Hierarchical Evolution of Linear Regressors

Francesc Teixidó-Navarro, Albert Orriols-Puig, and Ester Bernadó-Mansilla
Grup de Recerca en Sistemes Intel·ligents
Enginyeria i Arquitectura La Salle, Universitat Ramon Llull
Quatre Camins, 2, 08022 Barcelona, Spain
{fteixido,aorriols,esterb}@salle.url.edu

ABSTRACT

We propose an algorithm for function approximation that evolves a set of hierarchical piece-wise linear regressors. The algorithm, named HIRE-Lin, follows the iterative rule learning approach. A genetic algorithm is iteratively called to find a partition of the search space where a linear regressor can accurately fit the objective function. The resulting ruleset performs an approximation to the objective function formed by a hierarchy of locally trained linear regressors. The approach is evaluated in a set of objective functions and compared to other regression techniques.

Categories and Subject Descriptors

I.2.6 [**Learning**]: concept learning, knowledge acquisition

General Terms

Algorithms

Keywords

Genetic algorithms, machine learning, function approximation, regression

1. INTRODUCTION

Genetic algorithms (GAs) have proved to be valuable in machine learning and data mining applications. Particularly, genetic algorithms have been used in classification problems. Therein, a model is required to describe the relationship between the characteristics of the examples provided in a dataset and their associated class. Some of the benefits offered by genetic algorithms are the domain independence, the ability to evolve several types of representations (e.g., rulesets, trees), and high performance.

Among the current approximations dealing with rulesets, the Michigan approach [14] evolves a set of overlapping classifiers that together approximate the class boundary. The approach evolves a set of individuals that are incrementally evaluated. Since each individual codifies a single rule, the GA has to balance the competition-cooperation tradeoff to achieve a set of optimal classifiers jointly approximating the class boundary. Hence, the Michigan approach uses a genetic algorithm that searches simultaneously for several subsolutions that together can cover the whole search space. The Pittsburgh approach [23, 24] evolves a population of rulesets, where each ruleset usually works as a decision list [20]. The algorithm searches for the best ruleset among the set of possible rulesets. Thus, the search space is larger than in Michigan approaches; however, the evolutionary pressures can be adjusted to obtain simpler rulesets (e.g., see [3]). The so-called iterative rule learning (IRL) [13, 25] approach, also referred to as sequential covering algorithm [18], can alternatively be used to evolve rulesets. The IRL approach allows the GA to search for a single rule in each iteration. Each time a rule is obtained, the region of the search space covered by the rule is removed for the subsequent searches. Search complexity is bounded in each iteration in two respects. First, the evolution of a single rule in each iteration provides less complexity than in the Pittsburgh approach. Second, the search space is progressively reduced in each iteration. The evolved ruleset must be evaluated in order, analogously to a decision list.

Due to the benefits of reduced complexity, IRL algorithms are valuable to address large datasets. Moreover, the rule sets are highly interpretable because they contain fewer rules, possibly comparable to Pittsburgh rulesets.

Recently, some learning classifier systems such as XCS [29, 30] have been extended to deal with numeric prediction [28]. Numeric prediction (also called *regression*) can be seen as a variant of classification learning where the class is a numerical value rather than a category [31]. Herein, the emphasis of the learner is to perform function approximation. Much research has been conducted recently on Michigan approaches, particularly XCSF [28, 15, 8], and also on Pittsburgh approaches [4] for function approximation. In this paper, we extend the IRL approach to numeric prediction applications. The proposed system, named HIRE-Lin, evolves iteratively a set of linear regressors performing piece-wise linear approximations. Our aim is to propose a new architecture for the evolution of hierarchical linear regressors based on genetic algorithms, and thus, we wish to inherit the GA's capabilities such as robustness, domain independence, simplicity, and interpretability. Such capabilities will be evaluated and compared to classical approaches for regression. Moreover, such an approach would offer compound benefits from the Michigan and Pittsburgh approaches: a bounded search space complexity and high interpretable results.

GECCO'08, July 12–16, 2008, Atlanta, Georgia, USA.

The remainder of the paper is structured as follows. The next section describes HIRE-Lin for function approximation. Section 3 describes the experimental methodology. Section 4 analyzes HIRE-Lin on a case study and evaluates its behavior with different settings. Next, we compare HIRE-Lin with other types of regression techniques. Finally, we conclude and present future work lines.

2. ALGORITHM

HIRE-Lin is an Iterative Rule Learning (IRL) algorithm, inspired by Hider [1], that evolves a hierarchical rule set that approximates functions using linear regressors. At each iteration, the algorithm searches for a single rule that best approximates the data. The rule is added to the final ruleset, and instances covered by the rule are removed. The algorithm continues iteratively until the data set is covered. The final ruleset must be interpreted in the same order as rules were produced. The search is based on a genetic algorithm. The details of HIRE-Lin are as follows.

2.1 Rule Representation

HIRE-Lin iteratively evolves a rule set that works as a *decision list* as proposed in [20][1]. Given an example $\mathbf{e}$ and a ruleset H, each rule $r_i \in H$ is checked in order until a matching rule is found. That is, rule r_m will predict a given example $\mathbf{e}$ if there is not any preceeding rule $r_i \,|\, i < m$, that covers the example. In such a case, the linear predictor coded in rule r_m will be used to approximate the objective function $f(\mathbf{x})$ at $\mathbf{x} = \mathbf{e}$. Each rule has the form:

$$r_i = X \rightarrow W \tag{1}$$

where X stands for the condition (or antecedent), and W (the consequent) corresponds to the linear regressor applied when the condition is satisfied by the example. The condition defines a hyperrectangle in the search space, represented as a sequence of intervals $(x_1, x_2, \ldots, x_\ell)$, where ℓ is the dimension of the feature space. Each interval x_i is defined by its lower and upper bounds $[lb_i, ub_i]$, both real-valued. An example $\mathbf{e} = (e_1, e_2, \ldots, e_\ell)$ satisfies the rule if $\forall_i : 1 \leq i \leq \ell : (lb_i \leq e_i \leq ub_i)$.

