Using the Grid to improve the effectiveness of Learning Classifier Systems through clustering-based initialization

Fani Tzima, Fotis Psomopoulos, and Pericles Mitkas
Greece
Introduction
- Learning Classifier Systems (LCS)
- Different LCS flavors

ZCS-DM: Algorithmic description
- Rule representation
- Operation Cycle
- Clustering-based Initialization Component

Experiments and Results
- Experimental Setting for leveraging the Grid Infrastructure
- Qualitative Interpretation of Results
- Statistical Comparison of Results

Conclusions and Future Work
Learning Classifier Systems (LCS) [Holland, 1976] are a machine learning technique designed to work for both single-step and sequential decision problems.

LCS employ a population of classifiers (usually rules in the production system form) gradually evolving through the use of a reinforcement scheme and a GA-based search component.
• Smith’s approach, from the University of Pittsburgh \(\Rightarrow\) GA applied to a population of LCSs in order to choose the fittest

• “Michigan style” LCSs employ a population of gradually evolving, cooperative classifiers \(\Rightarrow\) each classifier encodes a fraction of the problem domain
Different LCS Flavors

- **Strength-based LCSs (ZCS)**
  - each classifier contains only one evaluation variable → both an estimation of the accumulated reward brought by its firing and its fitness for the population evolution

- **Accuracy-based LCSs (XCS)**
  - decoupling the RL process and the population evolution → fitness function not proportional to the expected reward, but to the accuracy of the latter’s prediction

- **Anticipatory LCSs (ALCS)**
  - [Condition] [Action] → [Effect] classifiers (instead of [Condition]→[Action])
  - [Effect] represents the expected effect (next state)
Rule representation

- Traditional production form of
  
  \[ \text{IF } \text{condition} \ \text{THEN} \ \text{action} \ [\text{Strength}] \ [\text{Fitness}] \]

- Condition comprises predicates of the form
  
  \(<\text{Attribute} \in \text{SetOfNominalValues} \mid \text{NumericInterval}>\)

- Encoded over the ternary alphabet \(0, 1, \#\).
  - The symbol \(\#\) ("wildcard" or "don’t care") allows for generalization.

- Actions are discrete

Both inputs \(11\) and \(10\) are matched by the rule condition \(1\#\)
ZCS-DM operation cycle

INITIALIZATION
• Initialize population of classifiers \( R = \{R_1, R_2, \ldots, R_N\} \)

PERFORMANCE COMPONENT
• Receive a binary encoded input vector \( V \)
• Determine an appropriate response based on the rules whose condition matches the input
• Produce a classification decision and update rules’ fitness values

REINFORCEMENT COMPONENT
• In case of successful classification, apportion a scalar reward \( R \) to the system classifiers according to a reinforcement scheme

DISCOVERY COMPONENT
• Change one individual of the classifier population by applying GA
Method Pros and Cons

✓ Intuitive representation
✓ Applicable for datasets where there is no prior knowledge of the attributes’ probability distributions
✓ Production of models storable in a compact form
✓ Fast (post-training) classification of new observations
✓ Resulting ruleset is ordered

Grid Resources for Parameter Optimization and Parallel Execution of Experiments

✗ Non-deterministic nature of the algorithm + Relatively long training times
✗ multiple experiments to reach statistically sound conclusions
✗ Large number of tunable parameters
Clustering-based Initialization

Start the evolutionary process from a non-random set of rules

Design a clustering-based initialization component

- Focus on the search-space optima more effectively and quickly

Clustering algorithms

- Clustering provides a representative set of points (centroids) for a given dataset

Transform centroids into rules suitable for the initialization phase
- Boost algorithm performance: predictive accuracy + training times
3 possible condition parts for the case of 2 numeric attributes
### Experimental Setting

**Evaluation of 4 versions of the algorithm**

- **ClusterInit100**
  Clustering-based initialization – Full training time (100 iterations)

- **RandomInit100**
  Random ruleset initialization – Full training time (100 iterations)

- **ClusterInit75**
  Clustering-based initialization – Reduced training time (75 iterations)

