Analysis and Improvements of the Classifier Error Estimate in XCSF

Daniele Loiacono[1], Jan Drugowitsch[2], Alwyn Barry[2], and Pier Luca Lanzi[1,3]

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

Abstract. The estimation of the classifier error plays a key role in accuracy-based learning classifier systems. In this paper we study the current definition of the classifier error in XCSF and discuss the limitations of the algorithm that is currently used to compute the classifier error estimate from online experience. Subsequently, we introduce a new definition for the classifier error and apply the Bayes Linear Analysis framework to find a more accurate and reliable error estimate. This results in two incremental error estimate update algorithms that we compare empirically to the performance of the currently applied approach. Our results suggest that the new estimation algorithms can improve the generalization capabilities of XCSF, especially when the action-set subsumption operator is used.

1 Introduction

XCS with computed prediction, namely XCSF [12], is a major advance in the field of learning classifier systems. It extends the typical idea of a classifier by replacing the classifier prediction parameter with a prediction function $p(s_t, \mathbf{w})$, that is used to compute the classifier prediction based on the current state s_t and a parameter vector $\mathbf{w}$ associated with each classifier. Since the introduction of XCSF, several studies focused on the classifier weight vector update rule [7,6] and on extending the form of the prediction function p (e.g. see [8]). However, very little work (e.g. see [1]) has concentrated on the classifier error estimate in XCSF, despite its important role in all accuracy-based learning classifier systems. In XCSF the classifier error is usually computed in the same way as in XCS [11]: it is defined as the estimate of the mean absolute prediction error and is updated by the Widrow-Hoff rule (also known as *Least Mean Squared* algorithm). In this paper we suggest the re-definition the classifier error as an estimate of the root mean squared prediction error and propose application of the Bayes Linear Analysis framework for computing the optimal classifier weight vector and the

J. Bacardit et al. (Eds.): IWLCS 2006/2007, LNAI 4998, pp. 117–135, 2008.

classifier error estimate simultaneously. Within such a framework we provide an accurate and reliable estimate of the classifier error. At the same time, we can also provide a better insight into the relationship between the expected error and the statistical knowledge we have about the problem at hand. We prove that the proposed approach, when applied to updating the classifiers in XCSF, (i) computes exactly the same weight vector as the Least Squares update rule introduced in [7], and (ii) provides an unbiased estimate of the root mean squared prediction error. Additionally, by exploiting the similarities to Least Squares, we also introduce a convenient recursive approach for updating the classifiers that combines the Recursive Least Squares weights update [7] and the error tracking algorithm derived in [1]. Finally, we provide an empirical comparison of the classifier error update rules presented in this paper. The experiments have been performed by applying XCSF to the approximation of several commonly used target functions [4]. Our experimental results suggest that the novel classifier error update rules are able to find a more accurate and reliable estimate. In particular, they may improve the generalization capabilities of the XCSF system and allows for using the action-set subsumption operator while preventing overgeneral solutions from taking over the evolved population.

2 The XCSF Classifier System

When compared to XCS, XCSF replaces the classifier 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 parameter vector $\mathbf{w}$ that replaces the usual scalar prediction parameter; to keep the notation uncluttered, for the rest of this paper we denote $\boldsymbol{\phi}_t \equiv \phi(\mathbf{s}_t)$ as the feature vector that corresponds to the sensory input $\mathbf{s}_t$, and $p(\boldsymbol{\phi}_t) \equiv p(\boldsymbol{\phi}_t, \mathbf{w})$ as the classifier 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 $\mathbf{s}_t$, so 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], in this paper we will exclusively consider the just introduced linear function.

To update the classifiers, 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*, as 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_t is returned to the system together with the next input. When XCSF is used as a pure function approximator (like in [12] and in this paper), there is only one dummy action which has no actual effect and the expected payoff is computed

by $P_t = r_t$. The expected payoff P_t is then used to update the weight vector $\mathbf{w}$ of the classifiers in [A] using the Widrow-Hoff rule, also known as the *modified delta rule* [10]. The weight vector, $\mathbf{w}$, of each classifier in [A] is adjusted as follows:

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

where η is the correction rate and $\|\boldsymbol{\phi}_t\|^2$ is the squared Euclidean norm of the input vector $\boldsymbol{\phi}_t$ [12]. Then the prediction error, ε, is updated by

$$\varepsilon \leftarrow \varepsilon + \beta \left(|\mathbf{w}^T \boldsymbol{\phi}_t - P_t| - \varepsilon\right). \tag{2}$$

Finally, the classifier fitness is updated as usual [11] and the discovery component is applied as in XCS.