The consequent, W, represents the parameters of a linear regressor. Given an example $\mathbf{e} = (e_1, e_2, \ldots, e_\ell)$, the linear regressor approximates $f(e)$ by the hyperplane:

$$y_i = w_0 + w_1 e_1 + w_2 e_2 + ... + w_\ell e_\ell \tag{2}$$

where $w_0, w_1, \ldots, w_\ell$ are the regressor parameters. Thus, a rule defines a hyperplane W which is applicable in the attribute domain defined by the hyperrectangle coded by X. The number of parameters of the linear regressor is $(\ell + 1)$. The final length of the rule is $(3\ell + 1)$, from where 2ℓ correspond to the antecendent and the remainder $\ell + 1$ belong to the consequent.

The linear regressor is computed by simple regression [12, 19] according to the least squares criterion. We used the multi-dimensional least squares fitting routine available with GNU Scientific Library [1] (GSL).

2.2 Learning Process

The algorithm of HIRE-Lin is depicted in Alg. 1. Given a dataset E, the algorithm iteratively evolves a hierarchical ruleset H. In each iteration, a genetic algorithm (GA) is fired to search for the best rule covering accurately a high number of instances of the dataset E. The best rule returned by the GA is added to H. Next, the instances covered by the rule are removed from the dataset. The process is repeated until E is empty. Finally, a default rule is added into ruleset H. Its antecedent covers the entire search space, although it will be only applicable when the previous rules do not match. The consequent is a rough approximation computed as the average of the value of the objective function of all training points.

Algorithm 1: Algorithm of HIRE-Lin.

```
setOfExamples E;
Rule r;
setOfHierarchicalRules H;
H := ∅;
while E not empty do
    r := evolve(E);
    H := H + r;
    E := E - CoveredInstances(r);
H := H + defaultRule;
```

2.3 Genetic Algorithm

Given a dataset E, the GA searches for the best rule that approximates the dataset. The search goal is to find the *best rule that approximates accurately the highest number of instances of* E, i.e., to search for the largest hyperrectangle that can be accurately approximated by a linear predictor.

The GA evolves a population P of N individuals, where each individual codifies a rule as described in equation 1. Each rule is a vector of $(3\ell + 1)$ real numbers. The GA only modifies the antecedent of the rule, which is of size 2ℓ. The consequent of the rule are the parameters of the linear predictor which are obtained by least squares.

2.3.1 Initialization

In the initialization phase, a population P is created. Each individual contains a rule which is initialized in two steps. First, the antecedent X of each rule is initialized using an example $\mathbf{e}$ randomly selected from the dataset E. For each attribute $e_i \,|\, i : 1 \ldots \ell$, an interval $[lb_i, ub_i]$ containing e_i is set according to: $lb_i = e_i - v_c$ and $ub_i = e_i + v_c$, where v_c is a value uniformly distributed in the interval $[0, r_0]$, being $r_0 \geq 0$ a configuration parameter. Both values are limited to the range of the attribute. Then, the consequent W is calculated as follows. A linear regressor is computed by a least squares procedure, considering only the examples enclosed in the hyperrectangle defined by the antecedent X.

2.3.2 Fitness function

The fitness of the rule codifies the search goals: to maximize the hyperrectangle while minimizing the approximation error of the linear regressor.

Given a rule r, a linear regressor is computed using only the instances covered by the hyperrectangle. Then, the fitness of the rule is computed as follows:

$$F(r) = coverage(r) * acc(r)^\gamma \tag{3}$$

where $coverage(r)$ is the portion of the search space covered by the rule, acc is the accuracy of the approximation, and γ is a user-defined parameter. $0 \leq coverage(r) \leq 1$ is the

[1] http://www.gnu.org/software/gsl

ratio of the subspace covered by the hyperrectangle divided by the search space defined by the original training dataset:

$$\begin{aligned} coverage(r) &= \prod_{i=1}^{\ell} \frac{coverage(r,i)}{range(i)} \quad (4) \\ coverage(r,i) &= ub_i - lb_i \\ range(i) &= UB_i - LB_i \end{aligned}$$

where $coverage(r,i)$ is the size of the interval of rule r for attribute i, and $range(i)$ is the difference between the maximum (UB_i) and minimum (LB_i) values of attribute i. *acc* considers the quality of the linear approximation and is defined as:

$$acc = R^2 \quad (5)$$

where R^2 is the coefficient of determination [19] of the linear regressor, computed as follows:

$$R^2 = 1 - \frac{SS_E}{SS_T} \quad (6)$$

where SS_E is the sum of squared errors and SS_T is the total sum of squares of the function value of the corresponding points. R^2 is the proportion of variability in the training points that is accounted for by the regressor model. That is, R^2 is a statistic that provides information on how well the regression line approximates the real data points. An R^2 of 1.0 indicates that the regression line perfectly fits the data. Thus, *coverage* represents the generalization of the rule, while *acc* is the accuracy or quality of the linear regressor trained for that rule. $\gamma \geq 1$ is a parameter that specifies the relevance of the accuracy term with respect to the generalization term. In the remainder of the paper, we will refer to it as *accuracy pressure* parameter.

Note that generalization and accuracy are two objectives that must be maximized. Our fitness function takes an aggregating approach [5], being γ a control parameter that specifies the relative weight of these two objectives.