- **RandomInit75**
  Random ruleset initialization – Reduced training time (75 iterations)
### “Vanilla-Setup” Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>N</td>
<td>Number of rules</td>
<td>400</td>
</tr>
<tr>
<td>I</td>
<td>Number of iterations</td>
<td>100/75</td>
</tr>
<tr>
<td>detAS</td>
<td>Deterministic action selection</td>
<td>True</td>
</tr>
<tr>
<td>S</td>
<td>Initial rule strength</td>
<td>100</td>
</tr>
<tr>
<td>R</td>
<td>Correct classification reward</td>
<td>1000</td>
</tr>
<tr>
<td>p</td>
<td>Penalty (except for cost-sensitive models)</td>
<td>0.5</td>
</tr>
<tr>
<td>τ</td>
<td>Tax for classifiers in NOTA</td>
<td>0.1</td>
</tr>
<tr>
<td>ρ</td>
<td>GA invocation rate</td>
<td>0.5</td>
</tr>
<tr>
<td>c</td>
<td>Crossover probability</td>
<td>0.15</td>
</tr>
<tr>
<td>m</td>
<td>Mutation probability</td>
<td>0.005</td>
</tr>
<tr>
<td>g</td>
<td>Generalization probability</td>
<td>0.1</td>
</tr>
<tr>
<td>φ</td>
<td>Covering invocation threshold</td>
<td>0.1</td>
</tr>
<tr>
<td>NC</td>
<td>Number of clusters</td>
<td>10</td>
</tr>
<tr>
<td>gc</td>
<td>Clustering generalization rate</td>
<td>0.5</td>
</tr>
</tbody>
</table>

**Number of iterations I** expresses the number of complete passes through the training set during the algorithm training phase.
### Benchmark Datasets

<table>
<thead>
<tr>
<th>Dataset</th>
<th>Attributes</th>
<th>Classes</th>
<th>Missing Values</th>
<th>Instances</th>
</tr>
</thead>
<tbody>
<tr>
<td>Balance Scale Weight &amp; Distance</td>
<td>4 nominal</td>
<td>3</td>
<td>0</td>
<td>625</td>
</tr>
<tr>
<td>Bupa Liver Disorders</td>
<td>6 numeric</td>
<td>2</td>
<td>0</td>
<td>345</td>
</tr>
<tr>
<td>Car Evaluation</td>
<td>6 nominal</td>
<td>4</td>
<td>0</td>
<td>1728</td>
</tr>
<tr>
<td>Contraceptive Method Choice</td>
<td>7 nominal + 2 numeric</td>
<td>3</td>
<td>0</td>
<td>1473</td>
</tr>
<tr>
<td>Hepatitis</td>
<td>13 nominal + 6 numeric</td>
<td>2</td>
<td>167</td>
<td>155</td>
</tr>
<tr>
<td>Pima Indians Diabetes</td>
<td>8 numeric</td>
<td>2</td>
<td>0</td>
<td>768</td>
</tr>
<tr>
<td>Connectionist Bench (Sonar)</td>
<td>60 numeric</td>
<td>2</td>
<td>0</td>
<td>208</td>
</tr>
<tr>
<td>Tic Tac Toe Endgame</td>
<td>9 nominal</td>
<td>2</td>
<td>0</td>
<td>958</td>
</tr>
<tr>
<td>Congressional Voting Records</td>
<td>16 nominal</td>
<td>2</td>
<td>392</td>
<td>435</td>
</tr>
<tr>
<td>Breast Cancer Wiskonsin</td>
<td>9 numeric</td>
<td>2</td>
<td>16</td>
<td>699</td>
</tr>
<tr>
<td>Wine</td>
<td>13 numeric</td>
<td>3</td>
<td>0</td>
<td>178</td>
</tr>
</tbody>
</table>

- 20 x 10-fold stratified cross-validation runs
- Comparison of the results based on accuracy rate
Comparative analysis of results

ClusterInit100 1.32
ClusterInit75 2.64
RandomInit100 2.68
RandomInit75 3.36
Statistical Evaluation of Results

- Statistical procedure [Demsar, 2006] for robustly comparing classifiers across multiple datasets
  - use the **Friedman test** to establish the significance of the differences between classifier ranks
  - use a **post-hoc test** to compare classifiers to each other
- In our case, the goal was to compare the performance of all algorithms to each other
  - the **Nemenyi test** was selected as the appropriate post-hoc test

**At \( \alpha = 0.05 \), the performance of the clustering-based initialization approach with full training times is *significantly better* than that of all its rivals.**