3 Squared Error or Absolute Error?

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 actually 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, after t classifier updates, the inputs $\{\mathbf{s}_i\}_{i=1}^t$ and their associated payoffs $\{P_i\}_{i=1}^t$, and that we want to estimate the classifier's weight vector $\mathbf{w}$ that minimises the MSE, given by

$$f_t(\mathbf{w}) = \frac{1}{t} \sum_{i=1}^{t} \left(\mathbf{w}^T \boldsymbol{\phi}_i - P_i\right)^2, \tag{3}$$

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

$$\mathbf{w}_t = \mathbf{w}_{t-1} + \frac{\eta \boldsymbol{\phi}_t}{\|\boldsymbol{\phi}_t\|^2} \left(P_t - \mathbf{w}_{t-1}^T \boldsymbol{\phi}_t\right), \tag{4}$$

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

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

$$\varepsilon_t = \frac{1}{t} \sum_{i=1}^{t} \left|\mathbf{w}^T \boldsymbol{\phi}_i - P_i\right|. \tag{5}$$

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

$$g_t^{\mathrm{MAE}}(\mathbf{w}) = \frac{1}{t}\sum_{i=1}^{t}\left(\varepsilon_t - \left|\mathbf{w}^T\boldsymbol{\phi}_i - P_i\right|\right)^2 \tag{6}$$

with respect to ε_t. Solving $\partial g_t^{\mathrm{MAE}}(\mathbf{w})/\partial\varepsilon_t = 0$ for ε_t results in Eq. 5, which confirms that we can indeed estimate ε_t by minimising $g_t^{\mathrm{MAE}}(\mathbf{w})$. Applying the delta rule (also known as the *Least Mean Squared* algorithm) to minimising $g_t^{\mathrm{MAE}}(\mathbf{w})$ results in the prediction error update

$$\varepsilon_t = \varepsilon_{t-1} + \beta\left(\left|\mathbf{w}_t^T\boldsymbol{\phi}_t - P_t\right| - \varepsilon_{t-1}\right), \tag{7}$$

where we have approximated the weight vector by its current estimate $\mathbf{w} \approx \mathbf{w}_t$. This update equation is equivalent to Eq. 2, 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 [8]), 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. 3) in the same way as the MAE (Eq. 5) by reformulating its estimation as a least squares problem that minimises

$$g_t^{\mathrm{MSE}}(\mathbf{w}) = \frac{1}{t}\sum_{i=1}^{t}\left(\varepsilon_t^2 - (\mathbf{w}^T\boldsymbol{\phi}_i - P_i)^2\right)^2 \tag{8}$$

with respect to ε_t^2, which denotes the estimate of the classifier squared error at time t. Applying the delta rule by again approximating $\mathbf{w}$ by its estimate $\mathbf{w}_t$ gives the update equation

$$\varepsilon_t^2 = \varepsilon_{t-1}^2 + \beta\left((\mathbf{w}_t^T\boldsymbol{\phi}_t - P_t)^2 - \varepsilon_{t-1}^2\right), \tag{9}$$

from which we compute the classifier error by

$$\varepsilon_t = \sqrt{\varepsilon_t^2}. \tag{10}$$

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 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. 9, but can track the solution to $f_t(\mathbf{w})$ directly, as we will show in the following section.

4 Improving the Error Estimate

In previous studies [6,1] the problem of computing the classifier prediction has been presented as a problem of incremental parameter estimation. In the following sections we show that both the optimal classifier weights and the estimate of the classifier error can be computed by applying the Bayes Linear Analysis [3] framework. Within this framework we are not only able to provide a very accurate and reliable estimate of a classifier's squared error but also give additional insight into the relationship between the expected prediction error and the statistical properties of the target payoff and the feature vector. Furthermore, we show how the proposed theoretical framework can be used in practice to extend the classifier update in XCSF. Finally, we discuss how the proposed extension is related to the least squares approach introduced in the literature for computing the classifier weights [7] and for incrementally estimating the classifier error [1].

