Improving Generalization in the XCSF Classifier System Using Linear Least-Squares

Daniele Loiacono
Artificial Intelligence and Robotics Laboratory
Politecnico di Milano
P.za L. da Vinci 32, I-20133, Milano, Italy
loiacono@elet.polimi.it

Pier Luca Lanzi[*]
Artificial Intelligence and Robotics Laboratory
Politecnico di Milano
P.za L. da Vinci 32, I-20133, Milano, Italy
lanzi@elet.polimi.it

ABSTRACT

XCSF is an extension of XCS in which classifier prediction is computed as a linear combination of classifier inputs and a weight vector associated to each classifier. XCSF can adjust the weight vector of classifiers to evolve accurate piecewise linear approximations of functions. The Widrow-Hoff rule, used to update the weight vectors, prevents (when some conditions hold) XCSF from exploiting the expected piecewise linear approximation. In this paper we replace the Widrow-Hoff rule with linear least-squares and we show that with this improvement XCSF can fully exploit its generalization capabilities.

Categories and Subject Descriptors

I.2.6 [**Artificial Intelligence**]: Learning—*Learning Classifier Systems*

General Terms

Theory, Experimentation, Algorithms

Keywords

learning classifier systems, XCSF, least-squares, generalization, function approximation

1. INTRODUCTION

XCSF is a model of learning classifier system, introduced by Wilson [10], that extends the typical concept of classifiers through the introduction of a computable classifier prediction. XCSF has been successfully applied to some function approximation problems, evolving accurate solutions [10]. However, a recent work [7] shows that XCSF may be unable to fully exploit its expected generalization capabilities when some condition on the inputs domain holds; in [7] the reason of this unexpected behavior has been addressed to the Widrow-Hoff rule used in XCSF for updating the weights of classifiers. In this work we introduce a version (XCSFls) of XCSF by replacing the Widrow-Hoff rule, in the update procedure, with linear least-squares. The experimental results reported show that XCSFls can fully exploit the expected linear approximation. Moreover results show that XCSFls can evolve solutions that are significantly more accurate and more compact compared to the ones evolved by XCSF.

[*]Advisor of the master thesis

The paper is organized as follows. We begin in Section 2 with a description of XCSF [10] and then in Section 3 we introduce XCSFls. In Section 4 we describe how experiments have been performed to compare XCSF and XCSFls. Then we discuss the the experimental results in Section 5. Finally in Section 6 we summarize the work and we discuss the results obtained.

2. THE XCSF CLASSIFIER SYSTEM

XCSF differs from XCS in three respects: (i) classifiers conditions are extended for numerical inputs, as done in XCSI [9]; (ii) classifiers are extended with a vector of weights w, that are used to compute classifier's prediction; finally, (iii) the original update of classifier prediction must be modified so that the weights are updated instead of the classifier prediction. These three modifications result in a version of XCS, XCSF [10], that maps numerical inputs into actions with an associated calculated prediction.

Classifiers. In XCSF, classifiers consist of a condition, an action, and four main parameters. The condition specifies which input states the classifier matches; as in XCSI [9], it is represented by a concatenation of interval predicates, $int_i = (l_i, u_i)$, where l_i ("lower") and u_i ("upper") are integers, though they might be also real. The action specifies the action for which the payoff is predicted; when XCSF is used as function approximator (as in this paper) there is only one dummy action which has no actual effect. The four parameters are: the weight vector w, used to compute the classifier prediction as a function of the current input; the prediction error ε, that estimates the error affecting classifier prediction; the fitness F that estimates the accuracy of the classifier prediction; the numerosity *num*, a counter used to represent different copies of the same classifier.

