ZCS Revisited: Zeroth-level Classifier Systems for Data Mining

Fani A. Tzima* and Pericles A. Mitkas
Dept. of Electrical and Computer Engineering,
Aristotle University of Thessaloniki,
GR-541 24, Thessaloniki, Greece
fani@olympus.ee.auth.gr, mitkas@eng.auth.gr

Abstract

Learning classifier systems (LCS) are machine learning systems designed to work for both multi-step and single-step decision tasks. The latter case presents an interesting, though not widely studied, challenge for such algorithms, especially when they are applied to real-world data mining problems. The present investigation departs from the popular approach of applying accuracy-based LCS to data mining problems and aims to uncover the potential of strength-based LCS in such tasks. In this direction, ZCS-DM, a Zeroth-level Classifier System for data mining, is applied to a series of real-world classification problems and its performance is compared to that of other state-of-the-art machine learning techniques (C4.5, HIDER and XCS). Results are encouraging, since with only a modest parameter exploration phase, ZCS-DM manages to outperform its rival algorithms in eleven out of the twelve benchmark datasets used in this study. We conclude this work by identifying future research directions.

1 Introduction

Learning classifier systems (LCS), originally proposed by Holland [17], belong to a class of machine learning systems designed to work for both single-step and sequential decision problems [29]. LCS employ a population of classifiers (usually rules in the traditional production system form) gradually evolving through the use of a reinforcement scheme and a genetic algorithm based search component. Although modern LCS research has mainly focused on the case of sequential decision problems, lately there has been a shift of interest toward applying LCS to single-step decision tasks, such as predictive data mining [22].

Data mining (DM) employs a variety of supervised or unsupervised learning algorithms, in order to automatically search large volumes of data and derive patterns that can be used for either *predictive* (classification/regression) or *descriptive* tasks (association rule mining, clustering, etc.). In the context of our current work, DM is used for classification, that can be formally defined as "...the task of learning a target function f that maps each attribute set x to one of the predefined class labels y" [30].

Among the various methods used to tackle classification problems, decision trees [25] and rule-based classifiers are particularly popular, because they combine:

i) intuitive representation that allows for easy interpretation of the resulting classification model;

ii) a nonparametric nature that is especially suited for exploring datasets where there is no prior knowledge of the attributes' probability distributions;

iii) fast, computationally inexpensive construction methods that produce models storable in a compact form; and

iv) fast classification of new observations, once the model has been constructed – worst-case complexity is O(depthOfTree) for decision trees or O(numberOfRules) for rule-based classifiers.

Inspecting the above list, one can easily conclude that LCS share most of the advantages of these methods, with the exception of the third point, as genetic algorithm-based search is an arguably slow and computationally expensive search method. Nevertheless, LCS preserve the advantages of intuitiveness and fast post-training classification, while adding the ability to evolve a ruleset that is ordered and may be used to produce both categorical decisions and outcome possibilities. For an example of LCS applied to a data mining task where the risk (possibility) of an outcome rather than a single categorical decision is essential to the targeted domain, the reader is referred to [19].

Having realized the potential of LCS as a data mining tool, attempts toward this direction date back to 1991 with

*Corresponding author.

the work of Bonelli and Parodi [4]. They were the first to investigate the effectiveness of a LCS called NEWBOOLE [5] on three medical data mining domains and reported that it could outperform its rival algorithms (CN2 and neural networks for that study). In another application for medical purposes, Holmes and colleagues successfully applied their modified version of NEWBOOLE [19, 3] to epidemiological databases, showing LCS potential in applications requiring adaptable models of risk. Starting in the late 1990s, and spurred by the introduction of accuracy-based LCS (XCS) by Wilson in [32], more investigations appeared supporting the claim that XCS could outperform traditional machine learning methods in several domains: the Wisconsin Breast Cancer Dataset [33], the monk's problems [28], the COIL 2000 Direct Marketing Competition [16], clinical research databases [20] and several other real-world datasets [10].

Despite the ongoing interest and impressive results applying accuracy-based LCS to data mining problems, the present work departs from this popular approach and takes a step forward, aiming to uncover the potential of strength-based LCS (and particularly Zeroth-level Classifier Systems – ZCS [31]) in data mining tasks. In this direction, a series of real-world data mining problems are tackled and the performance of our modified ZCS implementation, namely ZCS-DM, is compared with that of other state-of-the-art machine learning techniques (C4.5, HIDER and XCS). Results are encouraging, in that with only a modest parameter exploration phase, ZCS manages to outperform its rival algorithms in most of the benchmark data mining problems used in this study.

The rest of the paper is structured as follows: Section 2 provides a high-level description of ZCS from a data mining point of view, while Section 3 goes into more detail regarding representation and modeling choices of our modified ZCS implementation. Following a brief description of the benchmark problems and rival algorithms in Section 4, we continue with the presentation of our experimental setup and results in Section 5. This work is concluded in Section 6 with some insights on the factors affecting ZCS performance, depending on the characteristics of the data mined and the algorithm parameters. Future research directions are also identified, with the ultimate goal of providing a robust and effective data mining tool based on ZCS.