4.1 The Bayes Linear Analysis

In XCSF, we need to predict, with the highest accuracy possible, the value of the target payoff P on the basis of the observed features vector $\boldsymbol{\phi}$. Assuming a full knowledge on the probability distribution of both P and $\boldsymbol{\phi}$ we might derive the conditional density $p(P|\boldsymbol{\phi})$ of P given $\boldsymbol{\phi}$ and thus compute the classifier prediction as the conditional expectation $E[P|\boldsymbol{\phi}]$. Unfortunately, we usually do not have such knowledge and therefore cannot derive the conditional probability distribution. In XCSF, however, we limit our search for a suitable prediction model to the linear function $\mathbf{w}^T\boldsymbol{\phi}$ (see Section 2). This assumption allows us to apply Bayes Linear Analysis [3] to compute the classifier weights and errors. Accordingly, the classifier weight vector $\mathbf{w}$ is considered optimal if it solves the following minimisation problem,

$$\min_{\mathbf{w}} E[(P - \mathbf{w}^T\boldsymbol{\phi}(\mathbf{s}))^2], \tag{11}$$

which corresponds (see Appendix A for the derivation) to the classifier weight vector $\mathbf{w}$, defined as,

$$\mathbf{w} = E\left[\boldsymbol{\phi}\boldsymbol{\phi}^T\right]^{-1} E\left[P\boldsymbol{\phi}\right]. \tag{12}$$

By substituting $\mathbf{w}$ by Eq. 12 into the minimisation objective Eq. 11, we get the following classifier squared prediction error estimate (see Appendix A for the derivation):

$$\varepsilon^2 = E\left[(P - \mathbf{w}^T\boldsymbol{\phi})^2\right] = E\left[P^2\right] - E\left[P\boldsymbol{\phi}^T\right] E\left[\boldsymbol{\phi}\boldsymbol{\phi}^T\right]^{-1} E\left[P\boldsymbol{\phi}\right]. \tag{13}$$

Before showing how Eqs. 12 and 13 can be used in practice to update the classifiers in XCSF, it is worthwhile to discuss in more details the consequences of Eq. 13. First of all, given that in XCSF we have $\boldsymbol{\phi}^T = [x_0 \quad \mathbf{s}^T]$, Eq. 13 can be rewritten (see Appendix A for the derivation) as follows,

$$\varepsilon^2 = \mathrm{cov}(P, P)(1 - \rho^2), \tag{14}$$

where $\mathrm{cov}(P, P) = E\left[(P - E\left[P\right])^2\right]$ and ρ^2 is the squared correlation coefficient between P and $\mathbf{s}$, given by,

$$\rho^2 = \frac{\mathrm{cov}(P, \mathbf{s})^T \mathrm{cov}(\mathbf{s}, \mathbf{s})^{-1} \mathrm{cov}(P, \mathbf{s})}{\mathrm{cov}(P, P)} \tag{15}$$

where we have

$$\begin{aligned} \mathrm{cov}(P, \mathbf{s}) &= E\left[(P - E\left[P\right])(\mathbf{s} - E\left[\mathbf{s}\right])\right], \\ \mathrm{cov}(\mathbf{s}, \mathbf{s}) &= E\left[(\mathbf{s} - E\left[\mathbf{s}\right])(\mathbf{s} - E\left[\mathbf{s}\right])^T\right]. \end{aligned}$$

Equation 14 offers an interesting insight on the expected classifier prediction error. When P and $\mathbf{s}$ are completely uncorrelated, i.e. $\rho^2 = 0$, it is not possible to provide 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 maximally correlated, i.e. $\rho^2 = 1$, the target payoff can be predicted without error; therefore the expected square prediction error is equal to 0. In all the other cases, the higher 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 Its Relation to Least Squares