2.3.3 Genetic Operators

Selection of individuals is performed via *tournament selection* with tournament size S, where $0 < S \leq N$. Mutation is applied with probability p_m per gene. Let g_i be a gene representing either the lower or upper bound of an interval. The value is mutated to a value uniformly distributed in the range $[g_i - m_{off}, g_i + m_{off}]$, where m_{off} is a parameter. The new value is restricted to the range of the corresponding attribute so that the resulting interval is correct. The crossover operator is applied with probability p_c. Given two parents, two children are obtained that replace their parents in the population. One point crossover is implemented which chooses a cut point uniformly distributed in the range $[1..2\ell]$, where ℓ is the number of attributes. Elitism is applied to preserve the best solution found from one cycle to the next one.

3. EXPERIMENTAL METHODOLOGY

To analyze HIRE-Lin, we designed a set of artificial datasets that corresponded to functions of different orders, topologies (concave, convex), and dimensions. Table 1 shows the mathematical formula of each function together with the mnemonic we will use in the paper. Some of these functions have already been used as benchmarks to test regression models (see for example [17]).

Table 1: Functions test bed

Mnemonic	Function
f_{asx}	$f(x) = \lvert\sin(10x)\rvert$
f_{px}	$f(x) = 1 + x + x^2 + x^3$
f_{s4x}	$f(x) = \sin(x) + \sin(2x) + \sin(3x) + \sin(4x)$
f_{scx}	$f(x) = \sin(x) * \cos(x)$
f_{x2}	$f(x) = x^2$
f_{xsx}	$f(x) = x * \sin(10x)_{f(x)>0}$
f_{rxy}	$f(x,y) = \sqrt{xy}$
f_{scxy}	$f(x,y) = \sin(xy) * \cos(xy)$
f_{sxy}	$f(x,y) = \sin(3xy)_{f(x,y)>0}$
f_{x2y}	$f(x,y) = x^2 y$

To build the datasets, we uniformly sampled 100 instances per dimension. That is, if the function was defined by a single attribute (one dimension) the resulting dataset contained 100 instances. For functions defined by two attributes, the resulting dataset contained 10000 instances.

To analyze the quality of the model evolved by HIRE-Lin, we considered the error and the model size. To estimate the error, each example of the dataset is checked against the ruleset. The first rule that matches provides an approximation which is compared with the value of the objective function. Thus, the error ε was estimated according to the following formula:

$$\varepsilon = \frac{\sum_{i=1}^{N_E} \left(f(\mathbf{e}_k) - \widehat{f(\mathbf{e}_k)}_{w_0,\ldots,w_\ell} \right)^2}{N_E} \quad (7)$$

where $f(\mathbf{e}_k)$ is the value of the objective function at point $\mathbf{e}_k$, $\widehat{f(\mathbf{e}_k)}_{w_0,\ldots,w_\ell}$ is the function approximation provided by HIRE-Lin, and N_E is the number of examples. The model size was computed as the number of rules evolved.

A 10-fold cross-validation procedure was used to estimate the error of the method. For each fold, the method was trained 10 times with different seeds and the error was averaged. When needed, statistical tests were applied to compare several approaches and test for significant differences among them. Our methodology followed the guidelines provided by Demsar [7] for multiple comparison tests. Briefly, we first tested the null hypothesis that the group of learners performed equivalently by means of a Friedman's test. If this hypothesis could be rejected, then we applied a post-hoc test to compare the learners to the best performer. Specifically, the Bonferroni-Dunn's test was used.

Our study consists of two parts. First, in Sect. 4 we analyze the behavior of HIRE-Lin. By means of a graphical analysis centered on a case study, we investigate the influence of the accuracy pressure parameter γ. We analyze the quality of the approximation and the number and types of rules evolved. We further extend this study to the whole function test bed to validate the influence of the accuracy pressure parameter. Then, in Sect. 5 we compare HIRE-Lin to three other well-known regression techniques, so that we can place our approach within some of the state-of-the-art methodologies.

4. ANALYSIS OF HIRE-LIN

This section analyzes the behavior of HIRE-Lin. First, we use function f_{xsx} as a case study (see Table 1 for the de-

Table 2: Parameters of HiRe-Lin

Parameter	Description	Value
ec	Evolutionary cycles	300
N	Population size	100
S	Tournament selection size	10
p_c	Crossover probability	0.5
p_m	Mutation probability	0.02
m_{off}	Mutation offset (fraction)	0.25
r_0	Covering parameter (fraction)	0.25
γ	Accuracy pressure	1

tails). Then, we numerically compare the results obtained with the remaining test bed. In both cases, HIRE-Lin was run with pressures γ={1,10,100,1000} and parameter settings provided in Table 2.

4.1 A Case Study

Figures 1 to 4 plot the results of HIRE-Lin in function f_{xsx} with γ={1,10,100,1000} respectively. Each figure shows, in the upper part, the training points used to train the algorithm (plotted with points) and the function approximation provided by HIRE-Lin (with solid line). In the lower part, the subspace covered by the each rule is plotted. Note that the ruleset is hierarchical and must be checked in order.

Figure 1 shows the result of the algorithm with $\gamma = 1$. See that the algorithm evolves only two rules. The first one is a large rule covering the domain [0,0.927]. The linear regressor obtained for this subspace cannot fit the shape of the objective function. The second rule applies to the domain (0.927,0.935], which allows for a better approximation because the subspace is small enough for a linear fitting. Note that the domain of the second rule R_2 is [0,0.935]. However, since the previous rule already covers the subspace [0,0.927], R_2 is only applied in the range (0.927,0.935]. The range codified by R_2 which is hidden by the previous rule is plotted in dotted lines, while the effective domain is plotted in solid line. The ruleset adds another rule R_d, the default rule, which always covers the range [0,1], although the effective range depends on the previous rules in the list. In this case, R_d would be applied only in the interval (0.935,1]. The reason why rules R_1 and R_2 have not expanded to cover completely the search space is that there are no training instances defined outside [0,0.935]. Thus, no rules are evolved for the region (0.935,1]. If a test example from this subspace is given, R_d would be applied with the value of the average objective function of training points.

