Improving Classifier Error Estimate in XCSF

Daniele Loiacono*, Jan Drugowitsch‡, Alwyn Barry‡, and Pier Luca Lanzi* †

*Artificial Intelligence and Robotics Laboratory (AIRLab),
Politecnico di Milano. P.za L. da Vinci 32, I-20133, Milano, Italy
‡Department of Computer Science, University of Bath, UK
†Illinois Genetic Algorithm Laboratory (IlliGAL),
University of Illinois at Urbana Champaign,
Urbana, IL 61801, USA
J.Drugowitsch@bath.ac.uk, loiacono@elet.polimi.it,
A.M.Barry@bath.ac.uk, lanzi@elet.polimi.it

Abstract. We study the current definition of classifier error in XCSF and we discuss the limitations of the algorithm that is currently used to compute classifier error from online experience. We introduce a new definition of classifier error and study the performance of two novel estimation algorithms based on this definition. Our results suggest that the new estimation algorithms can be more robust and improve system generalization.

1 Introduction

Classifier error plays a key role in accuracy-based learning classifier systems. Since Wilson introduced XCS [9], classifier error has always been computed as an estimate of the mean absolute prediction error, adjusted using the Widrow-Hoff rule. Unfortunately, the prediction error update rule causes the error estimates to fluctuate which results in temporarily inaccurate and unreliable values, especially when computed prediction is used. XCS with computed prediction, namely XCSF [10], is a major advance in the field of learning classifier systems. It extends the typical idea of classifiers by replacing the classifier prediction parameter with a prediction function $p(s_t, \mathbf{w})$, which is used to compute classifier prediction based on the current state s_t and on a parameter vector $\mathbf{w}$ associated with each classifier.

The rule used in XCSF for updating the classifier weights was carefully analyzed and improved in [4]; in addition, different update rules have been compared in [6]. However, the classifier error in XCSF is still defined and computed exactly as in XCS and was not investigated until [1] have shown how to significantly improve that error estimate. In this paper we define a new classifier error *measure* for XCSF according to [1]: we suggest the replacement of the usually measured mean absolute prediction error with the mean square prediction error. Subsequently, we study the usual error update rule used in XCSF and we introduce two different approaches that improve the accuracy of the error computation.

We finally compare these new approaches with the usual classifier error computation and show that they enable better generalization and hence the evolution of more compact populations.

2 The XCSF classifier System

When compared to XCS, XCSF replaces the classifiers' scalar prediction parameter by a prediction function $p(\phi(\mathbf{s}_t), \mathbf{w})$ that is parameterised by a parameter vector $\mathbf{w}$. This function computes the prediction as a function of the feature vector $\phi(\mathbf{s}_t)$, extracted from the current sensory input $\mathbf{s}_t$, and the classifier's parameter vector $\mathbf{w}$ that replaces the scalar prediction parameter; to keep the notation uncluttered we will for the rest of this paper use $\boldsymbol{\phi}_t \equiv \phi(\mathbf{s}_t)$ for the feature vector that corresponds to the sensory input $\mathbf{s}_t$, and $cl.p(\boldsymbol{\phi}_t) \equiv p(\boldsymbol{\phi}_t, \mathbf{w})$ for the classifier's prediction for $\mathbf{s}_t$. Usually, $p(\boldsymbol{\phi}_t, \mathbf{w})$ is computed by the linear combination $p(\boldsymbol{\phi}_t, \mathbf{w}) = \mathbf{w}^T \boldsymbol{\phi}_t$, where the feature vector is given by $\boldsymbol{\phi}^T = [x_0, \mathbf{s}_t(1), \ldots, \mathbf{s}_t(n-1)]^T$, x_0 is a fixed parameter (that is, a constant term), and $n-1$ is the size of the sensory input vectors $\mathbf{s}_t$, such that the feature vectors $\boldsymbol{\phi}_t$ are of size n. Even though it is possible to use non-linear functions to compute the prediction in XCSF [5], we will in this paper exclusively consider the just introduced linear function.

To update the classifiers' parameter vectors and error estimates, 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 $\mathbf{s}_t$. For each action a_i in [M], XCSF computes the *system prediction*, which for action a is the fitness-weighted average of the predictions computed by all classifiers in [M] that promote this action. Next, XCSF selects an action to perform. The classifiers in [M] that advocate the selected action form the current *action set* [A]; the selected action is sent to the environment and a reward r is returned to the system together with the next input. XCSF uses this reward to update the parameters of classifiers in action set $[\mathrm{A}]_{-1}$, corresponding to the previous time step in which the reward was received. Note that, when XCSF is used for function approximation (a single-step problem), the reinforcement component acts on the current action set rather than the previous one. At time step t, the expected payoff P is computed as $P = r_{-1} + \gamma \max_{a \in A} P(\mathbf{s_t}, a)$, The expected payoff P is used to update the weight vector $\mathbf{w}$ of the classifiers in $[\mathrm{A}]_{-1}$ using the Widrow-Hoff rule, also known as the *modified delta rule* [7]. For each classifier $cl \in [A]_{-1}$, each component $\mathbf{w}(i)$ of the classifier's weight vector $\mathbf{w}$ is adjusted by a quantity $\Delta\mathbf{w}(i)$ given by

$$\Delta\mathbf{w}(i) = \frac{\eta \boldsymbol{\phi}_{t-1}(i)}{\|\boldsymbol{\phi}_{t-1}\|^2}(P - cl.p(\boldsymbol{\phi}_{t-1})), \tag{1}$$

where η is the correction rate and $\|\boldsymbol{\phi}_{t-1}\|^2$ is the squared Euclidean norm of the input vector $\boldsymbol{\phi}_{t-1}$ [10]. The values $\Delta\mathbf{w}(i)$ are used to update the weights of classifier cl by

$$\mathbf{w}(i) \leftarrow \mathbf{w}(i) + \Delta\mathbf{w}(i). \tag{2}$$

Then the prediction error ε is updated by

$$\varepsilon \leftarrow \varepsilon + \beta(|P - cl.p(\boldsymbol{\phi}_{t-1})| - \varepsilon). \tag{3}$$

Finally, the classifier fitness is updated as usual [9] and the discovery component is applied as in XCS.

3 Tracking the Root Mean Square Error in XCSF

In XCSF the classifier weight vector is adjusted to minimise the mean squared prediction error (MSE), while the classifier's error is an estimate of the mean absolute prediction error (MAE). Before discussing the consequences of this inconsistency, let us firstly show that this claim is correct.