2 A Brief Description of ZCS

The following is a high level description of ZCS, focusing on the components necessary for our current investigation of single-step decision tasks. For a more in-depth description and parameter settings see [31] and [6].

A Zeroth-level Classifier System employs a population $\mathbf{R}$=$\{R_1, R_2, ..., R_N\}$ of gradually evolving, cooperative classifiers, each encoding a fraction of the problem domain and, thus, collectively forming the overall solution to the target problem. Rules (classifiers) in ZCS are represented in the traditional production form of "IF *condition* THEN *action*" and are encoded over the ternary alphabet 0,1,#. The symbol # (usually termed as a *"wildcard"* or a *"don't care"*) allows for generalization in the rule condition part, such that both inputs 11 and 10 are matched by the rule-condition 1#. No generalization occurs in the action part – actions are discrete and usually integer-valued. Associated with each classifier, there is a scalar strength value expressing its expected reward.

Through each operation cycle (step t), ZCS receives a binary encoded input vector V_t from its environment, determines an appropriate response based on a rule (or a set of rules) whose condition matches the input, and produces a classification decision. Successful classification of an instance is associated with a scalar reward $\mathfrak{R}$ apportioned to the system classifiers according to a reinforcement scheme. Thus, at each discrete time-step, the system follows a cycle of *performance*, *reinforcement* and *discovery* component activation. This cycle is presented in algorithmic form below (Algorithm 1) and described in more detail in subsequent sections.

2.1 Performance Component

Upon receipt of a data instance from its environment, the system scans the population of classifiers for those whose condition matches the current input and forms the *matching set* $\mathbf{M}$. An action (classification decision) α is, then, selected among those advocated by rules in $\mathbf{M}$. Depending on the exploration-exploitation strategy chosen for the particular problem, action selection may be *deterministic* ($detAS = true$), with the action advocated by the strongest classifier being selected, or *stochastic* ($detAS = false$), where each action has a selection probability proportional to the sum of strengths of classifiers advocating it in $\mathbf{M}$. Next, the *action set* $\mathbf{A}$ is formed, containing all members of $\mathbf{M}$ advocating the selected action α. Finally, the selected action is forwarded to the effector interface, which has the responsibility of returning the classification decision to the environment.

2.2 Reinforcement Component

In case of a correct classification decision, the system receives an immediate reward $\mathfrak{R}$ and increments the strengths of all classifiers in $\mathbf{A}$ by $\mathfrak{R}/|\mathbf{A}|$, where $|\mathbf{A}|$ is the number of classifiers in $\mathbf{A}$. Next, all classifiers in the set difference $\mathbf{M} - \mathbf{A}$ (that is classifiers advocating actions other than the selected one) are penalized by reducing their strength by a fraction p.

It is worth noting that, when applied to multi-step tasks,

ZCS employs a more elaborate reinforcement mechanism that propagates the received rewards backwards, thus forming chains of cooperating rules, essential to achieving optimal performance in non-Markovian environments.

Algorithm 1 ZCS component activation cycle (at step t)

START Cycle
$V_t := readNextDataInstance()$
$\mathbf{M} := \emptyset$, $\mathbf{A} := \emptyset$
`== performance component ==`
for $i = 1$ to $|\mathbf{R}|$ **do**
  if $condition(R_i)$ matches V_t **then**
    $\mathbf{M} := \mathbf{M} \cup \{R_i\}$
  end if
end for
if $(\forall R_i \in \mathbf{M}: strength(R_i) < \phi \cdot \overline{strength}(\mathbf{R})) \vee (\mathbf{M} = \emptyset)$ **then**
  `== covering operator activation ==`
  $\alpha := randomDecision$
  $R_{new} := createRule(V_t, \alpha)$
  $\mathbf{R} := \mathbf{R} \cup \{R_{new}\}$
else
  $\alpha := chooseDecision(\mathbf{M}, detAS)$
  for $j = 1$ to $|\mathbf{M}|$ **do**
    if $action(R_j)$ matches α **then**
      $\mathbf{A} := \mathbf{A} \cup \{R_j\}$
    end if
  end for
  `== reinforcement component ==`
  if α matches $class(V_t)$ **then**
    for $k = 1$ to $|\mathbf{A}|$ **do**
      $strength(R_k) := strength(R_k) + \Re/|\mathbf{A}|$
    end for
  end if
  for $j = 1$ to $|\mathbf{M}|$ **do**
    if $(R_j \in \mathbf{M}) \wedge (R_j \notin \mathbf{A})$ **then**
      $strength(R_j) := strength(R_j) \cdot (1 - p)$
    end if
  end for
  `== GA-based rule discovery ==`
  if (t mod ρ) == 0 **then**
    $\{R_{new1}, R_{new2}\}$:= $applyGA(\mathbf{R})$
    $\mathbf{R} := \mathbf{R} \cup \{R_{new1}, R_{new2}\}$
  end if
end if
END Cycle

2.3 Discovery Component

ZCS employs two rule discovery mechanisms: (i) a *steady-state genetic algorithm (GA)* and (ii) a *covering operator*.