Figure 2 plots the results for $\gamma = 10$. Note that the ruleset contains more rules than with $\gamma = 1$ and each rule covers a smaller subspace. This allows for better fitting than with $\gamma = 1$. By increasing γ, we change the relative weight of accuracy with respect to generalization (see equation 3). With $\gamma = 1$, generalization was so important that a very general rule with a rough approximation was obtained. With $\gamma = 10$, generalization is decreased in favor of accuracy. Thus, less general but more accurate rules are given. The effect of further increasing γ is plotted in Figures 3 and 4 which show even finer approximations to the objective function. Higher γ values also produce larger rulesets.

4.2 Comparison on Several Datasets

We extended the study on the influence of γ parameter to the remaining test bed described in Table 1. We

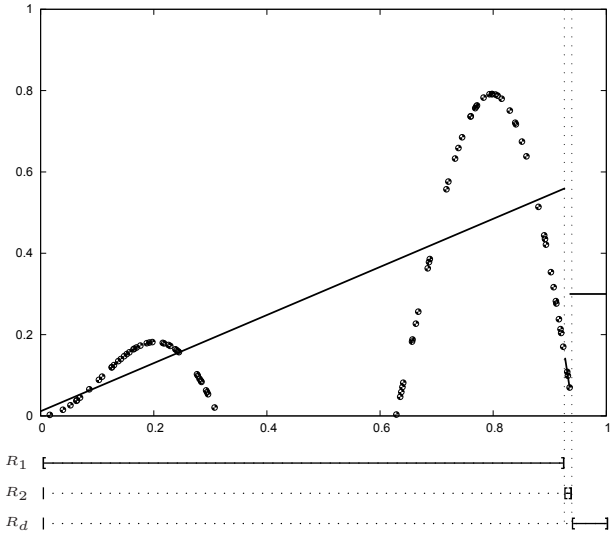

Figure 1: f_{xsx} approximation using $\gamma = 1$

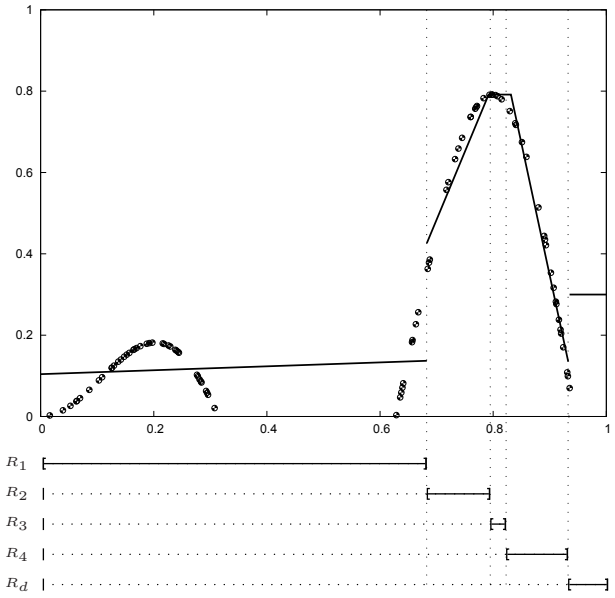

Figure 2: f_{xsx} approximation using $\gamma = 10$

studied the training error, the test error, and the number of rules obtained for the different accuracy pressures $\gamma = \{1, 10, 100, 1000\}$. The training error is a measure of the fit of the approximation to the training points, while the test error is an estimate of the generalization capability to unseen points. The number of rules is useful as a measure of interpretability of the final ruleset.

Tables 3 and 4 show the average and standard deviation of HIRE-Lin in the training dataset and test dataset respectively. As mentioned before, these values correspond to the error estimated by a 10-fold cross-validation procedure with 10 random seeds. The best approach giving the minimum average error is marked in bold. Regarding the training errors, larger values of γ yield smaller approximation errors. In

Table 3: Train error of HIRE-Lin with different accuracy pressures. Each cell gives the average and standard deviation of HIRE-Lin for the dataset in the row

DS	$\gamma = 1$	$\gamma = 10$	$\gamma = 100$	$\gamma = 1000$
f_{asx}	9.407e-02±3.60e-03	5.094e-02±2.72e-02	1.867e-03±1.46e-03	**1.653e-03±1.72e-03**
f_{px}	3.592e-03±1.85e-03	2.916e-03±1.63e-03	9.706e-04±1.07e-03	**1.574e-04±1.10e-03**
f_{s4x}	5.966e-02±5.07e-03	8.176e-03±2.92e-03	1.284e-03±3.11e-04	**9.511e-05±1.35e-04**
f_{scx}	2.368e-03±1.03e-04	2.610e-03±7.25e-04	**7.544e-04±8.58e-04**	1.343e-03±1.03e-03
f_{x2}	3.385e-03±1.35e-04	3.821e-03±2.62e-03	8.261e-04±1.67e-04	**6.935e-05±8.15e-06**
f_{xsx}	2.706e-02±3.00e-03	2.639e-03±2.07e-03	1.029e-04±4.12e-05	**7.108e-05±5.07e-04**
f_{rxy}	4.780e-03±1.25e-04	4.401e-03±4.19e-04	1.105e-03±1.24e-04	**1.068e-04±1.39e-05**
f_{scxy}	2.368e-03±1.03e-04	2.432e-03±9.34e-05	5.697e-04±8.79e-05	**5.090e-05±9.51e-06**
f_{sxy}	2.552e-02±5.58e-04	1.602e-02±2.69e-03	1.592e-03±3.01e-04	**1.717e-04±1.95e-04**
f_{x2y}	6.739e-03±7.77e-05	5.382e-03±6.37e-04	6.181e-04±2.27e-04	**4.391e-05±6.87e-06**

Table 4: Test error of HIRE-Lin with different accuracy pressures. Each cell gives the average and standard deviation of HIRE-Lin for the dataset in the row