For the rest of this paper we will consider a single classifier and will assume the sequence $t = 1, 2, \ldots$ to represent each time step in which this classifier participates in the action set (making it equivalent to the classifier's *experience*) and thus will be updated.

3.1 Re-Deriving the XCSF Weight Vector and Error Update

Let us assume that we have N states $\{\mathbf{s}_t\}_{t=1}^N$ and their associated payoffs $\{P_t\}_{t=1}^N$, and that we want to estimate the classifier's weight vector $\mathbf{w}$ that minimises the MSE, given by

$$f_N(\mathbf{w}) = \frac{1}{N}\sum_{t=1}^{N}\left(\mathbf{w}^T\boldsymbol{\phi}_t - P_t\right)^2, \tag{4}$$

where we have again used $\boldsymbol{\phi}_t = \phi(\mathbf{s}_t)$. Applying the modified delta rule (also known as the *Normalised Least Mean Squared* algorithm) to minimise $f_N(\mathbf{w})$ results in the weight vector update equation

$$\mathbf{w}_t = \mathbf{w}_{t-1} + \frac{\eta\boldsymbol{\phi}_t}{\|\boldsymbol{\phi}_t\|^2}\left(P_t - \mathbf{w}_{t-1}^T\boldsymbol{\phi}_t\right), \tag{5}$$

which is equivalent to Eqs. 1 and 2 and thus confirms that the XCSF weight vector update indeed aims at minimising the MSE $f_N(\mathbf{w})$.

To get the prediction error, on the other hand, let us assume that we want to estimate the MAE, given by

$$\varepsilon_N = \frac{1}{N}\sum_{t=1}^{N}\left|\mathbf{w}^T\boldsymbol{\phi}_t - P_t\right|. \tag{6}$$

This estimation problem can be reformulated as a least squares problem that minimises

$$g_N^{\mathrm{MAE}}(\mathbf{w}) = \frac{1}{N}\sum_{t=1}^{N}\left(\varepsilon_N - \left|\mathbf{w}^T\boldsymbol{\phi}_t - P_t\right|\right)^2 \tag{7}$$

with respect to ε_N. Solving $\partial g_N^{\text{MAE}}(\mathbf{w})/\partial \varepsilon_N = 0$ for ε_N results in Eq. 6, which confirms that we can indeed estimate ε_N by minimising $g_N^{\text{MAE}}(\mathbf{w})$. Applying the delta rule (also known as the *Least Mean Squared* algorithm) to minimising $g_N^{\text{MAE}}(\mathbf{w})$ results in the prediction error update

$$\varepsilon_t = \varepsilon_{t-1} + \beta \left(|\mathbf{w}_t^T \boldsymbol{\phi}_t - P_t| - \varepsilon_{t-1} \right), \tag{8}$$

where we have approximated the weight vector by its current estimate $\mathbf{w} \approx \mathbf{w}_t$. This update equation is equivalent to Eq. 3, which shows that XCSF estimates the mean absolute prediction error rather than the mean squared prediction error.

Consequently, the performance component of XCSF that estimates the weight vector aims at minimising the MSE, while the discovery component judges the prediction quality of classifiers based on the MAE. This inconsistency is usually not a serious issue because an optimal solution with respect to the MSE is also nearly optimal with respect to the MAE. Moreover, the MAE is superior to the MSE in terms of human readability; that is, while a threshold on the MAE of the classifier prediction can be easily related to the expected accuracy of the evolved approximation, a threshold on the MSE is not easily related to the final approximation accuracy. Unfortunately, it is rather difficult to find estimators that minimise the MAE (for an XCSF-related example see [**?**]), whilst there are numerous techniques in the literature that provide accurate estimates that minimise the MSE. This is a concern for XCSF, where the prediction error estimate should reflect the actual prediction error. Thus, we propose replacing the MAE estimate by the MSE estimate, as shown in the following section.

3.2 Estimating the Root Mean Squared Error

We can estimate the MSE (Eq. 4) in the same way as the MAE (Eq. 6) by reformulating its estimation as a least squares problem that minimises

$$g_N^{\text{MSE}}(\mathbf{w}) = \frac{1}{N} \sum_{t=1}^{N} \left(\varepsilon_N^2 - (\mathbf{w}^T \boldsymbol{\phi}_t - P_t)^2 \right)^2 \tag{9}$$

with respect to ε_N^2, which denotes the MSE estimate $\varepsilon_N^2 \equiv \hat{f}_N(\mathbf{w})$. Applying the delta rule by again approximating $\mathbf{w}$ by its estimate $\mathbf{w}_t$ gives the update equation

$$\varepsilon_t^2 = \varepsilon_{t-1}^2 + \beta \left((\mathbf{w}_t^T \boldsymbol{\phi}_t - P_t)^2 - \varepsilon_{t-1}^2 \right), \tag{10}$$

from which we compute the classifier error by

$$\varepsilon_t = \sqrt{\varepsilon_t^2}. \tag{11}$$

Thus, it is given by the estimated root mean squared error (RMSE) of the prediction. We use the RMSE instead of the MSE because (i) the RMSE is a standard error measure in the machine learning literature (for example, [**?**]),

and (ii) the RMSE has the same value range as the MAE that is usually used in XCSF.

Relying on the MSE instead of the MAE has the additional advantage that we do not need to estimate it by the delta rule, as in Eq. 10, but can track the solution to $f_N(\mathbf{w})$ directly, as we will show in the following section.

4 Improving the Error Estimate

In previous works [**?**,1] the problem of computing the classifier prediction has been presented as a problem of incremental parameters estimation. In this section we show that such a problem may be solved within a particular bayesian framework, the Bayes Lnear Analysis [**?**]. At first we show how both the classifier weights and the classifier error can be computed at the same time from a probabilistic point of view. Then we propose a simple implementation using only the samples matched by the classifier and we show that this implementation is equal to the least squares update introduced in [**?**]. Finally we show that the proposed framework is also consistent with the more convenient Recursive Least Squares approach described in [**?**] and the incremental error update introduced in [1].