The *GA* is invoked at a rate ρ, approximating the intervals needed for classifier strengths to settle to steady-state values. The evolutionary process employs strength-proportional parent selection: two parent classifiers are selected based on their strength and copied to form two offspring that are inserted in the classifier population after crossover and mutation operators have been applied to them with given probabilities. Parent and offspring strengths are accordingly adjusted, in order to conserve the initial total parent strength. In order to maintain a constant population size, inverse strength-proportional deletion is applied, when necessary.

The *covering operator* is activated either when the match set $\mathbf{M}$ is empty, or when there is no rule in $\mathbf{M}$ with strength greater than a fixed fraction ϕ of all classifiers' average strength. Covering creates a new classifier with a random action part and a condition part matching the current input and generalized by inserting # symbols with a given probability g. Inverse strength-proportional deletion is applied, again, if necessary, to conserve the population size constant after the insertion of the newly created classifier.

3 ZCS-DM: Modifications and Implementation

3.1 Rule Representation

Our choice of rule representation follows the simplest form possible for data mining tasks involving numeric attributes: the attribute-value (or propositional logic) representation [13]. According to this, the rule condition part consists of a conjunction of predicates of the form $<Attribute|Operator|Value>$, where $Operator \in \{\geq, <, \#\}$. The operator '=' has been deliberately left out, in order to avoid overly specific conditions that are generally meaningless for numeric attributes. An example of a rule for the Iris dataset, conforming to our implementation of the attribute-value representation, could be "$petallength < 9.4$ AND $petalwidth \geq 1.8$".

The chosen representation can readily accommodate real-valued attributes, as well as missing values by treating the predicates involving the corresponding attributes as always being true. Datasets with nominal attributes can be handled by the implemented representation by considering nominal values as ordered lists of integers. For example, we take a rule of the form "$waterProjectCostSharing \geq 1$ AND $physicianFeeFreeze < 1$" for the Congressional Voting Records dataset (whose attributes are all binary) to mean that someone voted *for* the first issue (waterProjectCostSharing = 1) and *against* the second (physicianFeeFreeze = 0). A more general approach that allows predicates of the form $<Attribute|BelongsTo|SetOfValues>$, may

be necessary for nominal attributes with more than two possible values.

As far as the rule action part (predicted class) is concerned, it is worth noting that, unlike traditional GA-based data mining algorithms, we follow the trend in LCS implementations and allow for the class attribute to be evolved along with its corresponding rule antecedent. This approach does not elude the drawback of associating perfectly good classifiers (rules with large coverage scores) with the wrong class, thus deteriorating their strength score and, possibly, leading the GA and deletion operations to make these rules extinct [12]. ZCS, though, seems to be inherently capable of circumventing this problem, by having the GA search the rule condition space for an adequate decomposition of the problem at hand into subproblems. Thus, each rule in the population provides the solution to at least one of these subproblems, while the reinforcement component evaluates the classifier predictions.

3.2 Further Implementation Issues

As already mentioned in Section 2, action selection in the performance component can be either deterministic or strength-proportional. Our implementation, namely ZCS-DM, allows for both methods, with the boolean parameter $detAS$ allowing the user to select between the two. However, following early experimentation with both methods, our method of choice, used in all experiments reported in Section 5, is deterministic action selection, where the action advocated by the strongest classifier in $\mathbf{M}$ is selected as the system decision.

Moving on to the reinforcement component, our implementation allows for cost-sensitive model building, offering the user the ability to define tunable rewards per class (using a diagonal $rewardMatrix$) and penalties per class-pair (using a $penaltyMatrix$ with 0-elements along its main diagonal). That is, the user can specify different reward values for correct decisions, according to the (actual) class of the instance being predicted. Likewise, penalties can be formulated to map the intended fault-tolerance of the system. For example, misclassification of instances of a particular class may be deemed less "acceptable" than others, if the former carry a potentially greater cost.

The GA used in ZCS-DM (and in LCS in general) is a steady-state GA, where individuals are changed one by one without the notion of generation used in classical (generational) GAs. This mechanism smooths the "disruptive" effects that may occur in generational GAs and allows for a more subtle interaction between the evolutionary search process and the reinforcement learning component.

Regarding the genetic operators, the *crossover operator* is implemented as a simple one-point crossover, invoked with probability c and applied on two parent classifiers selected using roulette-wheel selection. *Copying* of a classifier (again selected using roulette-wheel selection) is employed at a rate 1-c. *Mutation* is uniform, flipping each bit in a classifier with probability m. Mutation is not allowed to change the action part of a rule into a non-existing class value. In case this happens, the action part of the rule is reset to a random value from the allowed class labels range.

Finally, *covering* creates a new classifier with a condition part matching the current input and a random action in the allowed class labels range. *Generalization* occurs with probability g during covering, setting the operator part of a condition to #, thus rendering the corresponding predicate always true. Obviously, such generalization can also occur due to the crossover and mutation operators during the GA search phase.

3.3 Evolved Ruleset Use

The ruleset evolved by ZCS-DM contains up to N classifiers, where N is a user-defined parameter. The final classifier population (but also the population at any point during the training process) may contain classifiers that are identical. This may be a product of rule duplication or the combined effects of the genetic and covering/generalization operators.

