Evolving Classifier Ensembles with Voting Predictors

Pier Luca Lanzi, Daniele Loiacono and Matteo Zanini

Abstract—In XCS with computed prediction, namely XCSF, the classifier prediction parameter is replaced by a parametrized prediction function. So far, the works on the computed prediction in XCS have been limited to evolve a single type of prediction function at once. Recently, several works studied and extended the computed prediction in XCSF. However, it is still not clear how the most adequate prediction function should be chosen for a given problem. In this paper we introduce XCSF with voting predictors that extends XCSF to let it select best prediction function to use in each problem subspace. We compared XCSFV to XCSF on several problems. Our results suggest that XCSFV performs as well as XCSF with the best prediction function in all the tested problems. In addition, XCSFV finds the most accurate prediction function in each problem subspace.

I. INTRODUCTION

XCS with computed prediction [23], namely XCSF, extends the typical idea of classifiers by replacing the prediction parameter with a prediction function \( p(s_t, w) \). The prediction function, usually defined as a linear combination of \( s \) and \( w \), computes the classifier prediction on the basis of the current state \( s_t \) and a parameter vector \( w \) associated to each classifier. Since its introduction, XCSF has been extended with several techniques for computing the classifier prediction, ranging from polynomial functions [12], to Neural Networks [11], and Support Vector Machines [19]. However the experimental results reported so far [13], [11], [19] suggest that none of such techniques outperforms the other ones in every respect. In particular, as discussed in [11], a powerful predictor may solve complex problems but may learn more slowly or be unnecessary expensive in the simple ones. Therefore, the choice of the predictor in each problem still requires human expertise and good knowledge of the problem. In this paper we introduce XCSF with voting predictors, dubbed XCSFV, which evolves classifiers that select the most appropriate type of prediction function. In XCSFV each classifier has a set of prediction functions or predictors and computes the prediction by choosing among them.

The idea of evolving ensembles of classifiers has been widely studied in the LCS research. Early examples include Dorigo and Colombetti [9] that used a hierarchy of learning classifier systems for controlling an autonomous robot; a similar approach has been used in [1], [4] for modeling an economic process. More recently [17] proposed to evolve an ensemble of classifiers with an heterogeneous knowledge representations. Finally, Bull et al. showed in [6], [5] that an ensemble of learning classifier systems may learn faster and more reliably than a single one. Our work differs from the previous ones basically in two respects: (i) previous works usually evolve classifiers of the same type (e.g., [6], [5]) or focus on different representations (e.g., [17]), while we propose a framework for evolving an ensemble of classifiers with different predictor types; (ii) previous frameworks usually deal with evolving an ensemble of complete problem solutions, while our approach aims to find one problem solution by evolving an ensemble of classifiers in each problem subspace.

II. THE XCSF CLASSIFIER SYSTEM

XCSF extends XCS in three respects [23]: (i) classifier conditions are extended for numerical inputs, as done for XCSI [22]; (ii) classifiers are extended with a vector of weights \( w_i \), that are used to compute classifier prediction; finally, (iii) the weights \( w_i \) are updated instead of the classifier prediction.

A. Classifiers

In XCSF, classifiers consist of a condition, an action, and four main parameters. The condition specifies which input states the classifier matches; it is represented by a concatenation of interval predicates, \( int_i = (l_i, u_i) \), where \( l_i \) ("lower") and \( u_i \) ("upper") are reals (whereas in the original XCSF they were integers [23]). The action specifies the action for which the payoff is predicted. The four parameters are: the weight vector \( w \), used to compute the classifier prediction as a function of the current input; the prediction error \( \varepsilon \), 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. The weight vector \( w \) has one weight \( w_i \) for each possible input, and an additional weight \( w_0 \) corresponding to a constant input \( x_0 \), that is set as a parameter of XCSF.

B. 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 \( s_t \); if \([M]\) contains less than \( b_{\text{max}} \) actions, covering takes place; covering is controlled by the parameter \( r_0 \) and it works as in XCSI [22], [23] but considers real values instead of integers. The weight vector \( w \) of covering classifiers is initialized to zero; all the other parameters are initialized as in XCS.

