analyse the performance of prediction systems[1] using Case-Based Reasoning (CBR) or Genetic Algorithms (GA), corresponding to the framework of the project developed under grant CICYT/Tel98-0408-02. The work presented looks for interpreting the prediction rules obtained by the Genetic Classifier System as a weighting method for the features used in the Case Based Classifier System. We are also interested in the fact that this "hybrid method" outperforms both approaches, when they work independently. The results are obtained using a data set from a real-world problem. Our further work consists of outlining the hybrid method and evaluating this proposal in multiple domains.

The paper is structured as described. First, a detailed description of the prediction environment -the framework of this paper- is given. Second, we present the Case-Based Classifier System used and the preliminary results obtained. Then, analogously, we describe the Genetic Classifier System used and its preliminary results. In the next section, we propose a "hybrid system" among both methods: using the Genetic Classifier System as a weighting method for the Case-Based Classifier System. Finally, we analyse its results, and we present the conclusions and further work.

2 Framework

2.1 Project environment

The project entitled *Implementation and Study of a new generation network to support Open Distance Learning* (under grant CICYT/Tel98-0408-02) is the framework for the work presented. The main goal of this project is to develop an open-distance learning tool based on a WWW platform (like [5]). Different subjects (or part of them) are offered on the WWW environment using different multimedia contents: text, images, video, audio, which provide alternative ways of learning. It also allows a useful feedback between the learning process of the student and the Intelligent Tutor-

[1] In our work, a prediction system can also be seen as a classifier system.

ing System [19]. Using this feedback the platform can dynamically configure its contents in order to help the student's learning. This adaptive configuration can be done in different ways. The first one involves the way of teaching certain contents. That is, offering text lessons, or text lessons combined with images, or with audio and video, etc. depending on the detected preferences of the student. Another possibility is adapting the contents themselves, not only the way these contents are shown. Using the feedback from the student, the platform detects the areas that the student has to improve. Therefore the platform can offer extra support: additional documentation, detailed descriptions, exercises, practices, etc.

In order to perform any of these adaptations, we need:

- Student's features: a set of data about the student and the learning that he/she is performing. This information can be collected from tests, exams, practices, exercises, etc. that the student solves along the navigation through the Web pages of the subject.
- Prediction system: it takes the student's features and performs a prediction about its learning response.
- Adaptive contents system: it adapts, using the previous prediction, the contents and their presentation to the student in order to help the student learning.

This paper is focused on the prediction system. This system is the key point of the adaptive behaviour of the platform. This prediction is achieved using three different proposals: a Case-Based Classifier System, a Genetic Classifier System, and finally a hybrid system among both approaches. The hybrid system is the main goal of this paper.

The preliminary experiments we report are based on data collected in a "presential" subject called *Programming*. This subject is an introductory course to the basic concepts for design and programming. This course is planned for the first year students in Engineering [4].

2.2 Educational Environment

The educational environment is based on the subject *Programming*, and the data comes from the academic course 1998/1999. This subject is an annual course. The final qualification depends on different exams, practices, exercises, etc., that the student solves along his/her navigation. The preliminary results presented uses the information described in table 1.

Table 1: Initial features used for the prediction of the final qualification for the subject *Programming*. The value -1 represents Non Presentation.

Feature	Description
Theory Group (TG)	There are seven different groups: {A,B, C,D,E,F,G,X}. X value indicates a student with no group assigned.
First year? (FY)	Does the student repeat the course? Boolean value
First Exam (FE)	Qualification of the first exam. [-1,10]
Second Exam (SE)	Qualification of the second exam. [-1,10]
Practice 1 (P1)	Qualification of the first practice. [-1,10]
Practice 2 (P2)	Qualification of the second practice. [-1,10]

Table 2: Possible qualifications (*classes*) of the prediction.

Class	Description
E	Non Presentation to the final exam
D	A student with a qualification between 0 and 4 [0-4)
C	A student with a qualification between 4 and 7 [4-7)
B	A student with a qualification between 7 and 9 [7-9)
A	A student with a qualification upper to 9 [9-10]

There are five possible qualifications or classes (see table 2). Thus, the system predicts the class that fits the real final qualification. In fact, two classes can be used: *fail* or *pass*. Using five classes gives us more information in two senses:

1. Class C will be the most controversial one. Thus, controversial students can be easily detected using five classes than just two.
2. It also helps the system to model the user. Because using five classes more learning behaviours can be expressed.