**At \( \alpha = 0.05 \), the performance of the clustering-based initialization approach with reduced training times is *NOT significantly different* than that of the baseline approach with full training times.**
Comparison of Execution Times (1/2)

- Execution time (sec) on **personal computer** (Intel Core 2 Duo, CPU @2.00GHz – 4,00 GB RAM) Vs. the Grid Infrastructure

Single 10-fold Cross-Validation Run

- 25% improvement compared to PC times

- Grid - Clust100
- PC - Clust100
- Grid - Rand100
Grid DAG workflow

- Define concurrent execution strategy
- Parallel execution of experiments
  - Merging of results
  - Perform statistical tests and formulate output
  - Algorithms to be compared
  - Statistical tests
  - Parameter settings

Enabling Grids for E-sciencE

EGEE-III INFSO-RI-222667
Comparison of Execution Times (2/2)

- Execution time (hrs) on **personal computer** (Intel Core 2 Duo, CPU @2.00GHz – 4.00 GB RAM) Vs. the **Grid Infrastructure**

20 x 10-fold Cross-Validation Runs + Aggregate Statistical Tests

Scripted parallel execution on the Grid + automatic aggregation of results is 27 times faster than sequential execution on PC
• Clustering-based initialization proved to be a useful component
  – achieving the best prediction accuracy (on average) when full training times were employed
  – performing equally well with the baseline approach, even when reduced training times were employed

• The concurrent utilization of Grid resources allowed for an effective and time-efficient way to perform parameter optimization and/or algorithm comparison experiments

• The Grid is the ideal execution environment due to the embarrassingly parallel nature of the problem
  – jobs submitted simultaneously (organized in a DAG workflow)
  – different parameter set → independence of jobs
Future Work

• Design and implementation of a more in-depth parameter exploration strategy to be evaluated on the Grid infrastructure
  – effect on system performance

• Post-training processing steps
  – consistency and compactness of evolved rulesets

• Evaluation of the algorithm as an on-line data-mining tool for real-world domains (such as urban Air Quality)
  – the nature of the algorithm and the capability of LCS to tackle multi-step decision problems are encouraging
Thank you for your attention!

Fotis Psomopoulos
fpsom@issel.ee.auth.gr

Intelligent Systems and Software Engineering Labgroup
Informatics and Telematics Institute
Centre for Research and Technology-Hellas
Thessaloniki, Greece

Electrical and Computer Eng. Dept.
Aristotle University of Thessaloniki
START
for k = 1 to numberOfAttributes do

    if (Math.random() <= GENERALIZATION_RATE) then
        Switch activation bit of condition k off
    else
        Switch activation bit of condition k on
    end if

== NOMINAL ATTRIBUTES ==
if attribute_k is nominal then
    SetOfValues := ∅
    for all possible values of attribute_k
        if (Math.random() <= 0.5) then
            SetOfValues := SetOfValues ∪ currentValue
        end if
    end for
    SetOfValues := SetOfValues ∪ centroid.values[k]
    Create condition k as $attribute_k \in SetOfValues$

== NUMERIC ATTRIBUTES ==
else
    low_value = centroid.minValue
    high_value = centroid.maxValue
    Create condition k as $attribute_k \in [low\ value, high\ value]$
end if
Add condition k to the RuleConditionPart
end for
END
• Non-parametric statistical test for evaluating the differences between more than two related sample means

  – Performances of **k classifiers across N target datasets** (average ranks)

  \[ \chi^2_F = \frac{12N}{k(k+1)} \left[ \sum_j R_j^2 - \frac{k(k+1)^2}{4} \right] \]

  \( R_j \): average rank of j-th algorithm on i-th dataset

  – **Null hypothesis** (all classifiers perform the same and any observed differences are merely random) rejected if

  \[ F_F > F_{\text{critical}} (k-1,(k-1)*(N-1)) \]

  statistic distributed according to the F-distribution with \( k-1 \) and \( (k-1)*(N-1) \) degrees of freedom
• Evaluates the **relative performance** of all classifiers to each other

• The performance of two algorithms is **significantly different** if the corresponding average ranks differ by at least the **critical difference** $CD$

\[
CD = q_a \sqrt{k(k + 1)/6N}
\]

Critical values $q_a$ are those of the Studentized range statistic divided by $\sqrt{2}$ with a significance level of $\alpha$ and $k$ degrees of freedom.