So far we have assumed knowledge of $E\left[PP\right]$, $E\left[P\boldsymbol{\phi}\right]$ and $E\left[\boldsymbol{\phi}\boldsymbol{\phi}\right]$. Unfortunately such knowledge is not available and thus we cannot directly use Eqs. 12 and 13 in XCSF. For this reason we propose to replace the true expectations with their sample-based estimators, computed at each time step t as,

$$E_{PP} \approx \hat{E}_{PP,t} = \frac{1}{t}\sum_{i=1}^{t} P_i^2 = \hat{E}_{PP,t-1} + \frac{1}{t}(P_t^2 - \hat{E}_{PP,t-1}), \tag{16}$$

$$E_{\boldsymbol{\phi}\boldsymbol{\phi}} \approx \hat{E}_{\boldsymbol{\phi}\boldsymbol{\phi},t} = \frac{1}{t}\sum_{i=1}^{t} \boldsymbol{\phi}_i\boldsymbol{\phi}_i^T = \hat{E}_{\boldsymbol{\phi}\boldsymbol{\phi},t-1} + \frac{1}{t}(\boldsymbol{\phi}_t\boldsymbol{\phi}_t^T - \hat{E}_{\boldsymbol{\phi}\boldsymbol{\phi},t-1}), \tag{17}$$

$$E_{P\boldsymbol{\phi}} \approx \hat{E}_{P\boldsymbol{\phi},t} = \frac{1}{t}\sum_{i=1}^{t} P_i\boldsymbol{\phi}_i = \hat{E}_{P\boldsymbol{\phi},t-1} + \frac{1}{t}(P_i\boldsymbol{\phi}_t - \hat{E}_{P\boldsymbol{\phi},t-1}). \tag{18}$$

Using the above approximations in Eqs. 12 and 13, we obtain the following update rules that can be used for computing the classifier weights and error in XCSF:

$$\mathbf{w}_t = \hat{E}^{-1}_{\boldsymbol{\phi}\boldsymbol{\phi},t}\hat{E}_{P\boldsymbol{\phi},t}, \tag{19}$$

$$\varepsilon_t^2 = \hat{E}_{PP,t} - \hat{E}^T_{P\boldsymbol{\phi},t}\hat{E}^{-1}_{\boldsymbol{\phi}\boldsymbol{\phi},t}\hat{E}^T_{P\boldsymbol{\phi},t}. \tag{20}$$

Notice that the above update rules are more accurate than the usual Widrow-Hoff rule (Eqs. 1 and 9) in finding the optimal classifier weights and error estimates. On the downside, they are computationally more expensive: due to the computation of matrix $\hat{E}^{-1}_{\boldsymbol{\phi}\boldsymbol{\phi}}$ both update rules have a time complexity of $O(n^3)$, whilst the Widrow-Hoff has a time complexity of $O(n)$. Furthermore, it is necessary to store, for each classifier, the sample-based estimators used in Eqs. 19 and 20 with an additionally memory overhead of $O(n^2)$.

To reduce the time complexity of the above update equations it is useful to note that, using the sample-based estimators introduced before, it can be shown (see Appendix B) that Eq. 12 is equivalent to the Least Squares update rule that was introduced in [7] for computing the classifier weights. Additionally, the classifier error computed by Eq. 20 is equivalent to the sample-based estimator of the mean square prediction error (see Appendix B), given by $f_t(\mathbf{w})$ in Eq. 3. In the following section we show how this knowledge can be exploited to derive more efficient update equations.

4.3 Recursive Least Squares and Error Tracking

We have shown that by applying Bayes Linear Analysis we can effectively compute both the classifier weight vector and the classifier error. Unfortunately, as already mentioned before, Eqs. 19 and 20 are computationally expensive. This is a serious drawback because in XCSF the classifiers are updated incrementally and frequently. However this is a well known limitation of the Least Squares update, that is computed by Eq. 19. Thus, following the same procedure as in [7], we can instead use the less costly Recursive Least Squares algorithms (see Appendix C for more details) to incrementally update the classifier weights by

$$\beta^{\mathrm{RLS}} = 1 + \boldsymbol{\phi}_t^T\mathbf{V}_{t-1}\boldsymbol{\phi}_t, \tag{21}$$

$$\mathbf{V}_t = \mathbf{V}_{t-1} - \frac{1}{\beta^{\mathrm{RLS}}}\mathbf{V}_{t-1}\boldsymbol{\phi}_t^T\boldsymbol{\phi}_t\mathbf{V}_{t-1}, \tag{22}$$