3 Prediction using CBR

Case-Based Reasoning (CBR) integrates in one system two different characteristics: machine learning capabilities and problem solving capabilities. CBR uses a similar philosophy to that which humans sometimes use: it tries to solve new cases (examples) of a problem by using old previously solved cases [16]. The process of solving new cases contributes with new information and new knowledge to the system. This new information can be used for solving other future cases. The basic method can be easily described in terms of its four phases [1]. The first phase *retrieves* old solved cases similar to the new one. In the second phase, the system tries to *reuse* the solutions of the previously retrieved cases for solving the new case. The third phase *revises* the proposed solution. Finally, the fourth phase *retains* the useful information obtained when solving the new case.

Our problem consists of making a prediction about the final student's qualification. In this sense, we could consider this problem like a classification problem: once a student is given, we predict the class where the student belongs to. This task can be done using a Case-Based Classifier System.

In a Case-Based Classifier System, it is possible to simplify the reuse phase classifying the new case using the same class as the most similar retrieved case.

3.1 Description of the Case-Based Classifier System

We use CaB-CS (Case-Based Classifier System) [7, 8] and some extensions [11, 22]. The kernel of CaB-CS can be easily configured and modified in order to test different variants for all the phases of the CBR cycle. CaB-CS is developed using the ANSI C++ programming language on UNIX systems.

Following we present the different CaB-CS configurations used for the results presented in this paper.

3.1.1 Case Memory

The organisation of the cases and the size of the Case Memory (CM) can be two key factors for the performance of a Case-Based Classifier System. For instance, both help the similarity functions in order to retrieve cases from the CM quickly. But the aim of this paper is obtaining preliminary results, showing that the Case-Based Reasoning is a good technique for this kind of prediction. Thus, we want to achieve a good percentage of correctly classified examples. In this sense, the system offers a simple *organisation* of the CM and two criteria for its *initialisation*.

Organisation. The organisation of the Case Memory consists of a *list* of cases. In this work, we use a simple organisation because our goal is to evaluate the capability of a Case-Based Classifier System in the prediction task. Our further work looks for improving the management of the CM.

Initialisation. The system uses two different criteria for the initialisation of the CM: *Initial load* and *Initial training*.

- *Initial load*. This criterion consists of loading the initial set of cases into the case memory. This option is useful for evaluating the initial corpus of cases.
- *Initial training*. This criterion consists of simulating the CBR system when it is used to solve new cases. The goal is to add only the "relevant" cases into the CM. In this sense, we expect increasing the accuracy and obtaining a reduced and representative CM.

3.1.2 Retrieval phase

Phase 1 retrieves the *most similar* case or cases to the new case. Obviously, the meaning of *most similar* will be a key concept in the whole system. Similarity between two cases is computed using different similarity measures. The practical implementation (used in CaB-CS) of these measures are: the *NNA*, the *Minkowski's metric*, and the *Clark's distance*.

Nearest Neighbour Algorithm. The Nearest Neighbour Algorithm (NNA) [23] is defined as:

$$NNA(Case\_x, Case\_y) = \frac{\sum_{i=1}^{F} w_i \times sim(x_i, y_i)}{\sum_{i=1}^{F} w_i} \tag{1}$$

Where $Case\_x$ and $Case\_y$ are two cases, whose distance is computed; F is the number of features that describes one case; x_i, y_i represent the value of the *ith* feature of cases $Case\_x$ and $Case\_y$ respectively; w_i is the weight of the *ith* feature; and $sim(x_i, y_i)$ is the similarity computed between x_i and y_i such as $\mid x_i - y_i \mid$.

Minkowski's metric. The Minkowskian function is defined as:

$$Similarity(Case\_x, Case\_y) = \sqrt[r]{\sum_{i=1}^{F} w_i \times |x_i - y_i|^r} \quad (2)$$

Where *Case_x* and *Case_y* are two cases, whose similarity is computed; *F* is the number of features that describes the case; x_i, y_i represent the value of the *ith* feature of cases *Case_x* and *Case_y* respectively; and w_i is the weight of the *ith* feature.

In this study we test the Minkowski's metric for three different values of r: *Hamming distance* for $r = 1$, *Euclidean distance* for $r = 2$, and *Cubic distance* for $r = 3$.

Clark's distance. The Clark's distance is defined as:

$$Similarity(Case\_x, Case\_y) = \sqrt[2]{\sum_{i=1}^{F} \frac{|\,(x_i - y_i)\,|^2}{|\,(x_i + y_i)\,|^2}} \quad (3)$$

Where *Case_x* and *Case_y* are two cases, whose similarity is computed; *F* is the number of features that describes the case; and x_i, y_i represent the value of the *ith* feature of cases *Case_x* and *Case_y* respectively.