DS	$\gamma = 1$	$\gamma = 10$	$\gamma = 100$	$\gamma = 1000$
f_{asx}	1.005e-01±2.54e-02	6.983e-02±3.82e-02	9.010e-03±2.47e-02	**6.932e-03±1.92e-02**
f_{px}	3.463e-03±1.77e-03	3.272e-03±1.88e-03	1.269e-03±7.28e-04	**1.561e-04±6.03e-04**
f_{s4x}	6.676e-02±2.65e-02	1.215e-02±8.18e-03	1.784e-03±1.12e-03	**6.120e-04±3.97e-03**
f_{scx}	2.554e-03±1.22e-03	2.899e-03±2.91e-03	**3.531e-04±1.81e-04**	4.363e-04±2.71e-03
f_{x2}	3.694e-03±1.85e-03	3.450e-03±1.70e-03	1.121e-03±6.05e-04	**2.293e-04±9.36e-04**
f_{xsx}	3.179e-02±1.17e-02	3.505e-03±3.89e-03	**8.770e-04±6.28e-03**	1.542e-03±6.13e-03
f_{rxy}	4.805e-03±1.15e-03	4.618e-03±1.19e-03	1.413e-03±9.31e-04	**1.569e-04±7.09e-05**
f_{scxy}	2.483e-03±4.55e-04	2.473e-03±4.66e-04	6.596e-04±2.42e-04	**7.590e-05±5.94e-05**
f_{sxy}	2.558e-02±3.26e-03	1.765e-02±4.23e-03	**2.325e-03±1.73e-03**	2.860e-03±5.04e-03
f_{x2y}	6.798e-03±1.21e-03	5.438e-03±1.19e-03	6.698e-04±2.19e-04	**8.222e-05±3.98e-05**

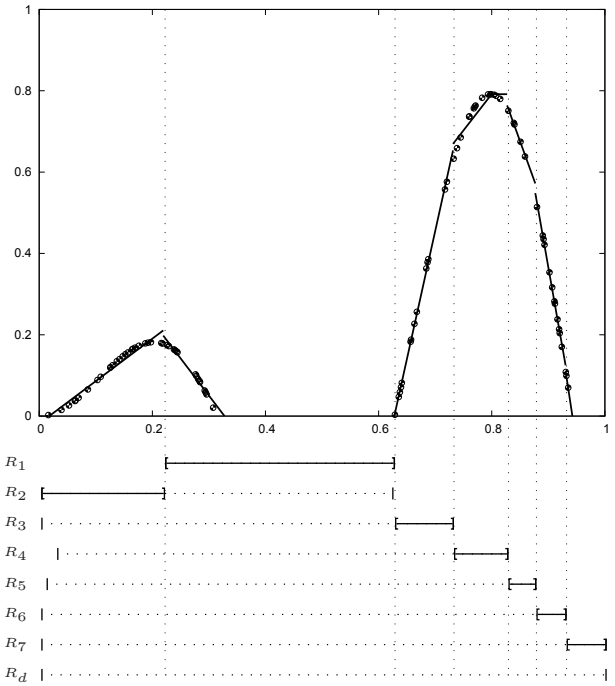

Figure 3: f_{xsx} approximation using $\gamma = 100$

all functions, except for f_{scx}, the smallest error is obtained for $\gamma = 1000$. Larger values of γ also tend to give smaller test errors. However, in functions f_{xsx} and f_{sxy} the largest accuracy pressure ($\gamma = 1000$) gives the smallest training error but this does not correspond to the smallest test error. This indicates that overfitting is occurring in these cases.

Table 5 shows the number of rules of the final ruleset. For a given problem, the number of rules obtained increases with larger values of γ, as it was already observed in the case study. For $\gamma = 1$ the average ruleset consists of a single rule, two at maximum (the default rule is not counted). This value is too extreme to get a good approximation. Larger pressures provide larger rulesets. Values of γ ranging from 100 to 1000 provide fairly good approximations. By adjusting parameter γ we can balance the compromise between accurate approximation and interpretability (smaller rulesets are usually more interpretable). Note also that the most complex problems, such as those with two attributes, require larger rulesets. f_{sxy} is the problem that requires the highest number of rules.

We statistically compared the accuracy and size of the models evolved with the different configurations. In Figure 5, each system is placed in the axes according to its average rank regarding the approximation error (x-axis) and its average rank regarding population size (y-axis). The vertical dashed lines delimit the region of the comparison space where the learners perform equivalently to the learner that presented the best performance according to a Bonferroni-Dunn test at a significance level of 0.10. Similarly, horizontal lines determine the region of equivalence to the method that created the smallest models. Note that HIRE-Lin with $\gamma = 100$ and $\gamma = 1000$ evolved the most accurate models of the comparison. On the other hand, HIRE-Lin with $\gamma = 1$ and $\gamma = 10$ built the most reduced rulesets, which went in detriment of the test accuracy.