For each action \( a_i \) in \([M]\), XCSF computes the system prediction. As in XCS, in XCSF the system prediction of
IV. XCSF WITH VOTING PREDICTORS

XCSF with voting predictors, dubbed XCSFV, extends XCSF basically in two respects: (i) the definition of classifiers prediction function and (ii) the update of classifiers prediction.

A. Classifiers in XCSFV

In XCSFV the classifiers are extended with a set of prediction functions of different types. For each prediction function, or predictor, the classifiers store a parameter vector and an error estimate. Accordingly, the classifiers prediction is computed as the output of predictor with the lowest error estimate among the predictors set associated to the classifiers. On the other hand, the update of classifier prediction is straightforward: the expected payoff is used to update at the same time all the predictors and their error estimates. The discovery component works exactly as in XCSF.

B. Predictor Ensembles

In principle, XCSFV can evolve ensembles of classifiers with any type of predictors. However, in this paper we focus on the two examples of predictor ensembles described in the following.

Polynomial Prediction Functions. As a first example of predictor ensemble, we considered three polynomial prediction functions: a linear, quadratic and a cubic prediction function [12]. Polynomial predictors compute the classifier prediction as a linear combination between a parameter vector $w$ and the input vector or an extended input vector. In particular, the linear predictor has a parameter vector with one weight $w_i$ for each possible input and an additional weight $w_0$ corresponding to a constant input $x_0$. The quadratic predictor extends the input vector with one quadratic term for each possible input. Accordingly also the weight vector is extended with one new weight for each inputs introduced. Finally, cubic predictor further extends the input vector and the weight vector adding cubic terms.

Ensembles of Constant, Linear and Neural Predictors.

The second ensemble studied involves three heterogeneous predictor types: (i) a constant prediction, that is the one used in XCS; (ii) a linear prediction function, that is the one used in XCSF; (iii) the neural prediction function, introduced in [11]. In particular, the neural prediction function exploits a feedforward neural network to compute the classifier prediction. In this work we used always neural networks with a fixed topology, although in [11], [18] we showed that the genetic algorithms can be exploited also to adapt the networks topology.

IV. XCSFV ON FUNCTION APPROXIMATION PROBLEMS

We compared XCSFV to XCSF on several function approximation problems. In the first set of experiment we focus on XCSFV with ensembles involving polynomial predictors, i.e. linear, quadratic and cubic predictors. In the second set of experiments we study XCSFV with the ensemble of predictors involving constant, linear and neural predictors.
All the experiments discussed in this section follow the standard design used in the literature [21], [23]. In each experiment the system has to learn to approximate a target function \( f(x) \); each experiment consists of a number of problems that the system must solve. For each problem, an example \((x, f(x))\) of the target function \( f(x) \) is randomly selected; \( x \) is input to the system, which computes the approximated value \( \hat{f}(x) \) as the expected payoff of the only available dummy action; the action is virtually performed (the action has no actual effect), and XCSF receives a reward equal to \( f(x) \). The system 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; during test problems it is turned off. Classifier parameters are always updated. The covering operator is always enabled, but operates only if needed. Learning problems and test problems alternate.

The performance of the compared systems is measured as the average system prediction error. In addition, we analyzed the size of the populations evolved as a measure of the systems generalization capabilities. All the experiments reported have been conducted on xcslib [10].

A. Experiments with Polynomial Predictors

In the first experiment we applied XCSFV with polynomial predictors to the approximation of the \( f_{lt} \) function (see Figure 1a), defined as follows:

\[
f_{lt}(x) = \begin{cases} 
5x + 2 & \text{for } x < 0.5 \\
4.5 + 0.1 \cdot (x - 0.5) & \text{for } 0.5 \leq x \leq 1 
\end{cases}
\]  

(5)