3.1.3 Weights

Both NNA and Minkowski's metric need to weigh the feature relevance. In this paper the system works using two basic options: *Without weights* (or *Weights0*) and *Weights1*.

Without weights (Weights0). The worst possibility (but often found) is that no information is known about the features relevance, and there are not automatic methods that could compute the weights. In this case, we consider that for each weight w_i, its value is 1.0.

Weights1. The *weights1* propose a set of weights for the available data, using the teacher's criteria. In this sense, let $w_1 = 0.0$, $w_2 = 1.0$, $w_3 = 2.25$, $w_4 = 2.25$, $w_5 = 0.75$ and $w_6 = 0.75$ be the proposed weights.

3.1.4 Retain phase

In order to decide whether a case is representative enough or not to be stored into the case memory, two different basic criteria (function modes) are tested: *Test mode* and *DifClas mode.*

Test mode. In this mode the system does not store any new case in the case memory. This criterion has been used for two reasons. On one hand, the results obtained using this mode can be compared, in equal conditions, to those obtained using other machine learning methods that do not include learning while solving new problems. On the other hand, it allows us to evaluate the initial *corpus* of the case memory.

DifClas mode. This mode represents a halfway solution between the *Test mode* and the possibilities of retaining all new cases. The system will store the new case if it can not classify it correctly. Otherwise it will not be stored.

3.2 Results

These results are obtained using an initial set of 648 students of the *Programming* subject, in the academic year 1998/1999. For each student we have a data set with the corresponding data about the features presented in table 1, and him/her final qualification obtained corresponding at the final exam (a real value that belongs to the range $[0 \cdots 10]$ or -1 represents Non Presentation). From these data, the testbed consists of:

- The initial set is divided on two sets: *training* and *test* set. This division may be done using different proportions. For instance, 90% of the students for the *training* set and the rest (10%) for the *test* set (called 90%-10% proportion). We consider the *proportions*: 10%-90%, 20%-80%, 30%-70%, 40%-60%, 50%-50%, 60%-40%, 70%-30%, 80%-20%, 90%-10%.
- For each proportion, the division between both sets (training and test) may be done in different ways. Each possible division will be called *version*. For each proportion we performed 50 different versions.
- The different *configurations* used by CaB-CS is shown in table 3, which have been described at section 3.1.

Table 3: CaB-CS configurations used for these preliminar results.

	Options
Initialisation	Initial load and Initial training
Retrieval phase	NNA, Minkowski's metric (Hamming, Euclidean, and Cubic) and Clark's distance
Weights	Weights0 and Weights1
Retain phase	Test and DifClas

Summarising, one *run* depends on one *proportion*, one *version* and one *configuration*. Thus, the results are obtained from all possible runs. Following, we analyse the results from two points of view. First, we perform the final prediction using five classes. Second, we consider only two classes: fail or pass.

3.2.1 Results using five classes

In order to evaluate the CBR capabilities for solving this problem; we analyse which configurations obtain the best results (the maximum value, the best mean, and the upper minimum value). Table 4 shows these results, we observe that the minimum value corresponds to a reduced initial case memory, so we have to work in this sense, in order to obtain a compact and representative CM. Table 5 shows the performance of the best configuration, in other words, for each *real* class prints the percentage that correspond to the *predicted* class.

Table 4: Results obtained by CaB-CS using five classes.

	%PA	Configuration
Mean	64.21	Initial Load, Eucl., Weights1, Test, 9010
Max.	**81.25**	Initial Load, Eucl., Weights1, Test, 9010
Min.	57.10	Initial Load, Ham., Weights1, Test, 5050

Table 5: Error analyse of the results obtained using five classes, for the best configuration of the table 4.

Prediction Real vs	E	D	C	B	A
E	**77.43**	18.26	4.28	0.02	0.00
D	35.74	**42.12**	19.98	2.16	0.00
C	8.65	27.76	**53.76**	9.83	0.00
B	1.34	4.75	21.59	**63.39**	8.92
A	0.00	0.00	0.00	68.94	**31.06**

3.2.2 Results using two classes

The same analysis as using five classes is made using two classes. To be exact, table 6 shows the different results obtained by the best configurations. We want to remark that when the system uses the configuration Initial Load, Euclidean distance, Weights1, Test mode, for the proportion 50%-50%, obtains the maximum percentage of correctly classifiers (or prediction accuracy (PA)): 98.44%.