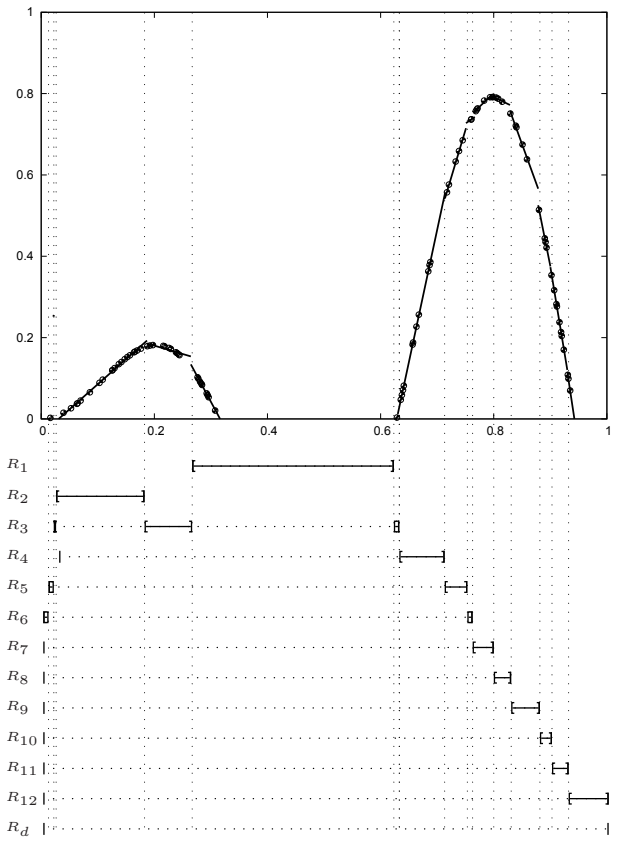

Figure 4: f_{xsx} approximation using $\gamma = 1000$

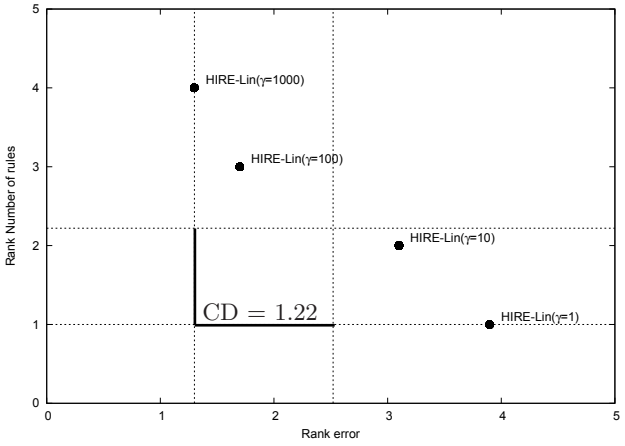

Figure 5: Average rank of HIRE-Lin with accuracy pressures {1,10,100,1000}. The x-axis plots the rank of HIRE-Lin with respect to the test error (the approach with the smallest error has the smallest rank). The y-axis plots the rank of each approach with respect to the smallest ruleset. The critical distance (CD) delimits the region of equivalence with the best learner in each objective. It is computed according to a Bonferroni-Dunn test at $\alpha = 0.10$.

Table 5: Number of rules (average and standard deviation) obtained by HIRE-Lin with different accuracy pressures

DS	$\gamma = 1$	$\gamma = 10$	$\gamma = 100$	$\gamma = 1000$
f_{asx}	1.25±0.44	4.13±1.55	11.39±0.90	16.88±1.35
f_{px}	1.05±0.22	1.86±0.35	2.42±0.50	4.47±0.50
f_{s4x}	1.85±0.36	2.05±0.22	3.12±0.33	5.21±0.41
f_{scx}	1.00±0.00	1.32±0.47	2.21±0.41	3.78±0.48
f_{x2}	1.00±0.00	1.58±0.50	3.04±0.20	5.75±0.61
f_{xsx}	1.96±0.20	3.71±0.56	6.23±0.78	11.23±1.21
f_{rxy}	1.01±0.10	1.68±0.67	8.96±1.22	25.07±2.97
f_{scxy}	1.01±0.10	1.34±0.52	9.68±1.32	29.81±2.97
f_{sxy}	1.01±0.10	4.14±1.34	15.36±3.74	43.07±3.79
f_{x2y}	1.00±0.00	2.89±0.60	10.88±2.17	39.55±4.33

Table 6: Comparison of (a) Linear Least Mean Squares (LMS), (b) Fuzzy Wang-Mendel (WM), (c) GAP, and (d) HIRE-Lin with $\gamma = 1000$ on a collection of eleven artificial problems.

DS	LMS	WM	GAP	HIRE-Lin
f_{asx}	0.10019	0.10056	0.15491	0.00693
f_{px}	0.00443	0.00022	0.00300	0.00016
f_{s4x}	0.07331	0.05779	0.00273	0.00061
f_{scx}	0.00359	0.00123	0.00075	0.00044
f_{x2}	0.00561	0.00001	0.00101	0.00023
f_{xsx}	0.04227	0.04028	0.05650	0.00154
f_{rxy}	0.00488	0.00000	0.00363	0.00016
f_{scxy}	0.00300	0.00125	0.00047	0.00008
f_{sxy}	0.02891	0.02690	0.00219	0.00286
f_{x2y}	0.00898	0.00003	0.00103	0.00008
rank	*3.73*	*2.27*	*2.64*	*1.36*

5. COMPARISON WITH OTHER REGRESSION TECHNIQUES

So far, we have analyzed the impact of the accuracy pressure in the size and accuracy of the models evolved by HIRE-Lin. In this section, we compare the behavior of HIRE-Lin with three regression techniques: Linear LMS [21], Fuzzy Wang-Mendel [26], and GAP [22]. Linear LMS uses the least mean square algorithm to create a linear approximation of the input data. Fuzzy Wang Mendel builds a set of Mandani fuzzy rules [6] that minimize the error with the covered instances. GAP is a method based on genetic algorithms and genetic programming that evolves a function represented in a tree. All these methods were run using KEEL [2]. We used the default configuration recommended in the software [2] to configure each method. We configured HIRE-Lin with the parameters specified in Table 2; besides, we set γ=1000, since, as shown in the last section, it yields accurate models of moderate size.