$$\mathbf{w}_t = \mathbf{w}_{t-1} + \mathbf{V}_t\boldsymbol{\phi}_t(P - \mathbf{w}_{t-1}^T\boldsymbol{\phi}_t), \tag{23}$$

where $\mathbf{V}_t$ is an estimate of the feature vector autocorrelation matrix. Note that the above update equations avoid the computation of the inverse matrix at each time and therefore have the lower computational complexity of $O(n^2)$. On the other hand, each classifier still need to store the actual $\mathbf{V}$ matrix with an additional memory overhead of $O(n^2)$.

Table 1. Target functions used to compare the performance of XCSFrls, XCSFb and XCSFrb ($x \in [0, 1]$)

$$
\begin{aligned}
f_p(x) &= 1 + x + x^2 + x^3, \\
f_{abs}(x) &= |\sin(2\pi x) + |\cos(2\pi x)|| \,, \\
f_{s3}(x) &= \sin(2\pi x) + \sin(4\pi x) + \sin(6\pi x), \\
f_{s4}(x) &= \sin(2\pi x) + \sin(4\pi x) + \sin(6\pi x) + sin(8\pi x).
\end{aligned}
$$

Regarding the classifier error update, the solution to Eq. 3 can be tracked by using the following recursive update (see [1] and Appendix D for more details):

$$\varepsilon_t^2 = \varepsilon_{t-1}^2 + \frac{1}{t}\left((P_t - \mathbf{w}_{t-1}^T \boldsymbol{\phi}_t)(P_t - \mathbf{w}_t^T \boldsymbol{\phi}_t) - \varepsilon_{t-1}^2\right), \tag{24}$$

where $\mathbf{w}_{t-1}$ and $\mathbf{w}_t$ are respectively the classifier weight vectors before and after the update. Note that Eq. 24 is as accurate as Eq. 13 in tracking the classifier's squared prediction error estimate, but has the same complexity as Eq. 9, that is $O(n)$ in time.

5 Experimental Design

All the experiments discussed in this paper aim at comparing the performance of the different classifier error updates introduced in the previous sections. For this purpose we use three different versions of XCSF: (i) XCS with RLS prediction, briefly XCSFrls, that uses RLS (Eqs. 21, 22 and 23) to update classifier weights, and the commonly applied Widrow-Hoff (Eq. 9) to update the classifier error estimate; (ii) XCS with Bayes Linear prediction, briefly XCSFb that updates the classifier weight and error estimate by Eqs. 19 and 20; (iii) XCS with recursive Bayes Linear prediction, briefly XCSFrb, that applies the RLS algorithm (as XCSFrls) to update the classifier weights, and uses Eq. 24 to track the classifier error estimate. Note that in all the XCSF variants the classifier error is defined as an estimate of the RMSE of the prediction, for the reasons discussed in Section 3.

The experimental analysis has been performed on several function approximation tasks, following the standard design used in the literature [12]. As target functions we used the four functions reported in Table 1 that are a real valued version of the ones used in [4].

In all the experiments performed in this work the feature vector is defined as $\boldsymbol{\phi} = [x_0 \quad x]^T$, with x_0 set to 1. The performance is measured as the accuracy of the evolved approximation $\hat{f}(x)$ with respect to the target function $f(x)$, evaluated, in each experiment, as the *root mean square error* (RMSE) given by

$$RMSE = \sqrt{\frac{1}{N}\sum_{i=1}^{N}(f(x_i) - \hat{f}(x_i))^2},$$

where $\{x_1, \cdots, x_N\}$ are the N input samples used to evaluate the approximation error. In practice we considered the average RMSE, dubbed $\overline{RMSE}$, over all

experimental runs. To measure the generalization capabilities we considered both the number of macroclassifiers evolved and the fitness-weighted generality of the evolved populations, where each classifier generality is computed as the expected fraction of inputs matched according to [4].