Table 6: Results obtained by CaB-CS using two classes.

	%PA	Configuration
Mean	87.65	Initial Load, Ham., Weights1, Test, 9010
Max.	**98.44**	Initial Load, Eucl., Weights1, Test, 5050
Min.	84.37	Initial Load, Eucl., Weights1, Test, 9010

4 Prediction using GA

Genetic Algorithms are methods inspired by the natural evolution of species [13, 9]. They maintain a set of potential solutions (*individuals*) to the problem being solved. This set, which is called *population*, is evolved by imposing mechanisms of selective pressure and recombination of the best individuals. At the end of the process, the population tends to converge to a "good solution", which is often the optimum or is very close to it.

Each individual has a *fitness* that measures how good it is relative to the population. The selection procedure is based on this fitness, so the best individuals have more chance of being selected and thus, more copies of the best individuals are obtained, while the worse individuals tend to disappear. The recombination mechanism is then applied by combining the structures of the new population, in order to explore new solutions and to improve the performance. Therefore, individuals with high fitness have more recombination probabilities and can spread their good characteristics through the population, leading the *genetic algorithm* to better areas of the search space.

A Genetic Algorithm is developed in order to obtain a set of rules capable of describing our classification problem. Traditionally, the application of Genetic Algorithms to Machine Learning problems, which have been called GBML (Genetic Based Machine Learning) systems, has been addressed from two different points of view: the Pittsburgh approach and the Michigan approach, early exemplified by LS-1 [20] and CS-1 [12] respectively.

In the Pittsburgh approach, each individual of the population represents a complete solution to the problem, which is a whole set of rules. On the contrary, the Michigan approach codifies only one rule in each individual. Therefore, the solution consists of all the population. This difference on representation leads to significant differences between both systems. Using the first approach, the GA can be applied directly. In the Michigan approach, the GA is limited to the exploration of new points of the search space and the learning process is performed by other algorithms (e.g. Bucket Brigade Algorithm [12], Q-Learning technique [24], etc.).

Our system is based on a previous one called GABL [21], which is developed under the Pittsburgh approach. We chose this approach because we are interested in obtaining rule sets that defines students behaviour.

4.1 Description of the Genetic Classifier System

We introduce here our own Genetic Based Classifier System called GAssist [3]. GAssist is a basic modular kernel designed for obtaining rules that cover a set of examples. It is based on the GABL [21] system, as early mentioned.

Representation: Each individual of the GA codifies a set of rules of variable size. A rule consists of a condition part and a classification part: *condition* $\rightarrow$ *classification*. The condition part is a conjunction of the tests over each feature: $if\ T_1 \wedge T_2 \wedge \ldots T_n$, where T_i is the test over the *ith* feature. The test of an feature is performed for all the nominal values of the feature, allowing an internal disjunction. For example, if the feature *Colour* can take the values { *Blue, Green, Yellow* }, a possible test can be: *if Colour is Blue or Green.* The examples that match the condition are classified by the classification part of the rule. This definitions can easily be translated to our classification problem. Features are the set of possible qualifications collected along the academic course, all nominally valued.

The internal representation of each rule is done in a binary string, whose length depends on the number of features and nominal values for each feature. As many bits represent a test over a feature as different nominal values it can have. Therefore, the previous feature *Colour* would be represented by 3 bits and the test *Colour is Blue or Green* would be codified as: 110. The classification part is also codified in a binary string.

Genetic operators: Several genetic operators have been tested, looking for a general overview of the system behaviour. The crossover operator used is *single-point crossover* (SPX) and *two-point crossover* (2PX), where cut points are randomly generated and can appear anywhere in the rule, not necessary in a rule boundary. Both operators generate semantically correct offsprings. Mutation has been implemented as the classical bit-level mutation. The selection process is performed by *RWS* (Roulette Wheel Selector) and *LRS* (Linear Ranking Selection). Both selection methods include *elitism* strategies.

Matching strategy: Two different matching strategies between rules and examples are tested. The first one (*List*) decodes the rules contained in individuals as a list of nested if clauses, thus the rule position into the set is important. The second strategy (*Specificity*) also decodes the rules as a list of nested if clauses, but the rule position depends on the specificity of the rule. More specific the rule is, earlier it is used.