Performance Component. XCSF works as XCS. At each time step t, XCSF builds a *match set* [M] containing the classifiers in the population [P] whose condition matches the current sensory input s_t. As in XCS, in XCSF the *system prediction* is computed by the fitness-weighted average of all matching classifiers. However, in contrast with XCS, in XCSF classifier prediction is computed as a function of the current state s_t and the classifier vector weight w. Accordingly, in XCSF system prediction is a function of the current state s_t, defined as:

$$P(s_t) = \frac{\sum_{cl\in[M]} cl.p(s_t) \times cl.F}{\sum_{cl\in[M]} cl.F} \tag{1}$$

where cl is a classifier, $cl.F$ is the fitness of cl; $cl.p(s_t) = cl.w_0 \times x_0 + \sum_{i>0} cl.w_i \times s_t(i)$ is the prediction of cl computed in the state s_t (where $cl.w_i$ is the weight w_i of cl associated to the input i and x_0 is a constant input). Next, XCSF performs the dummy action with no actual effect and a reward P is returned to the system.

Reinforcement Component. XCSF uses the incoming reward P to update the parameters of classifiers in the match set [M]. The weight vector w of the classifier in the match set [M] is updated using the Widrow-Hoff rule. For each classifier $cl \in [M]$, each weight $cl.w_i$ is adjusted by a quantity Δw_i computed as:

$$\Delta w_i = \frac{\eta}{|\vec{x}_{t-1}|^2}(P - cl.p(s_{t-1}))x_{t-1}(i) \tag{2}$$

where η is the correction rate and $\vec{x}_{t-1}$ is defined as the input state vector s_{t-1} augmented by a constant x_0 (i.e. $\vec{x}_{t-1} = \langle x_0, s_{t-1}(1), s_{t-1}(2), \ldots, s_{t-1}(n)\rangle$) and $|\vec{x}_{t-1}|^2$ is the norm of vector $\vec{x}_{t-1}$ for further details refer to [10]. The values Δw_i are used to update the weights of classifier cl as $cl.w_i \leftarrow cl.w_i + \Delta w_i$. Finally, prediction error and classifier fitness is updated as in XCS.

Discovery Component. The genetic algorithm in XCSF works as in XCSI [9]. On a regular basis depending on the parameter θ_{ga}, the genetic algorithm is applied to classifiers in [M]. It selects two classifiers with probability *proportional to their fitness*, copies them, and with probability χ performs crossover on the copies; then, with probability μ it mutates each allele. Crossover and mutation work as in XCSI [9, 10]. The resulting offspring are inserted into the population and two classifiers are deleted to keep the population size constant.

3. LINEAR LEAST-SQUARES FOR XCSF

The Widrow-Hoff update used in XCSF can converge very slowly when some conditions on the distribution of the classifiers inputs hold. In [7] is analyzed in detail the slow convergence of the Widrow-Hoff rule, showing how it may prevent XCSF from exploiting its generalization capabilities. Overall the experimental results reported in [7] show that XCSF may evolve a piecewise constant approximation of the target function, instead of the expected linear approximation. We refer the reader to [7] for details and discussion about this analysis. In the following we describe as Widrow-Hoff update can be replaced using linear least-squares in order to eliminate these drawbacks.

Linear least-squares have been already used in reinforcement learning [2, 1] as an alternative to the Widrow-Hoff rule. Linear least-squares are in fact more efficient than Widrow-Hoff since they need fewer data samples to converge [2] and their convergence speed is not influenced by the input distribution [4]. The Widrow-Hoff update used in XCSF [10] can be viewed as a gradient descent of the following error function:

$$\xi(\vec{w}) = \tfrac{1}{2}e^2(t) \quad \text{where } e(t) = f(x_t) - cl.p(x_t) \tag{3}$$

in linear least-squares this error function is replaced by,

$$\xi(\vec{w}) = \tfrac{1}{2}\sum_{i=1}^{t} e^2(i) \quad \text{where } e(i) = f(x_i) - cl.p(x_i). \tag{4}$$