4.1 The Bayes Linear Analysis

When computing the classifier prediction in XCSF we want to predict, with highest accuracy possible, the value of the target payoff, P_t, on the basis of the observed features vector, $\boldsymbol{\phi}_t$. From a probabilistic point of view our aim is to compute the conditional expectation $E[P|\boldsymbol{\phi}]$. The conditional expectation may be computed in the bayesian framework, but this approach requires a full knoweledge about the probability distribution of both P and $\boldsymbol{\phi}$, that is usually not available. On the other hand in XCSF we assume that the target payoff can be computed with a linear prediction function, $cl.p(\boldsymbol{\phi}_t) = \mathbf{w}^T\boldsymbol{\phi}_t$ (see Section 2). Under this assumption it is possible to apply the Bayes Linear Analysis for computing a linear model without a full probabilistic description of P and $\boldsymbol{\phi}$ [**?**]. Following the Bayes Linear Analysis, the optimal value for the classifier weights vector, $\mathbf{w}$, can be computed by solving the following minimisation problem:

$$\min_{\mathbf{w}} E[(P - \mathbf{w}^T\boldsymbol{\phi}(\mathbf{s}))^2] \tag{12}$$

The minimization problem stated by Equation 12 aims to find the classifier weights vector that minimizes the expected square error of the classifier prediction. Solving the minimization problem in Equation 12 we obtain the following weights vector (see Appendix **??** for the derivation),

$$\mathbf{w} = E_{\boldsymbol{\phi}\boldsymbol{\phi}}^{-1} E_{P\boldsymbol{\phi}} \tag{13}$$

where $E_{\boldsymbol{\phi}\boldsymbol{\phi}} = E\left[\boldsymbol{\phi}\boldsymbol{\phi}^T\right]$ is a $n \times n$ square matrix and $E_{P\boldsymbol{\phi}} = E\left[P\boldsymbol{\phi}\right]$ is a column vector $(n \times 1)$. Once the optimal value of weight vector is computed, the

square classifier error can be estimated as the expected square prediction error by replacing Equation 13 into the minimisation objective of Equation 12. This lead to the following classifier error,

$$\varepsilon^2 = E\left[(P - \mathbf{w}^T\boldsymbol{\phi})^2\right] = E_{PP}(1 - \rho^2), \tag{14}$$

where $E_{PP} = E\left[P^2\right]$ and ρ^2 is defined as

$$\rho^2 = \frac{E_{P\boldsymbol{\phi}}^T E_{\boldsymbol{\phi}\boldsymbol{\phi}}^{-1} E_{P\boldsymbol{\phi}}^T}{E_{PP}} \tag{15}$$

Before going on with the implementation details, it is worthwhile to discuss the meaning of Equation 14. In fact considering that, in XCSF, $\boldsymbol{\phi}$ is defined as $\boldsymbol{\phi} = [1\mathbf{s}^T]$, we can rewrited as:

$$\varepsilon^2 = cov(P, P)(1 - \rho_{P\mathbf{S}}^2), \tag{16}$$

where $cov(P, P) = E\left[(P - E[P])^2\right]$ is the variance of the target payoff, P, and $\rho_{P\mathbf{S}}^2$ is the square correlation coeffecient between P and $\mathbf{s}$ defined as,

$$\rho_{P\mathbf{S}}^2 = \frac{cov(P, \mathbf{s})^T cov(\mathbf{s}, \mathbf{s})^{-1} cov(P, \mathbf{s})}{cov(P, P)} \tag{17}$$

where $cov(P, \mathbf{s}) = E[(P - E[P])(\mathbf{s} - E[\mathbf{s}])]$ is the covariance vector $(n \times 1)$ between target payoff, P, and the state vector, $\mathbf{s}$; $cov(\mathbf{s}, \mathbf{s}) = E\left[(s - E[s])(\mathbf{s} - E[\mathbf{s}])^T\right]$ is the variance matrix of the state vector, $\mathbf{s}$. On the basis of Equation 16 we can provide an intuitive interpretation to the classifier error estimate. When P and $\mathbf{s}$ are completely uncorrelated (i.e. $\rho_{P\mathbf{S}}^2 = 0$), it is not possible to proved any prediction of the target payoff better than its expected value; therefore the expected square prediction error is equal to the variance of P. On the other hand, when P and $\mathbf{s}$ are linearly maximally correlated (i.e. $\rho_{P\mathbf{S}}^2 = 1$), the target payoff can be predicted without any error; therefore the expected square prediction error is equal to 0. In all the other cases, the higher is the correlation between P and $\mathbf{s}$, the more accurate is the target payoff prediction and, therefore, the lower is the expected square prediction error.

4.2 A Sample Based Implementation and Least Squares

In Equation 13 and Equation 14 we assumed the knoweledge of E_{PP}, $E_{P\boldsymbol{\phi}}$ and $E_{\boldsymbol{\phi}\boldsymbol{\phi}}$. Unfortunately such knowledge is not available when computing the classifier weights and error. In order to apply the previous introduced equation

to XCSF, we introduced the following sample-based approximations:

$$E_{PP} \approx \frac{1}{N}\sum_{i=1}^{N}(P_i)^2$$

$$E_{\boldsymbol{\phi}\boldsymbol{\phi}} \approx \frac{1}{N}\sum_{i=1}^{N}(\boldsymbol{\phi}_i\boldsymbol{\phi}_i^T)$$

$$E_{P\boldsymbol{\phi}} \approx \frac{1}{N}\sum_{i=1}^{N}(P_i\boldsymbol{\phi}_i)$$

where each term can be updated recursively during the learning process and is stored as a classifier parameter. In the rest of the paper we will call XCSFb the variant of XCSF obtained using the Equation 13 for computing the classifier weight vector and the Equation 14 for computing the square classifier error, where the above sample-based approximation are used.

Before introducing a different implementation it is wortwhile to say that, using the sample-based approximation introduced above, the Equation 13 is equal to the least squares update introduced in [**?**] (see Appendix A for the proof). It can also proved (Appendix A) that the error update Equation 14 is exactly the sample-based mean square prediction error of the least square solution, that is

$$\varepsilon^2 = \frac{1}{N}\sum_{i=1}^{N}\left(P_i - \mathbf{w}^T\boldsymbol{\phi}_i\right)^2$$

where $\mathbf{w}$ is computed according to Equation 13 that is equal to the least squares solution.