Fitness computation: Individuals of the GA are evaluated by testing their sets of rules on the training set of examples using some of the previous matching strategies. Two different fitness functions (f_1, f_2) have been tested. They are:

$$f_1(ind_i) = \left(\frac{C}{T}\right)^2 \tag{4}$$

$$f_2(ind_i) = \left(\frac{C - \lambda \cdot N}{T}\right)^2 \tag{5}$$

Where C is the number of correctly classified examples, N is the number of not classified examples and T is the number of examples tested. λ is the parameter that controls the bias of individual through rule sets with few unclassified examples. It was empirically set to 0.25.

The fitness function is a key point for guiding the GA towards a good set of rules. Both fitness functions previously presented tends to reward individuals having a high percentage of correct classified examples and covering the maximum number of examples.

Stop criterion: Each phase of the Genetic Algorithm is applied iteratively until the stop criterion is reached. This criterion establishes the stop of execution when: (a) the best individual has a correct classification rate of 100% or (b) the maximum number of GA iterations has been reached.

4.2 Results

The testbed used for GA runs is the same as the one used for testing CaB-CS. For the whole testbed five representative proportion were used (10%-90%, 30%-70%, 50%-50%, 70%-30%, 90%-10%). Due to computational constrains, we decided to use a representative subset of the testbed for obtaining the preliminary results presented here.

For each *proportion* two versions with different random seeds were tested -10 possible *versions*-. For each *version* the exhaustive possible configurations of the GA where also tested growing up to 1280 runs. Table 7 presents the three better performing configurations, among all of the tested ones.

Table 7: Configurations used in GAssist.

	Cnf1	**Cnf2**	**Cnf3**
Selection	RWS	RWS	LRS
Crossover	2PX	SPX	2PX
Matching	Spec.	Spec.	List
Fitness	f_1	f_2	f_2

4.2.1 Results using five classes

Table 8 prints their performance using the five classes information distribution early mentioned. **Cnf1** is the GA configuration with better results.

Table 8: Best configuration (**Cnf1**) prediction results using 5 classes.

Prediction Real vs	*E*	*D*	*C*	*B*	*A*
E	**88.61**	9.49	1.90	0.00	0.00
D	33.90	**61.02**	5.08	0.00	0.00
C	6.25	35.94	**50**	7.81	0.00
B	13.33	0.00	33.34	**53.33**	0.00
A	14.29	0.00	0.00	85.71	**0.00**

The prediction accuracy distribution of **Cnf1** configuration best run is shown in table 8. Tables 9 and 10 present the rules obtained and their associated information.

Table 9: Rules obtained using **Cnf1**. Sorted by specificity.

$r3$	$TG \neq \{B \vee D\} \wedge SE = \{E \vee A\}$ $\wedge P1 \neq A \wedge P2 = \{E \vee D\} \rightarrow E$
$r2$	$TG \neq B \wedge FY \wedge SE = \{E \vee C\}$ $\wedge P1 = \{C \vee B \vee A\} \rightarrow C$
$r8$	$TG \neq G \wedge SE \neq D$ $\wedge P2 = \{E \vee F\} \rightarrow B$
$r7$	$SE = \{C \vee B \vee A\}$ $\wedge FE = \{C \vee B \vee A\} \wedge P2 \neq D \rightarrow B$
$r1$	$FE \neq B \wedge SE = \{E \vee D \vee C\}$ $\wedge P1 = \{C \vee B \vee A\} \rightarrow C$
$r4$	$SE = \{E \vee D \vee A\}$ $\wedge P2 = \{E \vee D\} \rightarrow E$
$r5$	$SE = \{D \vee A\} \wedge P1 \neq D$ $\wedge P2 \neq \{D \vee A\} \rightarrow D$
$r6$	$SE = \{E \vee A\}$ $\wedge P1 \neq D \wedge P2 \neq A \rightarrow E$

Table 10: Rules' performance information.

Rule	**Activation Order**	**Accuracy**
r1	5	68.08
r2	2	0.00
r3	1	not used
r4	6	74.03
r5	7	48.65
r6	8	90.22
r7	4	64.00
r8	3	0.00

4.2.2 Results using two classes

Table 11 summarises the results using two classes *pass* and *fail*. From this table follows that two different kinds of mistakes can be made. The first one is to predict that a student will fail and at the end of the course passes. This is not a serious mistake because we will just propose extra work to the student. The second kind of mistake is very serious. It says that the student will pass and at the end he/she fails. It will lead to a wrong relaxation of the learning pressure. We want to remark the low rate of this kind of mistakes, just a 2.76%.

Table 11: Best configuration (**Cnf1**) prediction results using 2 classes.