XCSF has been compared to different XCSF systems: XCSF with (i) linear predictors, (ii) quadratic predictors, and (iii) cubic predictors. Experiments were performed with the same parameter settings used in [18]: \( N = 200, \beta = 0.2, \delta = 0.2, \gamma = 0.7, \theta_{GA} = 50, \chi = 0.8, \mu = 0.04, \epsilon_0 = 0.01, \nu = 5, \alpha = 0.1, \theta_{del} = 50, \) GA subsumption is on with \( \theta_{sub} = 50; \) action-set subsumption is not used; \( m_0 = 0.5, r_0 = 0.5 \) [23]; the classifier weight vector is updated using the Recursive Least Squares, with \( \delta_{rls} = 100 \) [16];

Figure 1c shows the performances of the compared systems: all the predictors compared, i.e. linear, quadratic and cubic perform almost in the same way. XCSFV seems to converge slightly faster than XCSF systems but the difference is not significant. The analysis of the generalization capabilities, reported in Figure 1d, shows that all the compared systems evolve populations of almost the same size. The population evolved by XCSFV seems slightly bigger than the others. In general we can state that the function \( f_{lt} \) is too simple to make the predictors perform differently. However it is interesting to analyze Figure 1b that shows what type of predictors have been selected by XCSFV for computing prediction over the input space. As expected, XCSFV mainly selects linear predictor, being \( f_{lt} \) a composition of two lines. Instead near the slope changing point, i.e., around \( x = 0.5, \) quadratic and cubic predictors are more suitable, as confirmed also by the experimental analysis.

Fig. 1. XCSFV and XCSF with linear, quadratic and cubic predictors applied to \( f_{lt}(x) \) function: (a) the target function, (b) frequencies of predictor types evolved, (c) performance, and (d) population size. Curves are averages over 50 runs.
In the second experiments we applied XCSF and XCSFV systems to the approximation of the function \( f_{abs} \) depicted in Figure 2a and defined as follows:

\[
f_{abs}(x) = |\sin(2\pi x) + |\cos(2\pi x)||
\]  

(6)

We used the same parameter settings of the previous experiment, except for the population size \( N = 800 \). Figure 2c shows that even if all the systems evolve an accurate approximation, XCSF with linear predictors is slightly slower to converge toward an accurate solution. On the other hand, XCSFV converges as fast as XCSF with quadratic and cubic predictors. Figure 2d compares the size of the populations evolved by XCSFV and XCSF. When linear predictors are used, XCSF evolves a slightly bigger population with respect to the one evolved using quadratic and cubic ones. Instead XCSFV evolves a population slightly more compact than the ones evolved by XCSF. Then, Figure 2b shows the predictor types selected by XCSFV over the input space. As expected, when \( x < 0.6 \), i.e., where \( f_{abs} \) is far from being linear, the cubic and quadratic predictors are selected more frequently than linear ones. Instead, when \( x > 0.6 \) also linear predictors can provide an accurate approximation of the target function on large input intervals. The capability of XCSFV of selecting different predictor types in different subspaces explains why it is able to evolve a slightly more compact solution.

To summarize, XCSFV performs almost as XCSF with the best predictor type. Both in terms of prediction accuracy and generalization capability. In addition, although so far we applied XCSFV to simple functions, our results suggest that XCSFV selects the most appropriate predictor type in each problem subspace.

### B. Experiments with Constant, Linear and Neural Predictors

In this second set of experiments, we move on the analysis of XCSFV with constant, linear and neural predictors ensemble. Please notice that XCSF with constant predictors is basically the Wilson’s XCSI [22] while XCSF with neural predictor is exactly XCSFN system introduced in [11].

In the first experiment we compared XCSFV to the XCSF on the \( f_3 \) function [7] defined as,

\[
f_3(x_1, x_2) = \sin(2\pi(x_1 + x_2)).
\]  

(7)

The parameters are set according to [7] as follows: \( N = 2000, \beta = 0.5, \delta = 0.5, \gamma = 0.9, \theta_{GA} = 50, \chi = 1.0, \mu = 0.05, \epsilon_0 = 0.01, \nu = 5, \alpha = 0.1, \theta_{del} = 20, \) GA subsumption is on with \( \theta_{sub} = 20; \) action-set subsumption is not used; \( m_0 = 1.0, \) \( r_0 = 0.5 \) [23]. Figure 3 compares the performances of the systems studied, both in terms of predictive accuracy (Figure 3a) and generalization capabilities (Figure 3b). Figure 3a shows that XCSF is able to evolve an accurate solution only with neural predictors (XCSF with constant predictors, not reported in the figure, reaches an average prediction error around 0.2). It is worthwhile to say that XCSFV is not only able to evolve an accurate solution but also learns almost as fast as XCSF with neural predictors. Figure 3b
Fig. 3. XCSFV compared to XCSF with constant, linear and neural network predictors on the approximation of $f_3(x_1, x_2)$: (a) performance and (b) population size. Curve are averaged over 20 runs.