4.3 Recursive Least Squares and Error Tracking

In [4] the Widrow-Hoff rule used in XCSF (Equation 1) was replaced by the Recursive Least Squares (RLS) algorithm, given by:

$$\beta_{rls} \leftarrow 1 + \boldsymbol{\phi}_{t-1}^T \times cl.V \times \boldsymbol{\phi}_{t-1}, \tag{18}$$

$$cl.V \leftarrow cl.V - \frac{1}{\beta_{rls}} cl.V \times \boldsymbol{\phi}_{t-1}^T \times \boldsymbol{\phi}_{t-1} \times cl.V, \tag{19}$$

$$\mathbf{w} \leftarrow \mathbf{w} + cl.V \times \boldsymbol{\phi}_{t-1} \times (P - cl.p(\boldsymbol{\phi}_{t-1})), \tag{20}$$

where $cl.V$ is an $|\boldsymbol{\phi}| \times |\boldsymbol{\phi}|$ matrix for storing the estimate of the autocorrelation matrix of the feature vector $\boldsymbol{\phi}$. The above update equations allow each classifier to precisely track the weight estimate $cl.w$ that minimises the mean squared error, given by the convex function

$$\frac{1}{N}\sum_{i=1}^{N}(P(\mathbf{s}_i) - cl.p(\boldsymbol{\phi}_i))^2,$$

where the sum is taken over all states that the classifier matches. Compared to the Widrow-Hoff rule that only performs gradient descent on the local error, the RLS algorithm is more effective, as analyzed and demonstrated in [4], where the classifier weight update was computed using the RLS algorithm, but the error estimate was still performed as in the original XCSF [10].

Recently [1], by employing the Kalman filter for weight update and error estimate we have shown that, under the assumption of a constant equation/observation error variance, the Kalman filter weight update is equivalent to the one performed by the RLS algorithm as given above, but the error can be more efficiently tracked by

$$\varepsilon_s \leftarrow \varepsilon_s + \frac{\left[(P - cl.p(\boldsymbol{\phi}_t))(P - cl.p'(\boldsymbol{\phi}_t)) - \varepsilon_s\right]}{cl.exp}, \tag{21}$$

where $cl.p(\boldsymbol{\phi}_t)$ is the classifier prediction for the feature vector $\boldsymbol{\phi}_t$ before the weights update, while $cl.p'(\boldsymbol{\phi}_t)$ is the classifier prediction after the weights update, and $cl.exp$ is the classifier experience. This error update equation allows us to accurately track the mean squared error as given above, based on the current prediction applied to all matched states that have been observed so far, and at low additional computational costs.

In accordance with [4], the XCSF version that replaces the Widrow-Hoff weights update (Equations 1 and 2) by above RLS algorithm (Equations, 18, 19, and 20) while still using the original XCSF error update will be called XCSFrls. The XCSF version that in addition to the RLS algorithm for weight update uses the Kalman filter error update (Equation 21) will be called XCSFkal.

5 Experimental Design

All the experiments discussed in this paper concern the comparison of XCSFrls, XCSFb, and XCSFkal on function approximation tasks and are performed following the standard design used in the literature [10].

The systems have been tested on approximating the four functions given in Table 1, which are the real-valued versions of those used in [3]. In all the experiments the state $\mathbf{s}$ is the independent variable x of the functions, while the feature vector is set to $\boldsymbol{\phi} = [1, x]^T$, i.e. the constant term x_0 is always set to 1. The performance is measured by 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 *root mean square error* (RMSE) given by

$$RMSE = \sqrt{\frac{1}{n}\sum_x (f(x) - \hat{f}(x))^2},$$

where n is the number of sample points used to approximate the error. In particular, we consider the average RMSE over the performed experiments, dubbed $\overline{RMSE}$.

$$
\begin{aligned}
f_p(x) &= 1 + x + x^2 + x^3, \\
f_{abs}(x) &= |\sin(x) + |\cos(x)||\,, \\
f_{s3}(x) &= \sin(x) + \sin(2x) + \sin(3x), \\
f_{s4}(x) &= \sin(x) + \sin(2x) + \sin(3x) + sin(4x).
\end{aligned}
$$

Table 1. Functions used to test XCSF over the range $x \in [0, 1]$.

Statistical Analysis. To analyze the results reported in this paper, we performed an analysis of variance (ANOVA) [2] on the resulting performance and evolved population size. For each experiment and for each setting, we analyzed the final performance and the number of macroclassifiers evolved by the different versions of XCSF; we applied an analysis of variance to test whether there was some statistically significant difference; in addition, we applied four *post hoc tests* [2], Tukey HSD, Scheffé, Bonferroni, and Student-Neumann-Keuls, to find which XCSF variants performed significantly different.

6 Experimental Results

In the first experiment we let a single classifier approximate f_{abs} over the input range $[0.4, 0.6]$, and compare the classifier error estimates computed using the techniques introduced above. Figure 1 shows the classifier errors estimated by XCSFrls, XCSFb, and XCSFkal when approximating f_{abs}. Despite the initial performance difference between XCSFb and XCSFkal their error estimates quickly converge to an accurate and reliable value. On the other hand the XCSFrls error estimate (i.e. the usual XCSF classifier error estimate) is very noisy and not reliable when $\beta = 0.2$ (a typical value in literature) due to the stochastic fluctuation of the Widrow-Hoff update rule This fluctuation can be reduced by decreasing β, but as shown in Figure 1 for $\beta = 0.001$, this has the effect of increasing the time that the estimate requires to converge to the correct value. The results also show that while XCSFb initially underestimate the classifier error XCSFkal does not.

In the second set of experiments we applied XCSFrls, XCSFb, and XCSFkal to all four functions in Table 1, using the following parameters: $N = 800$; $\beta = 0.2$; $\alpha = 0.1$; $\nu = 5$; $\chi = 0.8$, $\mu = 0.04$, $\theta_{del} = 50$, $\theta_{GA} = 50$, and $\delta = 0.1$; GA-subsumption is on with $\theta_{GAsub} = 50$; when action-set subsumption is on, $\theta_{ASsub} = 100$ is used; $m_0 = 0.2$, $r_0 = 0.1$ [10]; in XCSFrls, XCSFkal we set $\delta_{rls} = 100$ [4]. Experiments were performed with the minimal error ϵ_0 set to 0.05, 0.1, and 0.2. The performance of XCSFrls, XCSFb, and XCSFkal is reported in Table 2 and is computed as the average RMSE of the evolved solutions over 20 runs. The results show that when action-set subsumption is off, then all three XCSF versions are accurate in the sense that their function approximation features an error that is lower than the minimal error ϵ_0. We can

Fig. 1. Error estimates of XCSFrls, XCSFb, and XCSFkal when applied to f_{abs}. Error estimates are reported for the first 1000 learning problems and refer to a single classifier matching all the inputs in the range $[0.4, 0.6]$.

also see that the error of XCSFrls is generally lower than that of XCSFb and XCSFkal (and therefore further away from the desired minimal error), which comes hand in hand with a larger evolved population, as shown in Table 3. This is not surprising as a more reliable classifier error estimate can be expected to improve the classifier system's generalization capabilities, which is confirmed by the outcome of the experiment.