Prediction Real vs	*Fail*	*Pass*
Fail	**97.24**	2.76
Pass	31.68	**68.32**

5 Prediction using GA approach as a weighting method for CBR approach

One of the keys in order to obtain a good performance in a Case-Based Classifier System, as the CaB-CS, is having an accurate set of weights. The similarity functions, like Minkowski's metric, are sensitive in front of different weights. In this sense, a bad weighing may lead to a disastrous performance of the system. Although we are working on different automatic weighting techniques (like Sample Correlation [10], Shannon's Entropy [15] and using the Rough Sets theory [17, 18]), this paper analyses the possibility of interpreting the rules obtained by the Genetic Classifier System as a new value weighing method. We have observed that the rules obtained by the GA point out different weighing values in front of the relevance obtained from the experts.

5.1 Description of the hybrid method

We analysed the best set of rules obtained by the Genetic Classifier System. Then we weigh again the relevance of each feature used by the Case-Based Classifier System, proposing *weights2*.

Weights2. The *weights2* propose a set of weights for the available data, based on preliminary results using Genetic Algorithms. To be exact, the features have a feature weight value depending on the proportional appearance in the rules proposed by the GAssist. For instance, table 9 shows a representative subset of them. Analysing different sets of rules obtained by the GAssist, we propose the following weights: $w_1 = 0.0$, $w_2 = 0.5$, $w_3 = 0.25$, $w_4 = 2.5$, $w_5 = 0.75$ and $w_6 = 0.75$.

Then, we presented this proposal to the experts, and they made some pertinent modifications, obtaining the *weights3*.

Weights3. These weights become from the *weights2*. The weights obtained using the Genetic Classifier System were showed to the teachers of the corresponding subject, which tuned these values obtaining the following ones: $w_1 = 0.1$, $w_2 = 0.5$, $w_3 = 0.25$, $w_4 = 2.5$, $w_5 = 0.5$ and $w_6 = 0.5$.

5.2 Results

The hybrid method have been analysed using the same testbed defined at section 3.2. In the current testbed have been included the new weightings obtained by the Genetic Classifier System: *Weights2* and *Weights3*.

5.2.1 Results using five classes

Analysing these results, we can observe -respectively of the previous ones- that not only the system outperforms the results, but also the combination between a weighting criterion and a similarity function is very sensitive, obtaining easily opposed behaviours. Table 12 shows these results. When the system use the Initial Load, the Euclidean distance, the Weights2, the Test mode, for the proportion 90%-10% obtains the best results: 85.94%.

Table 12: Results obtained by the hybrid method using five classes.

	%PA	Configuration
Mean	67.82	Initial Load, Clark, Weights3, Test, 9010
Max.	**85.94**	Initial Load, Eucl., Weights2, Test, 9010
Min.	60.23	Init. Load, Ham., Weights3, DifClas, 2080

But these results are very good if we analyse where the system makes a mistake (see table 13). Mainly, the mistakes have been made among the neighbour classes. Due to help students is the goal, making a slight misprediction among neighbour classes is meaningless because their boundaries are blur.

Table 13: Error analyse of the results obtained using five classes, for the best configuration of the table 12.

Prediction Real vs	E	D	C	B	A
E	**81.66**	14.93	3.36	0.05	0.00
D	35.49	**42.17**	20.38	1.96	0.00
C	9.37	21.34	**57.66**	11.41	0.22
B	2.79	4.63	23.80	**62.61**	6.17
A	0.00	0.00	4.57	57.68	**37.76**

Figure 1 shows the mean behaviour of the different weightings, as a function of the initial size of the case memory. These printed results are obtained using the Hamming distance. We want to outline that *weights2* and *weights3* improve the previous results of CaB-CS.

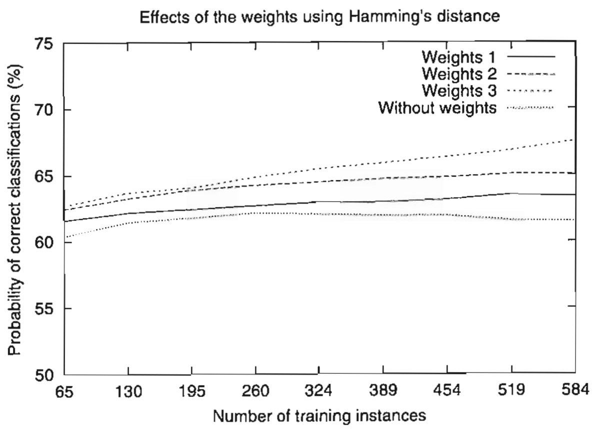

Figure 1: The mean behaviour of the different weighting criteria using five classes for the Hamming distance.