Let $X_t = [\vec{x_1}, \vec{x_2}, \ldots, \vec{x_t}]^T$ be a vector of inputs until time t, and let $\vec{f}(t) = [f(\vec{x_1}), f(\vec{x_2}), \ldots, f(\vec{x_t})]$ be the vector of all the desired outputs. The weight vector $\vec{w}$ that minimizes the error function in Equation 4, is the solution of the following equation:

$$(X_t^T X_t)\vec{w} = X_t^T \vec{f}(t). \tag{5}$$

The solution of the previous equation can be calculated in a robust way by applying the Singular Value Decomposition to $(X_t^T X_t)$ [8]. The update procedure described so far requires to store all the inputs and the desired outputs, but we're interested in an incremental update procedure. For this reason we implement the linear least-squares by keeping, for each classifier, a vector X_t^n of the last n visited inputs and a vector Y_t^n of the corresponding desired outputs. At each time step t, weight vector $\vec{w_t^*}$ is calculated as the solution of

$$(X_t^n)^T X_t^n \vec{w_t^*} = (X_t^n)^T Y_t^n, \tag{6}$$

and then the solution $\vec{w_t^*}$ is used to update the current weight vector as follows:

$$\vec{w_t} = (1-\eta)\vec{w_{t-1}} + \eta\vec{w_t^*}. \tag{7}$$

Under this formulation it's possible to use a small window of data (i.e. a small value of n), by choosing an adequate value of the learning rate η.

We dub XCSFls the version of XCSF obtained replacing the Widrow-Hoff rule with the linear least-squares. In this paper all the experiments performed with XCSFls have been done using the last $n = 50$ visited input and the corresponding desired outputs.

4. DESIGN OF EXPERIMENTS

In each experiment reported in this paper XCSF has to learn to approximate a target function $f(x)$; each experiment consists of a number of problems that XCSF must solve For each problem, an example $\langle x, f(x)\rangle$ of the target function $f(x)$ is randomly selected; x is input to XCSF whom computes the approximated value $\hat{f}(x)$ as the expected payoff of the only available dummy action action; the action is virtually performed (the action has no actual effect), and XCSF receives a reward equal to $f(x)$. XCSF learns to approximate the target function $f(x)$ by evolving a mapping from the inputs to the payoff of the only available action. Each problem is either a *learning* problem or a *test* problem. In *learning* problems, the genetic algorithm is enabled while it is turned off during *test* problems. The covering operator is always enabled, but operates only if needed. Learning problems and test problems alternate.

XCSF performance is measured as the accuracy of the evolved approximation $\hat{f}(x)$ with respect to the target function $f(x)$. To evaluate the evolved approximation $\hat{f}(x)$ we measure the *mean absolute error* (MAE) defined as:

$$MAE = \frac{1}{n}\sum_x |f(x) - \hat{f}(x)|$$

where n is the number of points for which $f(x)$ is defined. To evaluate instead the generalization capabilities of the system we measure also the average size of population evolved (calculated as the number of different classifiers) and the average *generality* of classifiers in the solution evolved (where generality is calculated as the size of the interval covered from its condition). All the statistics reported in this paper are averaged over 50 experiments. All the experiments reported have been conducted on `xcslib` [6].

5. EXPERIMENTAL RESULTS

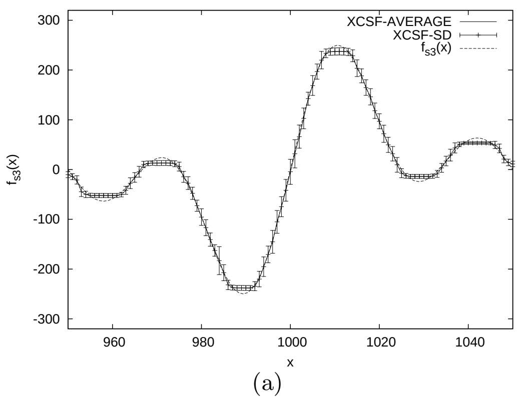
We compare XCSF with XCSFls on the following functions:

$$f_{s3}(x) = 100\left(\sin\left(\frac{2\pi x}{100}\right)+\sin\left(\frac{4\pi x}{100}\right)+\sin\left(\frac{6\pi x}{100}\right)\right), \quad (8)$$

$$f_{s4}(x) = 100\left(\sin\left(\frac{2\pi x}{100}\right)+\sin\left(\frac{4\pi x}{100}\right)+ + \sin\left(\frac{6\pi x}{100}\right)+\sin\left(\frac{8\pi x}{100}\right)\right), \quad (9)$$

$$f_{abs}(x) = 100\left|\sin\left(\frac{2\pi x}{100}\right)+\left|\cos\left(\frac{2\pi x}{100}\right)\right|\right|. \quad (10)$$

Equation 8 and Equation 9 are Koza's Sinus Three and Sinus Four [5] adapted to the integer domain $D = \{950 \le x \le 1050, x \in \mathbb{N}\}$, following what done by [10] for the sine function; Equation 10 is taken from [11] and adapted for integers.
In all the experiments we use the following parameter setting: $N = 400$, $\beta = 0.2$; $\alpha = 0.1$; $\nu = 5$; $\chi = 0.8$, $\mu = 0.04$, $\theta_{nma} = 1$, $\theta_{del} = 50$; $\theta_{GA} = 50$; $\delta = 0.1$; GA-subsumption is on with $\theta_{sub} = 50$; while action-set subsumption is off; the parameters for integer conditions are $m_0 = 20$, $r_0 = 10$; the parameters for the piecewise linear approximations are $\eta = 0.2$ and $x_0 = 1000$ [10]; each experiment consists of 50000 learning problems. For each experiment we use a different value of error threshold ε_0.
In the first set of experiments we compare the performance of XCSF and of XCSFls on the function $f_{s3}(x)$ when ϵ_0=10. Both the systems evolve solutions with an average error below the target threshold (Table 1); but when the error threshold ϵ_0 is lowered to 5 we can observe that XCSF evolve solutions with an average error higher than the ϵ_0, while XCSFls is able to evolve accurate solutions with respect to the target threshold. It's even more interesting to discuss the others distinctive features of the solutions evolved by the two systems: the average size of population and the average generality of classifiers. Experimental results in Table 1 show that XCSFls evolves solutions that are definitely more compact of the ones evolved by XCSF (e.g. in the case of function $f_{s3}(x)$ we can see that populations evolved by XCSF have size double or even more than the ones evolved by XCSFls). Moreover the generality of classifiers evolved by XCSFls is sensibly greater than the corresponding one evolved by XCSF.
In Figure 1 we report the approximation evolved by XCSF and by XCSFls in the case of $f_{s3}(x)$ with $\epsilon_0 = 10$. In Figure 1a we observe that approximation evolved XCSF have an higher variance compared to the one evolved by XCSFls reported in Figure 1b. The difference between the two versions of the system is even more evident in Figure 2 that shows the classifiers evolved in the best solution (i.e. the solution with the lowest average error over all the 50 runs) by XCSF (Figure 2a) and by XCSFls (Figure 2b). Classifiers evolved by XCSF cover only small intervals over the input range and overall the approximation is definitely piecewise constant. XCSFls instead fully exploits the linear approximation, evolving classifiers that cover large parts of the function domain.



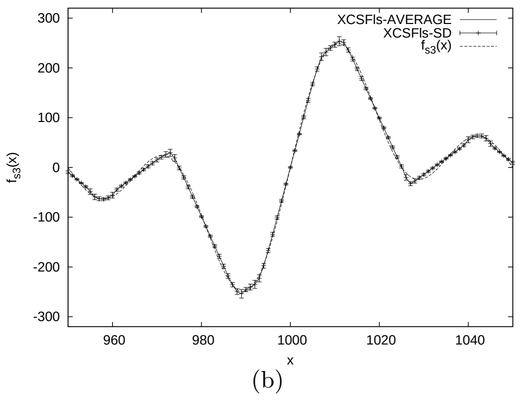


Figure 1: Comparison of XCSF and XCSFls applied to $f_{s3}(x)$ with $N = 400$ and $\epsilon_0 = 10$: (a) XCSF approximation; (b) XCSFls approximation;