On the other hand, when action-set subsumption is on, only XCSFb and XCSFkal are accurate, and XCSFrls features errors up to two times the desired error. This can be explained by action-set subsumption requiring accurate error estimates to only subsume classifiers that are indeed less general than they could be in order to still be considered accurate. In the case of XCSFrls the error estimates are noisy and so action-set subsumption can have disruptive effects on the evolved population by having overgeneral classifiers subsume optimally general classifiers because the overgeneral classifiers are temporarily considered to be accurate by the system.

The statistical analysis of the data reported in Table 2 and Table 3 shows that the differences between XCSFkal and XCSFb are always *not significant* with a 99.9% confidence. A further analysis of Table 2 shows that the differences between XCSFrls and the variants XCSFkal and XCSFb are *significant* with a 99.9% confidence for most of the experiments, with the differences not being significant only in the cases when complex function are required to be estimated with a low error, hence prohibiting generalization (e.g. f_{s3} and f_{s4} with $\epsilon_0 = 0.05$), or when the function is of low complexity and generalization is straightforward (e.g. f_p with $\epsilon_0 = 0.2$ and action-set subsumption).

With respect to the population size evolved by the three XCSF variants (given in Table 3), we have only performed significance tests on experiments where all

three variants produce an estimate error below the accuracy threshold ϵ_0, as we could produce an arbitrarily small population when ignoring the approximation error. The results indicate that when action-set subsumption is switched off, there is no significant difference between any of the systems under investigation. However, switching on action-set subsumption causes XCSFkal and XCSFb to always produce significantly smaller populations than XCSFrls, given that the estimation error of all three systems is below the accuracy threshold. This is not surprising because without action-set subsumption XCSFkal and XCSFb are not able to fully exploit their better error estimate and thus the differences are less significant.

In summary, the results confirm our hypotheses: the improved classifier error estimate in XCSFb and XCSFkal allows for better generalization and evolves more compact solutions closer to the desired error. In fact, the size of the populations evolved by XCSFb and XCSFkal are always significantly smaller than the ones evolved by XCSFrls except for when action-set subsumption is on and XCSFrls is not accurate. It is also worthwhile to observe that when action-set subsumption is off, XCSFkal evolves populations slightly more compact than XCSFb, which can be explained by its faster convergence to a good error estimate, as already shown in the first experiment.

$f(x)$	$\epsilon 0$	XCSFrls	XCSFb	XCSFkal	XCSFrls with ASS	XCSFb with ASS	XCSFkal with ASS
f_p	0.05	0.01 ± 0.00	0.02 ± 0.00	0.02 ± 0.00	0.08 ± 0.01	0.04 ± 0.00	0.04 ± 0.00
f_p	0.10	0.02 ± 0.00	0.03 ± 0.00	0.03 ± 0.00	0.18 ± 0.01	0.08 ± 0.01	0.08 ± 0.01
f_p	0.20	0.05 ± 0.01	0.09 ± 0.01	0.10 ± 0.01	0.19 ± 0.00	0.19 ± 0.00	0.19 ± 0.00
f_{s3}	0.05	0.03 ± 0.00	0.03 ± 0.01	0.03 ± 0.00	0.05 ± 0.01	0.04 ± 0.00	0.04 ± 0.02
f_{s3}	0.10	0.05 ± 0.00	0.05 ± 0.01	0.06 ± 0.00	0.10 ± 0.01	0.07 ± 0.00	0.07 ± 0.00
f_{s3}	0.20	0.08 ± 0.01	0.09 ± 0.01	0.10 ± 0.01	0.27 ± 0.02	0.14 ± 0.01	0.13 ± 0.01
f_{s4}	0.05	0.05 ± 0.04	0.04 ± 0.02	0.04 ± 0.02	0.06 ± 0.01	0.06 ± 0.03	0.04 ± 0.01
f_{s4}	0.10	0.05 ± 0.00	0.06 ± 0.00	0.07 ± 0.02	0.11 ± 0.01	0.07 ± 0.00	0.07 ± 0.00
f_{s4}	0.20	0.08 ± 0.01	0.10 ± 0.01	0.10 ± 0.01	0.28 ± 0.03	0.14 ± 0.01	0.14 ± 0.01
f_{abs}	0.05	0.02 ± 0.00	0.02 ± 0.00	0.02 ± 0.00	0.05 ± 0.00	0.03 ± 0.00	0.03 ± 0.00
f_{abs}	0.10	0.03 ± 0.00	0.05 ± 0.00	0.04 ± 0.00	0.14 ± 0.02	0.07 ± 0.01	0.07 ± 0.01
f_{abs}	0.20	0.08 ± 0.01	0.11 ± 0.01	0.11 ± 0.01	0.29 ± 0.01	0.16 ± 0.01	0.16 ± 0.01

Table 2. Performance of XCSFrls, XCSFb, and XCSFkal applied to f_p, f_{s3}, f_{s4}, and f_{abs}.

7 Conclusions

In this paper we proposed the modification of the usual classifier error definition: instead of the commonly used estimate of the mean absolute error, we defined the classifier error as the estimate of the mean squared prediction error. We also showed how the usual error computation techniques can result in a noisy and inaccurate estimate and we thus introduced two new error computation techniques to improve the classifier error estimate. The resulting XCSF versions,

$f(x)$	$\epsilon 0$	XCSFrls	XCSFb	XCSFkal	XCSFrls with ASS	XCSFb with ASS	XCSFkal with ASS
f_p	0.05	53.85 ± 6.10	52.35 ± 5.69	48.65 ± 5.02	15.45 ± 5.48	7.50 ± 2.09	7.85 ± 2.94
f_p	0.10	50.95 ± 5.75	44.90 ± 7.59	43.85 ± 4.91	5.30 ± 2.78	8.35 ± 3.29	9.75 ± 2.75
f_p	0.20	47.00 ± 5.89	42.05 ± 7.04	39.50 ± 4.98	4.25 ± 2.45	3.80 ± 2.38	3.80 ± 2.01
f_{s3}	0.05	83.70 ± 6.00	76.40 ± 5.62	75.95 ± 5.46	37.90 ± 4.77	39.95 ± 5.63	41.20 ± 5.72
f_{s3}	0.10	69.65 ± 6.18	69.40 ± 8.83	66.40 ± 6.49	25.35 ± 3.89	25.45 ± 5.07	25.55 ± 4.15
f_{s3}	0.20	66.15 ± 5.19	57.45 ± 7.54	56.70 ± 5.56	25.20 ± 4.06	17.70 ± 4.05	16.30 ± 4.01
f_{s4}	0.05	88.05 ± 7.39	90.40 ± 7.57	85.45 ± 5.75	47.60 ± 5.18	51.40 ± 4.76	53.10 ± 5.20
f_{s4}	0.10	79.15 ± 5.76	74.05 ± 8.24	70.60 ± 5.31	35.95 ± 5.31	36.80 ± 4.50	37.44 ± 6.42
f_{s4}	0.20	71.65 ± 5.60	68.75 ± 7.43	63.70 ± 6.51	31.60 ± 5.81	24.45 ± 4.25	24.55 ± 4.24
f_{abs}	0.05	62.15 ± 7.55	57.20 ± 7.03	60.35 ± 7.07	23.45 ± 3.29	18.35 ± 3.10	19.60 ± 2.68
f_{abs}	0.10	60.75 ± 5.41	55.30 ± 8.94	58.95 ± 8.24	21.50 ± 4.84	14.75 ± 2.43	15.85 ± 4.23
f_{abs}	0.20	55.80 ± 6.35	50.85 ± 5.15	52.40 ± 7.32	4.65 ± 1.96	11.15 ± 3.53	10.30 ± 3.14