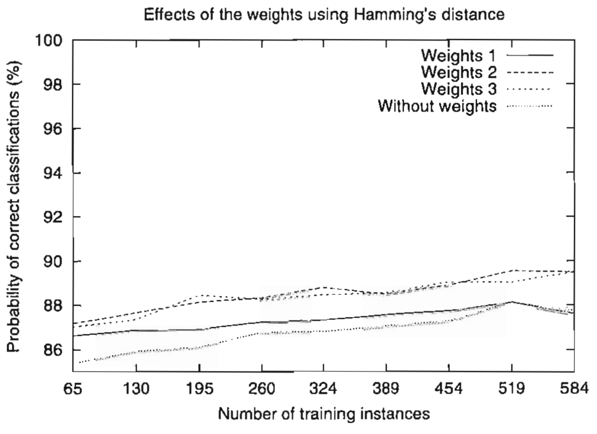

Figure 2: The mean behaviour of the different weighting criteria using two classes for the Hamming distance.

5.2.2 Results using two classes

In this section, we present the same experiments, when the system predicts using only two classes.

Table 14 shows the results (percentage of the prediction accuracy) obtained for the best configurations.

Table 14: Results obtained by the Hybrid method using two classes.

	%PA	Configuration
Mean	90.66	Initial Load, Cubic, Weights3, Test, 9010
Max.	**98.44**	Initial Load, Cubic, Weights3, Test, 9010
Min.	87.04	Ini. Load, Eucl., Weights3, DifClas, 5050

Table 15 shows that using two classes, the system -for the best configuration- only make one error (corresponding to 2.27%), but this mistake ends favourable to the student.

Table 15: Error analyse of the results obtained using two classes, for the best configuration of the table 14.

Prediction Real vs	*Fail*	*Pass*
Fail	**97.73**	2.27
Pass	0.0	**100.0**

Finally, figure 2 shows the mean behaviour of the different weightings using two classes. These results are also obtained using the Hamming distance. Again, *weights2* and *weights3* improve the previous results using CaB-CS.

6 Conclusions and further work

The preliminary results of our work show that the use of a Genetic Classifier System as a heuristic weighting method for a Case-Based Classifier System outperforms the results reached using both systems independently. These results point out that this combination among both methods would be a good hybrid system to work on. Besides, we can verify that the systems obtain a notable accuracy when they predict the final student's qualification.

The future work will be focused on a deeper study of the integration of both systems (Genetic Classifier System and Case-Based Classifier System), in order to create a real hybrid method. The future work will also evaluate this proposal in multiple domains. We also want to analyse better this hybrid -as an automatic weighting method- comparing its performance with others automatics methods: a) in respect of own automatic methods like Sample Correlation [10], Shannon's Entropy [15] and using the Rough Sets theory [17, 18]; b) in respect of the automatic method proposed by the literature like the summary of methods proposed by D. Aha [2]. Finally, we are planning to integrate the prediction system to its final scenario: an open-distance learning tool based on a WWW platform.

Acknowledgements

We would like to thank the *Comisión Interministerial de Ciencia y Tecnología* for its support under grant number CICYT/Tel98-0408-02. The results of this work were obtained using the equipment co-funded by the *Direcció General de Recerca de la Generalitat de Catalunya (D.O.G.C 30/12/1997)*. We would also like to thank the teachers of the subject *Programming* of the Computer Science Department at Enginyeria i Arquitectura La Salle (EALS), Universitat Ramon Llull (URL). And finally, we would like to thank EALS for their support to our AI Research Group.

References

[1] Aamodt, A. and Plaza, E., Case-Based Reasoning: Foundations Issues, Methodological Variations, and System Approaches, *AI Communications*, 7 (1994), 39-59.

[2] Aha, D.W., *Feature weighting for lazy learning algorithms*, Technical Report AIC-98-003, Navy Center for Applied Research in AI, Washington D.C., (1998).

[3] Bacardit, J., *Creació d'un assistent digital per la predicció del fracàs acadèmic basat en computació evolutiva*, Projecte Final de Carrera, Enginyeria i Arquitectura La Salle, Universitat Ramon Llull, (Juliol 2000).