In order to use the final set of rules effectively, a preprocessing step is required, during which the system discards redundant classifiers, keeping only one copy of each rule in the population. The remaining classifiers' strengths, of course, are increased by the sum of strengths of their deleted copies. Thereafter, the ruleset may be used as *ordered*, with the strongest classifier matching a certain input imposing its decision, or in combination with a (possibly weighted) *voting scheme*.

Further processing is possible in an effort to produce reduced or more easily understandable classifier sets [34, 11]. Such operations, however, are beyond the scope of our current investigation.

4 Benchmark Datasets and Rival Algorithms

4.1 Benchmark Datasets

The benchmark datasets employed in this work are listed in Table 1 and they are all readily available from the UCI repository [2]. A major factor for the choice of the particular datasets was the availability of corresponding results for two state-of-the-art evolutionary algorithms, namely XCS [10] and HIDER [1]. Furthermore, the selected datasets are all from real-world domains and, arguably, present a diverse challenge for the algorithms under comparison, since they comprise a mixture of nominal and numeric attributes, a wide range of attributes numbers (4-60) and classes (2-7),

Table 1. Datasets used in experiments: attribute number and types, number of classes, missing values and number of instances

Dataset	Attributes	Classes	Missing Values	Instances
Tic-Tac-Toe Endgame	9 nominal	2	0	958
Iris	4 numeric	3	0	150
Zoo	15 nominal + 2 numeric	7	0	101
Breast Cancer Wisconsin [23]	9 numeric	2	16	699
Congressional Voting Records	16 nominal	2	392	435
Wine	13 numeric	3	0	178
Lenses	4 nominal	3	0	24
Hepatitis	13 nominal + 6 numeric	2	167	155
Pima Indians Diabetes	8 numeric	2	0	768
Connectionist Bench (Sonar)	60 numeric	2	0	208
Bupa Liver Disorders	6 numeric	2	0	345
Glass Identification	9 numeric	6	0	214

several dataset sizes (24-958 instances) and some cases of missing values.

4.2 Rival Algorithms

HIDER [1], an acronym for HIerarchical DEcision Rules, employs a sequential covering GA [24], implementing the most basic type of "niching" method: the algorithm produces a single rule for each run, sequentially eliminating correctly covered instances from the training set supplied to the GA, until the instance space is totally covered. The result of this process is a hierarchical set of ordered rules, wherein the meaning and relative importance of any rule is dependent on all the rules preceding it in the ruleset. During the testing phase and for each instance presented to the system, the ruleset is scanned and the first matching rule is used to produce a classification decision.

The results presented in the next sections for HIDER are taken from Aguilar et al. [1] and correspond to experiments run using their original implementation of the above algorithm.

XCS, originally proposed by Wilson in [32], is a LCS where the focus is on decoupling the reinforcement component from the evolutionary search process. This is achieved by the use of a separate fitness function expressing the accuracy of the classifier's prediction of reward (and not the expected reward as in ZCS). A further difference, between ZCS and XCS is that the latter employs a niched genetic algorithm which is applied only on classifiers accountable for recently performed actions (classifiers in the current action set $\mathbf{A}$). For a more detailed algorithmic description of XCS the user is referred to [8] and [7].

The results presented in the next sections for XCS are taken from Dixon et al. [10] and correspond to experiments run using their own customized implementation that is not able to cope with real-valued attributes directly, but uses a preprocessing step of discretization to circumvent this shortcoming.

Finally, **C4.5** is the well-known decision tree induction algorithm developed by Ross Quinlan [27] as an extension of his earlier work on the ID3 algorithm [26]. C4.5 builds decision trees from a set of training data in the same way as ID3, using the concept of Information Entropy and can only be used for classification, i.e. it is not applicable to regression tasks.

The results presented in the next sections for C4.5 are taken from Aguilar et al. [1] and correspond to experiments run using the code distributed with Quinlan's book [27].

5 Experiments and Results

5.1 Experimental Setting and Parameter Setup

Table 2 gives the parameter values used through all experiments reported in subsequent sections. The only parameter not introduced so far is the number of iterations I, which expresses the number of complete passes through the training set during the algorithm training phase. With a value of I=100 and a dataset of 200 instances, for example, the system would be presented with each training example 100 times and, thus, would perform 20000 cycles of performance, reinforcement and discovery component activation, before arriving to the final ruleset to be used for testing.

As far as parameter exploration is concerned, inspecting the above table, one can easily observe that only a subset of the available (tunable) parameters were kept constant, while for the rest a modest exploration process was used in order

Table 2. ZCS-DM parameters through all conducted experiments: parameters varied are the ones with values indicated as ranges

Parameter	Description	Value
N	Number of rules	200-400
I	Number of iterations	100
$detAS$	Deterministic action selection	true
S	Initial rule strength	100
$\mathfrak{R}$	Correct classification reward (except for cost-sensitive models)	1000
p	Penalty (except for cost-sensitive models)	0.5-0.7
τ	Tax for classifiers in **NOTA**	0.1
ρ	GA invocation rate	1
c	Crossover probability	0.15
m	Mutation probability	0.005
g	Generalization probability	0.33-0.5
ϕ	Covering invocation threshold	0.5

to acquire the best values per dataset [1]. This exploration was mainly guided by the dataset size and complexity, in terms of the number of attributes and classes, and involved tuning the allowed number of rules (N) and the penalty (p) and generalization probability (g) parameters.