Table 3. Average number of macroclassifiers evolved by XCSFrls, XCSFb, and XCSFkal applied to f_p, f_{s3}, f_{s4}, and f_{abs}.

called XCSFb and XCSFkal, were compared to the usual XCSFrls. Our results show that in general XCSFrls and XCSFkal evolve more compact solutions and, in particular, are able to evolve accurate solutions also when action-set subsumption is on. Our results do not show particular differences between XCSFb and XCSFkal besides the slightly faster convergence of XCSFkal, although further investigation of the precise difference is necessary.

A Bayes Linear Estimator and Linear Least Squares

In this section we show that linear least squares as the Bayes linear estimator implementation introduced in is equivalent to linear least squares Given the following definitions:

$$\boldsymbol{\phi} = [1|s]^T$$

$$\boldsymbol{\Phi} = \begin{bmatrix} \boldsymbol{\phi}_1^T \\ \boldsymbol{\phi}_2^T \\ \vdots \\ \boldsymbol{\phi}_N^T \end{bmatrix}$$

$$\boldsymbol{\Pi} = \begin{bmatrix} P_1 \\ P_2 \\ \vdots \\ P_N \end{bmatrix}$$

Now we prove that the weights vector of Bayes Linear Estimator (Equation **??**) solve the following least squares normal equations:

$$\boldsymbol{\Phi}^T \boldsymbol{\Phi} \mathbf{w} = \boldsymbol{\Phi}^T \boldsymbol{\Pi}$$

where

$$\mathbf{\Phi}^T\mathbf{\Phi} = \left[\begin{array}{c|c} N & \sum_{i=1}^N \mathbf{s}_i^T \\ \hline \sum_{i=1}^N \mathbf{s}_i & \sum_{i=1}^N \mathbf{s}_i\mathbf{s}_i^T \end{array}\right]$$

and

$$\mathbf{\Phi}^T\mathbf{\Pi} = \left[\begin{array}{c} \sum_{i=1}^N P_i \\ \hline \sum_{i=1}^N P_i\mathbf{s}_i \end{array}\right]$$

According to the previous definition and the sample-based approximations introduced in Section 4 we can now write the least squares normal equations as

$$N \left[\begin{array}{c|c} 1 & \bar{\mathbf{s}}^T \\ \hline \bar{\mathbf{s}} & \Lambda_s s + \bar{\mathbf{s}}\bar{\mathbf{s}}^T \end{array}\right] w = N \left[\begin{array}{c} \bar{P} \\ \hline \Lambda_{Ps} + \bar{P}\bar{\mathbf{s}} \end{array}\right]$$

where N can be simplified. The equivalence is proved by replacing in the previous equation w with the value in Equation **??** and showing it is an identity:

$$\begin{aligned}
\left[\begin{array}{c|c} 1 & \bar{\mathbf{s}}^T \\ \hline \bar{\mathbf{s}} & \Lambda_s s + \bar{\mathbf{s}}\bar{\mathbf{s}}^T \end{array}\right] \left[\begin{array}{c} \bar{P} - \Lambda_{Ps}^T\Lambda_{ss}^{-1}\bar{\mathbf{s}} \\ \hline (\Lambda_{Ps}^T\Lambda_{ss}^{-1})^T \end{array}\right] &\geqslant \left[\begin{array}{c} \bar{P} \\ \hline \Lambda_{Ps} + \bar{P}\bar{\mathbf{s}} \end{array}\right] \\
\left[\begin{array}{c} \bar{P} - \Lambda_{Ps}^T\Lambda_{ss}^{-1}\bar{\mathbf{s}} + \bar{\mathbf{s}}^T(\Lambda_{Ps}^T\Lambda_{ss}^{-1})^T \\ \hline \bar{P}\bar{\mathbf{s}} - \bar{\mathbf{s}}\Lambda_{Ps}^T\Lambda_{ss}^{-1}\bar{\mathbf{s}} + \Lambda_{ss}(\Lambda_{Ps}^T\Lambda_{ss}^{-1})^T + \bar{\mathbf{s}}\bar{\mathbf{s}}^T(\Lambda_{Ps}^T\Lambda_{ss}^{-1})^T \end{array}\right] &\geqslant \left[\begin{array}{c} \bar{P} \\ \hline \Lambda_{Ps} + \bar{P}\bar{\mathbf{s}} \end{array}\right] \\
\left[\begin{array}{c} \bar{P} - \Lambda_{Ps}^T\Lambda_{ss}^{-1}\bar{\mathbf{s}} + (\Lambda_{Ps}^T\Lambda_{ss}^{-1}\bar{\mathbf{s}})^T \\ \hline \bar{P}\bar{\mathbf{s}} - \bar{\mathbf{s}}\Lambda_{Ps}^T\Lambda_{ss}^{-1}\bar{\mathbf{s}} + \Lambda_{ss}(\Lambda_{ss}^T)^{-1}\Lambda_{Ps} + \bar{\mathbf{s}}(\Lambda_{Ps}^T\Lambda_{ss}^{-1}\bar{\mathbf{s}})^T \end{array}\right] &\geqslant \left[\begin{array}{c} \bar{P} \\ \hline \Lambda_{Ps} + \bar{P}\bar{\mathbf{s}} \end{array}\right] \\
\left[\begin{array}{c} \bar{P} \\ \hline \bar{P}\bar{\mathbf{s}} + \Lambda_{Ps} \end{array}\right] &= \left[\begin{array}{c} \bar{P} \\ \hline \Lambda_{Ps} + \bar{P}\bar{\mathbf{s}} \end{array}\right]
\end{aligned}$$

where some basic algebraic properties have been used (i.e. $A^TB^T = (BA)^T$ and $(A^1)^T = (A^T)^-1$) and (i) Λ_{ss} is a symmetric matrix and (ii) $\Lambda_{Ps}^T\Lambda_{ss}^{-1}\bar{\mathbf{s}}$ is a scalar, i.e. $\Lambda_{Ps}^T\Lambda_{ss}^{-1}\bar{\mathbf{s}} = (\Lambda_{Ps}^T\Lambda_{ss}^{-1}\bar{\mathbf{s}})^T$.