Fig. 4. Frequencies of the predictors types selected by XCSFV applied to $f_3(x_1, x_2)$: (a) the target function, (b) constant predictors, (c) linear predictors, (d) neural predictors.

Fig. 5. XCSFV compared to XCSF with constant, linear and neural network predictors on the approximation of $f_4(x_1, x_2, x_3, x_4)$: (a) performance and (b) population size. Curve are averaged over 20 runs.

compares the size of the populations evolved by the systems. XCSF with neural predictors and XCSFV evolve a more compact solution than the one evolved by XCSF. Also in this experiment, we analyzed the predictor types selected by XCSFV over the problem space. Figure 4 shows that XCSFV mainly evolved neural predictors being the most accurate predictor. However XCSFV evolves also a number of linear predictors in two problem subspaces: when both $x_1$ and $x_2$ are close to 0 and, especially, when they are both close to 1. In such subspaces, in fact, the target function can be approximated accurately with a linear function.

In the next experiment we applied XCSFV and XCSF to the four variable function $f_4(x_1, x_2, x_3, x_4)$ [7], defined as

$$f_4(x_1, x_2, x_3, x_4) = x_2 + x_4 + 2\pi \sin x_1 + 2\pi \sin x_3$$

We used the same parameter settings of the previous experiment. Figure 5a compares the performances of XCSFV and XCSF (XCSF with constant predictors, not reported in the figure, converges to an average prediction error around 0.3). Results show that XCSF with linear predictors and XCSFV have almost an identical performance and they converge faster than XCSF with neural predictors toward an accurate solution. As discussed in [18], this is due to the fact that in the $f_4$ function each variable contribute is linearly separable. Figure 5b shows the size of the populations evolved by the systems. Notice that, although neural predictors learns slower, they let XCSF to evolve a more compact solution than linear ones. On the other hand XCSFV evolves a...
solution not as compact as XCSF with neural predictors, but more compact than the one evolved using linear predictors. In this experiment we cannot show the type of predictors selected over the input space due to the high dimensionality of the problem. However, XCSFV selected overall 63.7% of linear predictors, 36.3% of neural predictors, and a negligible fraction of constant predictors.

V. XCSFV ON MULTISTEP PROBLEMS

So far, we focused on function approximation problems. Now we compare XCSFV to XCSF on multistep problems, following the standard experimental design used in the literature [21], [14], [15]. For the sake of simplicity, we studied only ensembles of polynomial predictors on this set of experiments.

A. 2D Continuous Gridworld

In first experiments, we applied XCSFV to a class of real valued multistep environments, the 2D Continuous Gridworld [3]. The empty continuous gridworld, Grid(s) in brief, is a two dimensional environments in which the current state is defined by a pair of real valued coordinates (x, y) in [0, 1]^2, the only goal is in position (1, 1), and there are four possible actions (left, right, up, and down) coded with two bits. Each action corresponds in a step of size s in the corresponding direction; actions that would take the system outside the domain [0, 1]^2 take the system to the nearest position of the grid border. The system can start anywhere but in the goal position and it reaches the goal position when both coordinates are equal or greater than one. When the system reaches the goal it receives a reward of 0, in all the other cases it receives -0.5. The performance is computed as the average number of steps needed to reach the goal position during the last 100 test problems. Given the step-size s, the average optimal number of steps to reach the goal in Grid(s) is equal to (s+1)/s [21], [14]. In this paper we set s = 0.05 a rather typical value used in the literature [21], [14].