Table 6 provides the test error obtained for each problem and learner. The multi-comparison Friedman's test [10, 11] permitted us to reject the null hypothesis that all learners performed the same on average with $p = 8.22 \cdot 10^{-4}$. To analyze which learners performed significantly differently from HIRE-Lin, we used the post-hoc Bonferroni-Dunn test [9] at $\alpha = 0.10$. Figure 6 ranks the four learners and connects those that perform equivalently according to the Bonferroni-

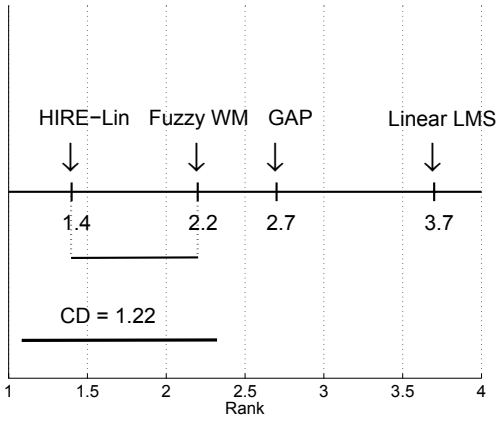

Figure 6: Comparison of the test performance of HIRE-Lin with the other methods by means of a Bonferroni-Dunn Test at $\alpha = 0.10$. Groups of classifiers that are not significantly different to the best ranked method are connected.

Dunn procedure. HIRE-Lin is the best ranked method, and outperforms the results obtained by Linear LMS and GAP. Linear LMS uses a least mean square approach to build a linear function that approximates the output. Note that HIRE-Lin uses the same approach to evolve piece-wise linear approximations of the function drawn by the input instances. Therefore, the partition of the feature space promoted by the genetic algorithm allows HIRE-Lin to achieve much better approximations. GAP is an evolutionary approach that evolves a function coded as a tree, which permits to represent more complex, non-linear expressions. Notice that HIRE-Lin significantly outperforms this technique on the collection of tested problems by evolving simple linear functions to approximate the input. It is worth highlighting that both LMS and GAP use a global approximation, while HIRE-Lin evolves an arbitrary number of rules that locally approximate the objective function. The number of rules evolved depends on both the non-linearity of the objective function and the accuracy pressure γ.

As the Bonferroni-Dunn test is said to be quite conservative, we also performed pairwise comparisons among learners by means of the non-parametric Wilcoxon signed-ranks test [27], assuming the risk of increasing the error of rejecting the null hypothesis when it is actually true. Table 7 provides the approximate p-values. The symbols $\oplus$ and $\ominus$ indicate that the method in the row significantly improves/degrades the performance obtained by the method in the column at a significance level of 0.05. The symbols $+/-$ denote a non-significant improvement/degradation. The pairwise analysis confirms the conclusions extracted from the Bonferroni-Dunn test; moreover, it also detects that HIRE-Lin outperforms Fuzzy Wang Mendel. Therefore, the pairwise analysis supports the conclusion that HIRE-Lin outperforms all the other methods in the comparison.

Table 7: Pair-wise comparison of the test performance achieved by HIRE-Lin with the accuracy obtained with Linear LMS (LMS), Fuzzy Wang Mendel (WM), and GAP.

	LMS	WM	GAP	HIRE-Lin
LMS		0.004	0.182	0.003
WM	$\oplus$		0.657	0.026
GAP	$+$	$-$		0.008
HIRE-Lin	$\oplus$	$\oplus$	$\oplus$	

6. CONCLUSIONS

In this paper, we proposed a regression algorithm that evolves a hierarchical set of rules performing piece-wise linear approximations. The algorithm is based on an iterative rule learning approach, which consists in evolving a single rule in each iteration. Each rule delimits a subspace where an optimal linear regressor is constructed. The search space of the algorithm is progressively reduced as rules are evolved.

The genetic algorithm is applied to search for the largest hyperrectangular subspace where the optimal linear regressor, trained for the data points enclosed in that subspace, accurately approximates the objective function.

The balance between generalization and fit of the model can be adjusted in the fitness function. We used an aggregation approach, where the relative influence of these objectives could be modified by the so-called accuracy pressure. As the accuracy pressure was increased, the model obtained finer approximations at the cost of evolving larger rulesets, compromising interpretability of the final ruleset and even leading to overfitting. We acknowledge that the search could be formulated as a multiobjective fitness function based on Pareto approaches. A key advantage of such an approach is to let the user to choose among alternative compromises between generalization and model fit. A possible aid to avoid overfitting is to use an additional validation set containing points different from those in the training dataset to evaluate whether the approximation is generalizing to these unknown points. In this sense, a Pareto-based multiobjective approach would be more flexible, because it would allow the user to choose the solution with less overfitting.

Our approach evolves a set of hierarchical piece-wise linear regressors. Similarly, non-hierarchical piece-wise linear regressors are evolved by XCSF, which belongs to the category of Michigan approaches. XCSF searches simultaneously for a set of overlapping piece-wise regressors which together cover the search space. A key point of our approach is that rulesets tend to be smaller than those usually obtained by Michigan approaches. However, this hypothesis must be further investigated. As a future work we aim at comparing the rulesets and model fitting of both approaches. Also XCSF has been trained to evolve other types of regressors such as neural and polynomial regressors [16]. This feature could also be included easily in HIRE-Lin.

The architecture was highly competitive with respect to other regression techniques, such as LMS, Fuzzy Wang Mendel and GAP. In fact, it is not surprising that HIRE-Lin surpasses the behavior of the linear regressor LMS, since our approach is a local approach and LMS a global approach training a single linear regression for the whole search space. HIRE-Lin also improves GAP, a global method evolving a regression function by means of genetic algorithms and genetic programming. Other types of regressors such as locally weighted regression [18] could be more advantegeous

than global methods and compare similarly to HIRE-Lin. Although this particular study remains for further work, we already demonstrated that HIRE-Lin is competitive with respect to Fuzzy Wang Mendel, which is a local approach.

Acknowledgements

The authors would like to thank the *Ministerio de Educación y Ciencia* for its support under project TIN2005-08386-C05-04, and *Generalitat de Catalunya* for its support under grants 2005FI-00252 and 2005SGR-00302.