B The Recursive Least Squares Algorithm

Let $\mathbf{\Phi}_N$ and $\mathbf{P}_N$ denote the feature and payoff matrix, respectively, after having observed N states, and given by

$$\mathbf{\Phi}_N = \begin{bmatrix} -\boldsymbol{\phi}_1^T- \\ \vdots \\ -\boldsymbol{\phi}_N^T- \end{bmatrix}, \qquad \mathbf{P}_N = \begin{bmatrix} P_1 \\ \vdots \\ P_N \end{bmatrix}. \tag{22}$$

The Recursive Least Squares (RLS) algorithm allows tracking the weight vector $\mathbf{w}_N$ that minimises the convex cost function

$$\sum_{t=1}^{N} \left(\mathbf{w}_N^T \boldsymbol{\phi}_t - P_t\right)^2 + \frac{1}{\delta^{\mathrm{RLS}}} \|\mathbf{w}_N\|^2, \tag{23}$$

and satisfies the equality

$$\left(\boldsymbol{\Phi}_N^T \boldsymbol{\Phi}_N + \frac{1}{\delta^{\mathrm{RLS}}} \mathbf{I}\right) \mathbf{w}_N = \boldsymbol{\Phi}_N^T \mathbf{P}_N, \tag{24}$$

where $\mathbf{I}$ denotes the identity matrix. Let $\mathbf{V}_N^{-1} = \boldsymbol{\Phi}^T \boldsymbol{\Phi}_N$ denote the feature autocorrelation matrix estimate, that satisfies the relation

$$\mathbf{V}_t^{-1} = \mathbf{V}_{t-1}^{-1} + \boldsymbol{\phi}_t^T \boldsymbol{\phi}_t, \tag{25}$$

with $\mathbf{V}_0 = \delta^{\mathrm{RLS}} \mathbf{I}$. Similarly, we have

$$\boldsymbol{\Phi}_t^T \mathbf{P}_t = \boldsymbol{\Phi}_{t-1}^T \mathbf{P}_{t-1} + \boldsymbol{\phi}_t P_t, \tag{26}$$

which, together with Eqs. 24 and 25 allows us to derive

$$\mathbf{V}_t^{-1} \mathbf{w}_t = \mathbf{V}_t^{-1} \mathbf{w}_{t-1} + \boldsymbol{\phi}_t (P_t - \mathbf{w}_{t-1}^T \boldsymbol{\phi}_t). \tag{27}$$

Pre-multiplying the above by $\mathbf{V}_t$ results in the RLS weight vector update

$$\mathbf{w}_t = \mathbf{w}_{t-1} + \mathbf{V}_t \boldsymbol{\phi}_t (P_t - \mathbf{w}_{t-1}^T \boldsymbol{\phi}_t). \tag{28}$$

To get the update for $\mathbf{V}$, we apply the Sherman-Morrison formula [**?**] to Eq. 25, resulting in

$$\mathbf{V}_t = \mathbf{V}_{t-1} - \frac{\mathbf{V}_{t-1} \boldsymbol{\phi}_t \boldsymbol{\phi}_t^T \mathbf{V}_{t-1}}{1 + \boldsymbol{\phi}_t^T \mathbf{V}_{t-1} \boldsymbol{\phi}}, \tag{29}$$

which can be written as

$$\beta^{\mathrm{RLS}} = 1 + \boldsymbol{\phi}_t^T \mathbf{V}_{t-1} \boldsymbol{\phi}_t, \tag{30}$$

$$\mathbf{V}_t = \mathbf{V}_{t-1} - \beta^{\mathrm{RLS}} \mathbf{V}_{t-1} \boldsymbol{\phi}_t \boldsymbol{\phi}_t^T \mathbf{V}_{t-1}, \tag{31}$$

and thus results in the final RLS update for $\mathbf{V}$. Note that the Sherman-Morrison formula is only applicable if $\mathbf{V}^{-1}$ is invertible, and thus $\mathbf{V}$ needs to be initialised to $\mathbf{V}_0 = \delta^{\mathrm{RLS}} \mathbf{I}$ with $\delta^{\mathrm{RLS}} < \infty$, such that $\mathbf{V}_0^{-1} = (1/\delta^{\mathrm{RLS}}) \mathbf{I} > 0 \mathbf{I}$.

C Incremental Mean Square Error Update

Let us assume that the weight vector $\mathbf{w}$ is estimated by the RLS algorithm, initialised with a very large $\delta^{\mathrm{RLS}} \to \infty$, and therefore by Eq. 24 at t satisfies the normal equation

$$\left(\boldsymbol{\Phi}_t^T \boldsymbol{\Phi}_t\right) \mathbf{w}_t = \boldsymbol{\Phi}_t^T \mathbf{P}_t, \tag{32}$$

which can also be written as

$$\mathbf{w}_t^T \mathbf{\Phi}_t^T \left(\mathbf{\Phi}_t \mathbf{w}_t - \mathbf{P}_t\right) = 0 \tag{33}$$

Our aim is to find an incremental update equation for the MSE, $f_t(\mathbf{W}_t)$, that, following Eq. 4, is in matrix notation given by

$$t f_t(\mathbf{w}_t) = \|\mathbf{\Phi}_t \mathbf{w}_t - \mathbf{P}_t\|^2, \tag{34}$$