Overall, we may infer the following rules of thumb from the parameter exploration phase:

i) the required number of rules is proportional to the size of the target dataset;

ii) increased generalization may prove useful for datasets with large numbers of attributes; and

iii) stricter penalties (usually combined with larger numbers or rules, due to increased rates of covering invocation) result in the system producing more accurate classification decisions.

5.2 Comparative Analysis of Results

Following the approach adopted by [1] and [10], ZCS-DM was applied to all twelve datasets listed in Table 1, using ten-fold stratified cross-validation. Comparison of the results is based on the accuracy rate of each of the algorithms and summarized in Table 3. No further comparison was possible (for example, based on the algorithms' detailed accuracy by class), as such figures are not available in [1] and [10].

[1] Details on the parameter setting per dataset are available on request.

Regarding the results presented in Table 3, it is worth noting that along with all other results, a baseline accuracy (second column) is also included, indicating the performance of a minimally experienced classifier that always predicts the dataset's prevalent class.

Based on the measured accuracy results, our approach clearly dominates the other three, with best results on eleven out of the twelve datasets. This is more accurately depicted in Fig. 1, where the ranking of the four algorithms (vertical axis) is schematically presented against the twelve benchmark datasets (horizontal axis). The average rank (last row of Table 3) also provides a clear indication of the studied algorithms' relative performance, with ZCS-DM receiving an average rank of 1.29 and XCS being the next to follow with an average rank of 2.38.

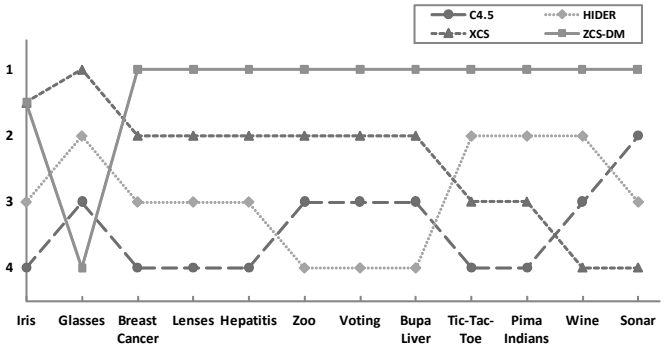

Figure 1. Algorithm ranking against the benchmark datasets

It is also worth noting, that, with the exception of the Glass Identification dataset, ZCS-DM performs considerably well even for datasets with skewed class distributions. A representative example is the Tic-Tac-Toe Endgame dataset, which clearly presents difficulties to two of the rival algorithms (C4.5 and XCS). Exploring the cost-sensitive model building (in terms of rewards) capabilities of our system, though, we managed to evolve rulesets far more accurate than those produced with the rival algorithms. The strategy we employed for determining the adjusted rewards in case of cost-sensitive model building was to make the rewards inversely proportional to the prevalence rate of each class. For example, for the Tic-Tac-Toe Endgame dataset, where there are two classes with prevalence rates 65% and 35% respectively, the following $rewardMatrix$ was used:

$$\mathbf{rewardMatrix} = \begin{bmatrix} 1000 & 0 \\ 0 & 1850 \end{bmatrix}$$

Finally, as far as execution times are concerned, C4.5 is obviously considerably faster than its rival algorithms,

Table 3. Classification accuracy per algorithm using ten-fold cross-validation. The ranks in parentheses are used in computation of the Friedman test.

Dataset	Baseline(%)	C4.5(%)	HIDER(%)	XCS (%)	ZCS-DM(%)
Tic-Tac-Toe Endgame	65.34	85.80 (4)	96.15 (2)	85.91 (3)	98.33 (1)
Iris	33.33	95.33 (4)	96.67 (3)	98.00 (1.5)	98.00(1.5)
Zoo	40.59	93.00 (3)	92.00 (4)	96.27 (2)	98.00 (1)
Breast Cancer Wisconsin [23]	65.52	93.72 (4)	95.71 (3)	96.27 (2)	96.99 (1)
Congressional Voting Records	60.38	93.81 (3)	93.36 (4)	94.23 (2)	96.55 (1)
Wine	39.89	93.29 (3)	96.05 (2)	92.74 (4)	96.08 (1)
Lenses	62.50	70.01 (4)	75.00 (3)	78.33 (2)	90.00 (1)
Hepatitis	79.35	78.58 (4)	80.59 (3)	81.29 (2)	85.04 (1)
Pima Indians Diabetes	65.10	67.94 (4)	74.10 (2)	68.62 (3)	74.48 (1)
Connectionist Bench (Sonar)	53.37	69.69 (2)	56.93 (3)	53.41 (4)	72.52 (1)
Bupa Liver Disorders	57.97	65.27 (3)	64.29 (4)	67.85 (2)	69.30 (1)
Glasses Identification	35.51	67.27 (3)	70.59 (2)	72.53 (1)	63.55 (4)
Average Rank	–	3.42	2.92	2.38	1.29

which all use evolutionary search techniques. While no explicit reporting of execution times is available for HIDER and XCS, Dixon et al. [10] note that "both [of them] may typically take several hours on a modern desktop PC to provide a single ten-fold cross validation result on one of the larger of the databases tested". For our implementation of ZCS, namely ZCS-DM, execution times for a single cross-validation run, with each fold involving 100 full passes through the training set, varied from 15 seconds (for the smallest dataset – Lenses) to 30 minutes (for the largest dataset – Tic-Tac-Toe Endgame – and the dataset with the largest number of attributes – Connectionist Bench) on a Pentium 3 GHz with 1 GB of RAM.