The experiments on the function $f_{s4}(x)$ and on the function $f_{abs}(x)$ (not reported here for brevity) shows exactly the same behavior discussed for the function $f_{s3}(x)$; however all the experimental results are summarized in Table 1.
Statistical Analysis. We apply a *one-way* analysis of variance or ANOVA [3] to test whether the differences in approximation accuracy and generalization capabilities observed for the two versions of XCSF (Table 1) are statistically significant. Then, we apply a series of (typical) post-hoc procedures (namely Tukey, Scheffé, and Bonferroni) to analyze the differences between XCSF and XCSFls. First we analyze the mean absolute error ($\overline{MAE}$). The ANOVA test performed on the data of mean absolute error shows that the two versions of XCSF perform significantly different, with a confidence level of the 99.99%. Also the three subsequent *post-hoc* procedures (Scheffé, LSD, and Tukey) show that the performance of XCSF and XCSFls is significantly different. Then we apply the same analysis to the data of the population size and to the data of classifiers' generality obtaining the same results: the difference between XCSF and XCSFls both in terms of size of the evolved solutions both in terms of generality of the classifiers in the final populations is statistically significant, that is, least-squares approach significantly improves the generalization capability of XCSF.

$f(x)$	ϵ_0	XCSF			XCSFls		
		$MAE \pm \sigma$	$\|[P]\| \pm \sigma$	G([P])$\pm\sigma$	$MAE \pm \sigma$	$\|[P]\| \pm \sigma$	G([P])$\pm\sigma$
$f_{s3}(x)$	10	8.6 ± 1.1	47.2 ± 3.4	4.1 ± 3.5	6.0 ± 0.3	21.8 ± 3.8	12.6 ± 4.3
$f_{s3}(x)$	5	8.0 ± 1.1	55.8 ± 5.0	3.3 ± 2.9	3.2 ± 0.2	24.5 ± 3.5	8.5 ± 3.5
$f_{s4}(x)$	10	11.2 ± 1.6	49.1 ± 4.3	3.9 ± 3.1	6.6 ± 0.4	24.0 ± 3.3	9.4 ± 3.4
$f_{s4}(x)$	5	11.7 ± 1.7	48.9 ± 4.3	4.0 ± 3.1	3.6 ± 1.4	29.7 ± 4.2	6.5 ± 3.0
$f_{abs}(x)$	5	4.1 ± 0.5	45.1 ± 4.2	4.5 ± 4.1	2.4 ± 0.2	22.1 ± 3.6	15.0 ± 2.5

Table 1: Generalization with XCSF and XCSFls: $f(x)$ is the target function; ϵ_0 is the error threshold; $\overline{MAE}\pm\sigma$ is the value of the average mean absolute error with the standard deviation; $|[P]| \pm \sigma$ is the average size of the evolved solution with the standard deviation; $G([P]) \pm \sigma$ is the average generality of classifiers in the evolved solution with the standard deviation. Statistics are averages over 50 runs.