In first experiments, we applied XCSFV to the Grid(0.05) problem, with the following parameter settings: N = 5000, β = 0.2, δ = 0.2, γ = 0.95, θ_G,N = 50, χ = 0.8, μ = 0.04, ε_0 = 0.005, ν = 5, α = 0.1, θ_del = 50, GA subsumption is on with θ_sub = 50; action-set subsumption is not used; rm_q = 0.5, rq = 0.5 [23].

Figure 6a compares the performance of XCSFV and XCSF with different predictors. All the systems converge to an optimal solution, but XCSF with linear predictor learns slower than XCSF with quadratic and cubic predictors. On the other hand, XCSFV converges almost as fast as XCSF with quadratic predictors. XCSF with cubic predictors, instead, seems slightly faster than all the other system. Figure 6b shows that all the systems evolve populations of almost the same size. As we did for the function approximation problems, we analyzed the type of predictors selected by XCSFV over the problem space. Results (see Figure 7) suggest that the selected predictors are quite evenly divide between the available types over the problem space. This can be explained observing that typically in multistep problems generalization is very limited. Therefore all the polynomial predictors used, are equally able to provide an accurate local approximation of the action-value function.

B. 2D Continuous Gridworld with Puddles

In the second set of experiments we move to a slightly more complex problem, that is the 2D Continuous Gridworld with Puddles, Puddles(s) in brief. This problem adds some obstacles to 2D Continuous Gridworld, defined as areas involving an additional cost for moving. These areas are called “puddles” [3], since they actually create a sort of puddle in the optimal value function. We used the same experimental design followed in [14]. Figure 8 shows the Puddles(s) environment. In this paper we set s = 0.05, a rather typical value in the literature. Moving through a puddle results in an additional negative reward of -2; in the area where the two puddles overlap, the darker gray region, the negative rewards sum up, i.e., the additional negative reward is -4.

We compared XCSF to XCSFV, using the same parameter settings of the previous experiment. Figure 9 compares the performance of XCSFV and of XCSF with different predictors. Results shows that all the systems compared perform basically the same and evolve almost the same number of macroclassifiers. The optimal value function of the Puddles(0.05), reported in Figure 10a, allows even less generalizations with respect to Grid(0.05) problem.
Fig. 7. Types of predictor selected by XCSFV: (a) the optimal value function of \(\mathrm{Grid} (0.05)\), (b) linear predictors, (c) quadratic predictors, and (d) cubic predictors. Statistics are averaged over 10 runs.

Therefore, for the same reason previously discussed, the ratios of classifier predictors are almost the same.

VI. CONCLUSIONS

In this paper we proposed an approach to extend XCSF to let it evolve ensembles of classifiers with different types of prediction functions. Our approach differs from previous works on classifier ensembles basically in two respects: (i) previous works usually evolve classifiers of the same type or focus on classifier representation, while we evolve classifiers with different types of predictors; (ii) previous works usually involve many populations or many systems, while we deal with evolving different types of classifiers in the same population. Even if, in principle, in XCSFV any type of predictor can be used, we focused on the two specific ensembles: (i) ensembles of polynomial predictors, i.e., linear, quadratic and cubic and (ii) ensembles of constant, linear and neural predictors. We investigated the capabilities of XCSFV by comparing it to XCSF on several problems. Our results suggest that XCSFV performs as XCSF with the best

Fig. 8. The Puddles \(s\) problem. The goal is placed in \((1, 1)\). Gray zones represent the “puddles”.

Fig. 9. XCSFV and XCSF with linear, quadratic and cubic predictors applied to Puddles \(0.05\): (a) performance and (b) population size. Curves are averaged over 10 runs.

Fig. 10. Types of predictor selected by XCSFV: (a) the optimal value function of Puddles \(0.05\), (b) linear predictors, (c) quadratic predictors, and (d) cubic predictors. Statistics are averaged over 10 runs.
predictor type. Finally, we analyzed the type of predictors selected by XCSFV. Results show that XCSFV selects the most appropriate predictor type in each problem subspace, i.e., the one that provides the most accurate approximation of the payoff landscape.

REFERENCES