5.3 Statistical Evaluation of Results

In order to evaluate the statistical significance of the measured differences in algorithm ranks, we have used the procedure suggested by Demsar [9] for robustly comparing classifiers across multiple datasets. This procedure involves the use of the Friedman test [14, 15] to establish the significance of the differences between classifier ranks and, potentially, a post-hoc test to compare classifiers to each other. In our case, the goal was to compare the performance of the rival algorithms to that of our ZCS implementation (control classifier). As noted by Demsar [9], pairwise comparisons should not be used "when we in fact only test whether a newly proposed method is better than the existing ones". Thus, the Holm procedure [18] was selected as the appropriate post-hoc test.

The Friedman test [14, 15] is a non-parametric statistical test for evaluating the differences between more than two related sample means - where the related samples are, in our case, the performances of k classifiers across N target datasets. The null hypothesis being tested is that all classifiers perform the same and any observed differences are merely random. This is equivalent to testing whether the measured average ranks r_j (j=$[1, N]$) are significantly different from the expected mean rank $\overline{r} = \frac{k+1}{2}$. The statistic used in the Friedman test

$$\chi_F^2 = \frac{12N}{k(k+1)}\left[\sum_j r_j^2 - \frac{k(k+1)^2}{4}\right]$$

is distributed according to χ_F^2 with k-1 degrees of freedom when N and k are big enough.

Iman and Davenport [21] showed that Friedman's χ_F^2 is undesirably conservative and derived a better statistic

$$F_F = \frac{(N-1)\chi_F^2}{N(k-1)-\chi_F^2}$$

which is distributed according to the F-distribution with $k-1$ and $(k-1)(N-1)$ degrees of freedom.

Provided that the Friedman test rejects the null hypothesis, we can proceed with a post-hoc test that evaluates the relative performance of the studied algorithms against a control algorithm r_0. One such test is the Holm step-down procedure [18] that tests hypotheses sequentially ordered by their significance. The z statistic comparing the i-th classifier against the control one

$$z = (r_i - r_0) \Big/ \sqrt{\frac{k(k+1)}{6N}} = \frac{(r_i - r_0)}{\sqrt{SE}}$$

is used to compute the corresponding probabilities from the table of normal distribution, which are, in turn, compared with an appropriate α. Given the ordered p values

$p_1, p_2, ..., p_{k-1}$ (such that $p_1 \leq p_2 \leq ... \leq p_{k-1}$), the Holm procedure starts with the most significant value and compares each p_i with the adjusted value $\alpha/(k-i)$ to compensate for multiple $(k-1)$ comparisons. If p_i is below $\alpha/(k-i)$, the corresponding hypothesis is rejected and we proceed to comparing p_{i+1} with $\alpha/(k-i-1)$. As soon as a certain null hypothesis cannot be rejected, all the remaining hypotheses are retained as well.

The procedure described above has been applied to the data in Table 3, which compares the four studied algorithms: C4.5, HIDER, XCS and ZCS-DM. As already stated, average ranks provide a fair comparison of the algorithms, revealing that, on average, ZCS-DM and XCS rank second, while C4.5 and HIDER rank third. Given the measured average ranks, the Friedman test checks whether the average ranks are significantly different form the mean rank $\overline{r} = 2.5$ expected under the null hypothesis:

$$\chi_F^2 = \frac{12 \cdot 12}{4 \cdot 5}\left[(3.42^2 + 2.92^2 + 2.38^2 + 1.29^2) - \frac{4 \cdot 5^2}{4}\right] \Rightarrow$$

$$\chi_F^2 = 17.93$$

$$F_F = \frac{11 \cdot 17.93}{12 \cdot 3 - 17.93} = 10.91$$

With four algorithms and 12 data sets, F_F is distributed according to the F-distribution with 3 and 33 degrees of freedom. The critical value of $F(3, 33)$ for $\alpha = 0.05$ is 2.89, so we reject the null hypothesis.

Having found that the measured average ranks are significantly different (at $\alpha = 0.05$), we proceed with the Holm post-hoc test to decide whether our proposed method, namely ZCS-DM, is significantly better than its rival algorithms. We first have to compute the corresponding statistic and p values. The standard error is $SE = \sqrt{\frac{4 \cdot 5}{6 \cdot 12}} = 0.527$, while the z statistic and corresponding p values for the hypotheses to be tested are depicted in Table 4. The Holm procedure rejects all hypotheses, since the corresponding p values are smaller than the adjusted α values in all cases. Our analysis, thus, reveals that *at $\alpha = 0.05$, the performance of all rival algorithms (C4.5, HIDER, XCS) is significantly worse than that of ZCS-DM*.