7. REFERENCES

[1] J. S. Aguilar-Ruiz, J. C. R. Santos, and M. Toro. Evolutionary learning of hierarchical decision rules. *IEEE Transactions on Systems, Man, and Cybernetics, Part B*, 33(2):324–331, 2003.

[2] J. Alcalá-Fdez, L. Sánchez, S. García, M. J. del Jesus, S. Ventura, J. M. Garrell, J. Otero, C. Romero, J. Bacardit, V. M. Rivas, J. C. Fernández, and F. Herrera. Keel: A software tool to assess evolutionary algorithms to data mining problems. *Soft Computing*, forthcoming.

[3] J. Bacardit and J. Garrell. Bloat control and generalization pressure using the minimum description length principle for a Pittsburgh approach Learning Classifier System. In *Learning Classifier Systems, Revised Selected Papers of the International Workshop on Learning Classifier Systems 2003-2005*, volume 4399 of *Lecture Notes in Computer Science*, pages 59–79. Springer, 2003.

[4] B. Carse, T. Fogarty, and A. Munro. Evolving fuzzy rule based controllers using genetic algorithms. *Fuzzy Sets and Systems*, 80:273–293, 1996.

[5] C. C. Coello, D. V. Veldhuizen, and G. Lamont. *Evolutionary Algorithms for Solving Multi-Objective Problems*. Kluwer Academic Publishers, 2002.

[6] O. Cordón, F. Herrera, F. Hoffmann, and L. Magdalena. *Genetic fuzzy systems: Evolutionary tuning and learning of fuzzy knowledge bases*, volume 19 of *Advances in Fuzzy Systems—Aplications and Theory*. World Scientific, 2001.

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

[8] J. Drugowitsch and A. Barry. A Formal Framework and Extensions for Function Approximation in Learning Classifier Systems. *Machine Learning*, 70(1):45–88, 2008.

[9] O. Dunn. Multiple comparisons among means. *Journal of the American Statistical Association*, 56:52–64, 1961.

[10] M. Friedman. The use of ranks to avoid the assumption of normality implicit in the analysis of variance. *Journal of the American Statistical Association*, 32:675–701, 1937.

[11] M. Friedman. A comparison of alternative tests of significance for the problem of m rankings. *Annals of Mathematical Statistics*, 11:86–92, 1940.

[12] S. Glantz and B. Slinker. *Primer of Applied Regression & Analysis of Variance*. McGraw-Hill, 2001.

[13] A. González and F. Herrera. Multi-stage Genetic Fuzzy Systems Based on the Iterative Rule Learning Approach. *Mathware & Soft Computing*, 4:233–249, 1997.

[14] J. H. Holland. Adaptation. In R. Rosen and F. Snell, editors, *Progress in Theoretical Biology*, volume 4, pages 263–293. Academic Press, 1976.

[15] P. Lanzi, D. Loiacono, S. Wilson, and D. Goldberg. Generalization in the XCSF Classifier System: Analysis, Improvement, and Extension. *Evolutionary Computation Journal*, 2007.

[16] P. L. Lanzi, D. Loiacono, S. W. Wilson, and D. E. Goldberg. Extending XCSF beyond linear approximation. In *GECCO '05*, pages 1827–1834. ACM, 2005.

[17] P. L. Lanzi, D. Loiacono, S. W. Wilson, and D. E. Goldberg. Prediction update algorithms for XCSF: RLS, Kalman filter, and gain adaptation. In *GECCO '06*, pages 1505–1512. ACM, 2006.

[18] T. M. Mitchell. *Machine Learning*. McGraw-Hill, 1997.

[19] D. Montgomery and G. Runger. *Applied Statistics and Probability for Engineers*. John Wiley & Sons, 2003.

[20] R. L. Rivest. Learning decision lists. *Machine Learning*, 2(3):229–246, 1987.

[21] J. Rustagi. *Optimization Techniques in Statistics*. Academic Press, 1994.

[22] L. Sánchez, I. Couso, and J. A. Corrales. Combining GP operators with SA search to evolve fuzzy rule based classifiers. *Information Sciences*, 136(1–4):175–191, 2001.

[23] S. F. Smith. Flexible Learning of Problem Solving Heuristics through Adaptive Search. In *Proc. of the 8th International Joint Conference on Artificial Intelligence*, pages 422–425, 1983.

[24] W. M. Spears and D. F. Gordon. Adaptative Strategy Selection for Concept Learning. In *Proc. of the First International Workshop on Multistrategy Learning (MSL-91)*, pages 231–246. Harpers Ferry, MD, 1991.

[25] G. Venturini. SIA: A Supervised Inductive Algorithm with Genetic Search for Learning Attribute Based Concepts. In *Proc. European Conf. Machine Learning*, pages 280–296, 1993.

[26] L. Wang and J. Mendel. Generating fuzzy rules by learning from examples. *IEEE Transactions on Systems, Man and Cybernetics*, 22(6):1414–1427, 1992.

[27] F. Wilcoxon. Individual comparisons by ranking methods. *Biometrics*, 1:80–83, 1945.

[28] S. Wilson. Classifiers that approximate functions. *Journal of Natural Computing*, 1(2–3):211–234, 2002.

[29] S. W. Wilson. Classifier Fitness Based on Accuracy. *Evolutionary Computation*, 3(2):149–175, 1995.

[30] S. W. Wilson. Generalization in the XCS Classifier System. In J. Koza, W. Banzhaf, K. Chellapilla, K. Deb, M. Dorigo, D. Fogel, M. Garzon, D. Goldberg, H. Iba, and R. Riolo, editors, *Genetic Programming: Proc. of the Third Annual Conference*, pages 665–674. San Francisco, CA: Morgan Kaufmann, 1998.

[31] I. H. Witten and E. Frank. *Data Mining. Practical Machine Learning Tools and Techniques with Java Implementations*. Morgan Kaufmann Publishers, 2000.