[4] Bernadó, E. and Garrell, J.M. and Román, M. and Salamó, M. and Camps, J. and Abella, J., Introducción a la programación en el ámbito de diversas ingenierías, *IV Jornadas de Enseñanza Universitaria de Informática* (JENUI'98) (1998), 435-442.

[5] Brusilovsky, P. and Schwarz, E. and Weber G., ELM-ART: An intelligent tutoring system on world wide web, *Proceedings of the Third International Conference on Intelligent Tutoring Systems* (ITS'96), Montreal, (1996), 261-269.

[6] De Jong, K.A. and Spears, W.M., Learning Concept Classification Rules Using Genetic Algorithms, *Proceedings of the Twelfth International Joint Conference on Artificial Intelligence*, Morgan Kaufmann (1991), 651-656.

[7] Garrell, J.M., Golobardes, E., Bernadó, E., Llorà, X., Automatic Classification of mammary biopsy images with machine learning techniques, *Proceedings of First International ICSC Symposium on Engineering of Intelligent Systems* (EIS'98), ICSC Academic Press, Vol. 3 (1998), 411-418.

[8] Garrell, J.M. and Golobardes, E. and Bernadó, E. and Llorà, X., Automatic diagnosis with Genetic Algorithms and Case-Based Reasoning, *Artificial Intelligence in Engineering* 13(4), Elsevier Science Ltd. (1999), 367-372.

[9] Goldberg, D.E., *Genetic Algorithms in Search, Optimization and Machine Learning*, Addison-Wesley Publishing Company, 1989.

[10] Golobardes, E. and Garrell, J.M., Avaluació d'un sistema classificador basat en casos per a la diagnosi de biòpsies de teixit de glàndules mamàries, *Proceedings de les Jornades d'Intel·ligència Artificial (JIA'97)*, 1997.

[11] Golobardes, E. and Vernet, D. and Salamó, M., Prediction in an Educational Environment using Case-Based Reasoning, *Learning'00* (IEEE), (2000), To appear.

[12] Holland, J.H., Escaping Brittleness: The Possibilities of General Purpose Learning Algorithms applied to Parallel Rule-Based Systems, *Machine Learning: An Artificial Intelligence Approach, Vol.II*, pages 593-623, 1986.

[13] Holland, J.H., *Adaptation in Natural and Artificial Systems*, MIT Press, 1992.

[14] Llorà, X. and Golobardes, E. and Salamó, M. and Martí, J., Diagnosis of microcalcifications using Case-Based Reasoning and Genetic Algorithms, *Second International ICSC Symposium on Engineering of Intelligent Systems* (EIS'2000) (2000), 254-263

[15] Martí, J. and Español, J. and Golobardes, E. and Freixenet, J. and Garcia, R. and Salamó, M., Classification of microcalcifications in digital mammograms using Case-Based Reasoning, *Fifth International Workshop on Digital Mammography* (IWDM-2000) (2000)

[16] Riesbeck, C.K. and Schank, R.C., *Inside Case-Based Reasoning*, Lawrence Erlbaum Associates, Hillsdale, NJ, US (1989).

[17] Salamó, M. and Golobardes, E. and Vernet, D. and Nieto, M., Weighting methods for a Case-Based Classifier System, *Learning'00* (IEEE), (2000), To appear.

[18] Salamó, M. and Golobardes, E., BASTIAN: incorporating the Rough Sets theory into a Case-Based Classified System, *3r Congrés Català d'Intel·ligència Artificial* (CCIA'2000), (2000), To appear.

[19] Self, J., Theoretical Foundations for Intelligent Tutoring Systems, *Journal of Artificial Intelligence in Education*, 1 (4), (1990), 3-14.

[20] Smith, F., Flexible Learning of Problem Solving Heuristics through Adaptive Search, *Proceedings of the 8th International Joint Conference on Artificial Intelligence*, pages 422-425, 1983.

[21] Spears, W.M. and De Jong, K.A., Using Genetic Algorithms For Supervised Concept Learning, *Machine Learning*, 13 (1993), 161-188.

[22] Vernet, D., *Avaluació de les tècniques de Raonament Basat en Casos aplicades a un entorn educatiu*, Projecte Final de Carrera, Enginyeria i Arquitectura La Salle, Universitat Ramon Llull, (Juliol 2000).

[23] Watson, I. and Marir, F., Case-Based Reasoning: A review, *The Knowledge Engineering Review*, 9(4) (1994), 327-354.

[24] Wilson, S.W., Classifier Fitness based on Accuracy, *Evolutionary Computation*, 3(2):149-175, 1995.