Table 4. Z statistic and p values calculated for the post-hoc Holm test, comparing the performance of ZCS-DM to its rival algorithms. ($R_0 = 1.29, \alpha = 0.05$)

i	Algorithm	$z = (R_i - R_0)/SE$	p	α/i
1	C4.5	4.03	0.0001	0.0167
2	HIDER	3.08	0.0021	0.0250
3	XCS	2.06	0.0394	0.0500

6 Conclusions and Further Work

In this paper, we have presented an experimental investigation of the potential of strength-based LCS, and particularly Zeroth-level Classifier Systems, in data mining tasks. Following a detailed discussion on the mechanisms underlying ZCS and the modifications of ZCS-DM, our customized implementation of the algorithm, we went on to present a comparative analysis of the latter's performance against three widely known and quite successful data mining algorithms, namely Quinlan's C4.5, Aguilar et al.'s HIDER and Dixon et al.'s implementation of XCS. This analysis was based on results obtained from the use of our algorithm on twelve real-world datasets and corresponding results reported by the authors of the other algorithms in [1] and [10].

Overall, ZCS-DM proved to be a robust and accurate data mining tool, managing to outperform its rival algorithms in most of the benchmark datasets used in this study and achieving a prediction accuracy well above the baseline on all of them. Furthermore, evaluation of the experimental results using the Holm procedure of the Friedman test, a robust method for comparing classifiers over multiple datasets, reveals that at $\alpha = 0.05$, the performance of all rival algorithms (C4.5, HIDER, XCS) is significantly worse than that of ZCS-DM. Therefore, we consider the presented results particularly encouraging, especially when taking into account that this first set of experiments did not include an elaborate fine-tuning of the parameters.

Several issues deserve further investigation in order to obtain maximal results. A first step toward this direction is the design and implementation of a more in-depth exploration strategy for the algorithm parameters and their effect on system performance. Further, we believe the implementation of more general rule representations to be a priority, as such a step holds the promise of boosting our algorithm's expressive ability and performance. We envisage these generalized rule representations to be able to deal, on one hand, with conditions of the form "attribute$_i \in$ [x,y]" for numeric attributes, and, on the other hand, with conditions of the form "attribute$_i \in \{$value$_k$, ..., value$_r\}$" in the nominal case. Such an effort, of course, will have to be combined with suitable modifications of the genetic operators and, more importantly, post-training processing steps to ensure the consistency and minimality of the evolved rulesets.

Acknowledgments

This paper is part of the 03ED735 research project, implemented within the framework of the "Reinforcement Programme of Human Research Manpower" (PENED) and cofinanced by National and Community Funds (25% from the

Greek Ministry of Development-General Secretariat of Research and Technology and 75% from E.U.-European Social Funding).

References

[1] J. Aguilar-Ruiz, J. Riquelme, and M. Toro. Evolutionary Learning of Hierarchical Decision Rules. *IEEE Transactions on Systems, Man, and Cybernetics, Part B*, 33(2):324–331, 2003.

[2] A. Asuncion and D. Newman. UCI machine learning repository [http://www.ics.uci.edu/∼mlearn/MLRepository.html]. Irvine, CA: University of California, School of Information and Computer Science, 2007.

[3] A. Barry, J. H. Holmes, and X. Llorá. Data Mining Using Learning Classifier Systems. In L. Bull, editor, *Applications of Learning Classifier Systems. Studies in Fuzziness and Soft Computing*, pages 15–67. Springer, Heidelberg, 2004.

[4] P. Bonelli and A. Parodi. An Efficient Classifier System and Its Experimental Comparison with Two Representative Learning Methods on Three Medical Domains. In R. K. Belew and L. B. Booker, editors, *Proceedings of the 4th International Conference on Genetic Algorithms (ICGA)*, pages 288–295, San Diego, CA, USA, 1991. Morgan Kaufmann.

[5] P. Bonelli, A. Parodi, S. Sen, and S. Wilson. NEWBOOLE: a fast GBML system. In *Proceedings of the 7th International Conference on Machine Learning*, pages 153–159, San Francisco, CA, USA, 1990. Morgan Kaufmann Publishers Inc.

[6] L. Bull and J. Hurst. ZCS redux. *Evolutionary Computation*, 10(2):185–205, 2002.

[7] M. Butz, T. Kovacs, P. Lanzi, and S. Wilson. Toward a theory of generalization and learning in XCS. *Evolutionary Computation, IEEE Transactions on*, 8(1):28–46, 2004.

[8] M. Butz and S. W. Wilson. An Algorithmic Description of XCS. In *IWLCS '00: Revised Papers from the Third International Workshop on Advances in Learning Classifier Systems*, pages 253–272, London, UK, 2001. Springer-Verlag.

[9] J. Demšar. Statistical Comparisons of Classifiers over Multiple Data Sets. *Journal of Machine Learning Research*, 7:1–30, 2006.