Using $-\mathbf{P}_t = -\mathbf{\Phi}_t \mathbf{w}_t + (\mathbf{\Phi}_t \mathbf{w}_t - \mathbf{P}_t)$ and Eq. 33 allows us to derive

$$\begin{aligned} \mathbf{P}_t^T \mathbf{P}_t &= \mathbf{w}_t^T \mathbf{\Phi}_t^T \mathbf{\Phi}_t \mathbf{w}_t - 2\mathbf{w}_t^T \mathbf{\Phi}_t^T (\mathbf{\Phi}_t \mathbf{w}_t - \mathbf{P}_t) + (\mathbf{\Phi}_t \mathbf{w}_t - \mathbf{P}_t)^T (\mathbf{\Phi}_t \mathbf{w}_t - \mathbf{P}_t) \\ &= \mathbf{w}_t^T \mathbf{\Phi}_t^T \mathbf{\Phi}_t \mathbf{w}_t + \|\mathbf{\Phi}_t \mathbf{w}_t - \mathbf{P}_t\|^2, \end{aligned} \tag{35}$$

and thus we can express the sum of squared errors by

$$\|\mathbf{\Phi}_t \mathbf{w}_t - \mathbf{P}_t\|^2 = \mathbf{P}_t \mathbf{P}_t - \mathbf{w}_t^T \mathbf{\Phi}_t^T \mathbf{\Phi}_t \mathbf{w}_t. \tag{36}$$

To express $\|\mathbf{\Phi}_t \mathbf{w}_t - \mathbf{P}_t\|^2$ in terms of $\|\mathbf{\Phi}_{t-1} \mathbf{w}_{t-1} - \mathbf{P}_{t-1}\|^2$, we combine Eqs. 25, 26 and 36, and use $\mathbf{V}_t^{-1} \mathbf{w}_t = \mathbf{\Phi}_t^T \mathbf{P}_t$ after Eq. 32 to get

$$\begin{aligned} & \|\mathbf{\Phi}_t \mathbf{w}_t - \mathbf{P}_t\|^2 \\ &= \mathbf{P}_t^T \mathbf{P}_t - \mathbf{w}_t^T \mathbf{\Phi}_t^T \mathbf{\Phi}_t \mathbf{w}_t \\ &= \|\mathbf{\Phi}_{t-1} \mathbf{w}_{t-1} - \mathbf{P}_{t-1}\|^2 + P_t^2 + \mathbf{w}_{t-1}^T \mathbf{V}_{t-1}^{-1} \mathbf{w}_{t-1} - \mathbf{w}_t^T \mathbf{V}_t^{-1} \mathbf{w}_t \\ &= \|\mathbf{\Phi}_{t-1} \mathbf{w}_{t-1} - \mathbf{P}_{t-1}\|^2 + P_t^2 \\ &\quad + \mathbf{w}_{t-1}^T \left(\left(\mathbf{V}_{t-1}^{-1} + \boldsymbol{\phi}_t \boldsymbol{\phi}_t^T \right) \mathbf{w}_t - \boldsymbol{\phi}_t P_t \right) - \mathbf{w}_t^T \left(\mathbf{V}_{t-1}^{-1} \mathbf{w}_{t-1} + \boldsymbol{\phi}_t P_t \right) \\ &= \|\mathbf{\Phi}_{t-1} \mathbf{w}_{t-1} - \mathbf{P}_{t-1}\|^2 + P_t^2 + \mathbf{w}_{t-1}^T \boldsymbol{\phi}_t \boldsymbol{\phi}_t^T \mathbf{w}_t - \mathbf{w}_{t-1}^T \boldsymbol{\phi}_t P_t - \mathbf{w}_t^T \boldsymbol{\phi}_t P_t \\ &= \|\mathbf{\Phi}_{t-1} \mathbf{w}_{t-1} - \mathbf{P}_{t-1}\|^2 + (\mathbf{w}_{t-1}^T \boldsymbol{\phi}_t - P_t)(\mathbf{w}_t^T \boldsymbol{\phi}_t - P_t). \end{aligned}$$

Thus, we get

$$t f_t(\mathbf{w}_t) = (t-1) f_{t-1}(\mathbf{w}_{t-1}) + (\mathbf{w}_{t-1}^T \boldsymbol{\phi}_t - P_t)(\mathbf{w}_t^T \boldsymbol{\phi}_t - P_t), \tag{37}$$

which, using $\varepsilon_t^2 \equiv f_t(\mathbf{w}_t)$, can be rewritten to

$$\varepsilon_t^2 = \varepsilon_{t-1}^2 + \frac{1}{t} \left((\mathbf{w}_{t-1}^T \boldsymbol{\phi}_t - P_t)(\mathbf{w}_t^T \boldsymbol{\phi}_t - P_t) - \varepsilon_{t-1}^2 \right). \tag{38}$$

References

1. J. Drugowitsch and A. M. Barry. A formal framework and extensions for function approximation in learning classifier systems. Technical Report CSBU2006-01, Dept. Computer Science, University of Bath, 2006. ISSN 1740-9497, submitted in slightly modified form to Machine Learning, 06/03/2006.
2. S. A. Glantz and B. K. Slinker. *Primer of Applied Regression & Analysis of Variance*. McGraw Hill, 2001. second edition.

3. P. L. Lanzi, D. Loiacono, S. W. Wilson, and D. E. Goldberg. Extending XCSF beyond linear approximation. In *Genetic and Evolutionary Computation – GECCO-2005*, Washington DC, USA, 2005. ACM Press.
4. P. L. Lanzi, D. Loiacono, S. W. Wilson, and D. E. Goldberg. Generalization in the XCSF classifier system: Analysis, improvement, and extension. Technical Report 2005012, Illinois Genetic Algorithms Laboratory – University of Illinois at Urbana-Champaign, 2005. also available as a technical report of the Dipartimento di Elettronica e Informazione – Politecnico di Milano.
5. P. L. Lanzi, D. Loiacono, S. W. Wilson, and D. E. Goldberg. XCS with Computed Prediction for the Learning of Boolean Functions. In *Proceedings of the IEEE Congress on Evolutionary Computation – CEC-2005*, Edinburgh, UK, 2005. IEEE.
6. P. L. Lanzi, D. Loiacono, S. W. Wilson, and D. E. Goldberg. Prediction update algorithms for XCSF: RLS, kalman filter, and gain adaptation. Technical Report 2006008, Illinois Genetic Algorithms Laboratory – University of Illinois at Urbana-Champaign, 2006.
7. B. Widrow and M. E. Hoff. *Adaptive Switching Circuits*, chapter Neurocomputing: Foundation of Research, pages 126–134. The MIT Press, Cambridge, 1988.
8. S. W. Wilson. Mining Oblique Data with XCS. pages 158–174.
9. S. W. Wilson. Classifier Fitness Based on Accuracy. *Evolutionary Computation*, 3(2):149–175, 1995. http://prediction-dynamics.com/.
10. S. W. Wilson. Classifiers that approximate functions. *Journal of Natural Computating*, 1(2-3):211–234, 2002.