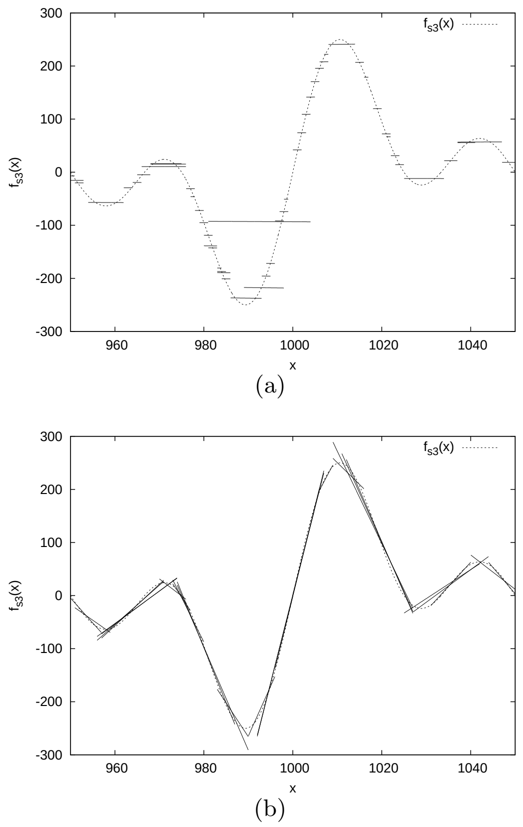


Figure 2: XCSF and XCSFls applied to $f_{s3}(x)$: (a) best population evolved by XCSF and (b) best population evolved by XCSFls. Segments identify the approximations provided by classifiers.

6. CONCLUSIONS

The Widrow-Hoff update used in XCSF may interfer with the generalization capabilities of the system and may prevent the reproduction of general and accurate classifiers [7]. In this work we introduced XCSFls a new version of XCSF in which the Widrow-Hoff update is replaced by a linear least-squares based update. Here we presented a simple experiment showing that while XCSF evolves an unexpected piecewise constant approximation, XCSFls can effectively exploit linear approximation, evolving also a more compact solution. Then we reported the statistical analysis performed on a larger number of experiments: our analysis showed that (i) XCSFls evolves solution statistically more accurate than the ones evolved by XCSF (ii) linear least-square approach lead to an higher and more effective generalization; i.e. the solutions evolved by XCSFls are more compact than the ones evolved by XCSF and are formed by classifiers with an higher average generality.
The least-squares approach discussed in this work requires more computational resources of the original Widrow-Hoff approach: XCSFls needs not only a larger amount of memory for storing the last visited inputs, but also more computational time in order to solve at each step the Equation 6 for each classifier in the match set. Anyway these additional computational resources could be dramatically reduced by implementing a recursive version of linear least-squares that requires a smaller amount of memory and computational time.

7. REFERENCES

[1] J. A. Boyan. Least-squares temporal difference learning. In *Proc. 16th International Conf. on Machine Learning*, pages 49–56. Morgan Kaufmann, San Francisco, CA, 1999.

[2] S. J. Bradtke and A. G. Barto. Linear least-squares algorithms for temporal difference learning. *Machine Learning*, 22(1-3):33–57, 1996.

[3] S. A. Glantz and B. K. Slinker. *Primer of Applied Regression & Analysis of Variance*. McGraw Hill, 2001. second edition.

[4] S. Haykin. *Adaptive Filter Theory*. Prentice-Hall, 1986.

[5] J. Koza. *Genetic Programming*. MIT Press, 1992.

[6] P. L. Lanzi. The xcs library. `http://xcslib.sourceforge.net`, 2002.

[7] P. L. Lanzi, D. Loiacono, S. W. Wilson, and D. E. Goldberg. Generalization in the xcsf classifier system: Analysis, improvement, and extension. Technical report, Dipartimento di Elettronica e Informazione – Politecnico di Milano, 2005.

[8] W. H. Press, S. A. Teukolsky, W. T. Vetterling, and B. P. Flannery, editors. *Numerical Recipes in C++: The Art of Scientific Computing*. Cambridge University Press, Feb. 2002.

[9] S. W. Wilson. Mining Oblique Data with XCS. volume 1996 of *Lecture notes in Computer Science*, pages 158–174. Springer-Verlag, Apr. 2001.

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

[11] I. Zelinka. Analytic programming by means of soma algorithm. In *Proceedings of the 8th International Conference on Soft Computing, Mendel'02*, pages 93–101, Brno,Czech Republic, 2002.