[10] P. W. Dixon, D. Corne, and M. J. Oates. A Preliminary Investigation of Modified XCS as a Generic Data Mining Tool. In P. L. Lanzi, W. Stolzmann, and S. W. Wilson, editors, *Advances in Learning Classifier Systems, 4th International Workshop, IWLCS '01, Revised Papers*, volume 2321 of *Lecture Notes in Computer Science*, pages 133–150, London, UK, 2002. Springer-Verlag.

[11] P. W. Dixon, D. W. Corne, and M. J. Oates. A Ruleset Reduction Algorithm for the XCS Learning Classifier System. In *Learning Classifier Systems, 5th International Workshop, IWLCS 2002, Granada, Spain, September 7-8, 2002, Revised Papers*, volume 2661 of *Lecture Notes in Computer Science*, pages 20–29. Springer Berlin / Heidelberg, 2003.

[12] A. A. Freitas. *Data Mining and Knowledge Discovery with Evolutionary Algorithms*. Springer-Verlag New York, Inc., Secaucus, NJ, USA, 2002.

[13] A. A. Freitas. *Evolutionary Algorithms for Data Mining*, volume The Data Mining and Knowledge Discovery Handbook, pages 435–467. Springer, January 2005.

[14] M. Friedman. The Use of Ranks to Avoid the Assumption of Normality Implicit in the Analysis of Variance. *Journal of the American Statistical Association*, 32(200):675–701, 1937.

[15] M. Friedman. A Comparison of Alternative Tests of Significance for the Problem of m Rankings. *The Annals of Mathematical Statistics*, 11(1):86–92, 1940.

[16] A. Greenyer. Coil 2000 Competition: The use of a learning classifier system JXCS. Technical report, The Database Group, Colston Tower, Colston Street, Bristol.

[17] J. H. Holland. *Adaptation in Natural and Artificial Systems: An Introductory Analysis with Applications to Biology, Control and Artificial Intelligence*. University of Michigan Press, Ann Arbor, MI, USA, 1975.

[18] S. Holm. A Simple Sequentially Rejective Multiple Test Procedure. *Scandinavian Journal of Statistics*, 6:65–70, 1979.

[19] J. H. Holmes. Discovering Risk of Disease with a Learning Classifier System. In T. Bäck, editor, *Proceedings of the 7th International Conference on Genetic Algorithms (ICGA97)*, San Francisco, CA, 1997. Morgan Kaufmann.

[20] J. H. Holmes and J. A. Sager. Rule Discovery in Epidemiologic Surveillance Data Using EpiXCS: An Evolutionary Computation Approach. In S. Miksch, J. Hunter, and E. T. Keravnou, editors, *AIME*, volume 3581 of *Lecture Notes in Computer Science*, pages 444–452. Springer, Heidelberg, 2005.

[21] R. L. Iman and J. M. Davenport. Approximations of the critical region of the Friedman statistic. *Communications in Statistics*, A9(6):571–595, 1980.

[22] P. Lanzi. Learning Classifier Systems: Then and Now. *Evolutionary Intelligence*, 1(1):63–82, 2008.

[23] O. L. Mangasarian and W. H. Wolberg. Cancer Diagnosis via Linear Programming. *SIAM News*, 23(5):1 & 18, 1990.

[24] T. M. Mitchell. *Machine Learning*. McGraw-Hill Higher Education, 1997.

[25] S. K. Murthy. Automatic Construction of Decision Trees from Data: A Multi-Disciplinary Survey. *Data Mining and Knowledge Discovery*, 2(4):345–389, 1998.

[26] J. R. Quinlan. Induction of Decision Trees. *Machine Learning*, 1(1):81–106, 1986.

[27] J. R. Quinlan. *C4.5: programs for machine learning*. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 1993.

[28] S. Saxon and A. Barry. XCS and the Monk's Problems. In *Learning Classifier Systems, From Foundations to Applications*, pages 223–242, London, UK, 2000. Springer-Verlag.

[29] O. Sigaud and S. W. Wilson. Learning Classifier Systems: a survey. *Soft Computing*, 11(11):1065–1078, 2007.

[30] P.-N. Tan, M. Steinbach, and V. Kumar. *Introduction to Data Mining, (First Edition)*. Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA, 2005.

[31] S. W. Wilson. ZCS: A Zeroth-level Classifier System. *Evolutionary Computation*, 2(1):1–18, 1994.

[32] S. W. Wilson. Classifier fitness based on accuracy. *Evolutionary Computation*, 3(2):149–175, 1995.

[33] S. W. Wilson. Mining Oblique Data with XCS. In P. L. Lanzi, W. Stolzmann, and S. W. Wilson, editors, *Advances in Learning Classifier Systems, Third International Workshop, IWLCS 2000, Paris, France, September 15-16, 2000, Revised Papers*, volume 1996 of *Lecture Notes in Computer Science*, pages 158–176. Springer, Heidelberg, 2001.

[34] S. W. Wilson. Compact Rulesets from XCSI. In *IWLCS '01: Revised Papers from the 4th International Workshop on Advances in Learning Classifier Systems*, pages 197–210, London, UK, 2002. Springer-Verlag.