Statistical Analysis. To analyze the results reported in this paper, we performed an analysis of variance (ANOVA) [2] on the resulting performances, evolved population size and generality. For each experiment and for each setting, we analyzed the final performance, the number of macroclassifiers evolved, and their fitness-weighted average generality for the different versions of XCSF; we applied the 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

The experimental analysis is organized as follows. At first we study the different update rules using a single classifiers for approximating a target function. Then we compare XCSFrls, XCSFrb, and XCSFb on several function approximation problems without using the action-set subsumption operator (Section 6.2) and using it (Section 6.3).

6.1 Single Classifier Error

In the first experiment we compare the classifier error updates in XCSFrls, XCSFb and in XCSFrb. For this purpose we focus on the error updates of a single classifier approximating f_{abs} for $x \in [0.4, 0.6]$. Figure 1 shows the classifier error estimate of a single run of the same classifier, as computed by XCSFrls, XCSFb, and XCSFrb when applied to f_{abs}. As reference, we also report the true error of the classifier, computed at the end of the experiment. Note that, although all the three error estimates are about the same on the average, the estimate computed by XCSFrls is very noisy and therefore not particularly reliable. On the other hand, both XCSFb and XCSFrb compute a very reliable and accurate classifier error estimate. Also, the estimates of XCSFrb and XCSFb initially differ slightly due to the bias induced by the initialization of $\mathbf{V}$ in XCSFrb (see Appendices C and D), but they converge very quickly to the same estimate. In conclusion, notice that the reliability of the error estimate of XCSFrls might be improved using a smaller value of the learing rate β; on the other hand, the smaller β the slower the convergence of the error estimate toward the true error. In all the experiments in the rest of the paper we always set $\beta = 0.2$ because tuning the value of β is tricky and the best value is, in general, problem dependent.

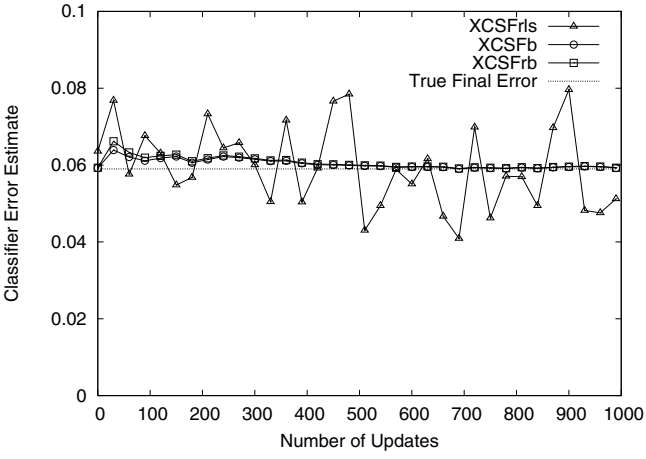

Fig. 1. Comparison of the error updates of a single run of XCSFrls, XCSFb, and XCSFrb applied to f_{abs}. The reported error estimates are that of a single classifier that matches all the inputs in the range $[0.4, 0.6]$.

6.2 Analysis of Generalization

In the second set of experiments we apply XCSFrls, XCSFb, and XCSFrb to all four functions in Table 1, using the following parameters setting: $N = 400$; $\beta = 0.2$; $\alpha = 0.1$; $\nu = 5$; $\chi = 0.8$, $\mu = 0.04$, $\theta_{del} = 25$, $\theta_{GA} = 25$, and $\delta = 0.1$; GA-subsumption is on with $\theta_{GAsub} = 25$; action-set subsumption is not used; $m_0 = 0.2$, $r_0 = 0.1$ [12]; in XCSFrls and XCSFrb we set $\delta_{rls} = 10000$ [7]; the value of ϵ_0 is set to either 0.05, 0.1 or 0.2. Table 2 reports the performance of XCSFrls, XCSFb, and XCSFrb, measured by the average RMSE of the evolved solutions

Table 2. Performance of XCSFrls, XCSFb, and XCSFrb applied to f_p, f_{s3}, f_{s4}, and f_{abs}. The action-set subsumption is not used. Statistics are averages over 50 runs.

$f(x)$	$\epsilon 0$	XCSFrls	XCSFrb	XCSFb
f_p	0.05	0.0111 ± 0.0014	0.0173 ± 0.0024	0.0172 ± 0.0027
f_p	0.10	0.0206 ± 0.0026	0.0289 ± 0.0032	0.0300 ± 0.0032
f_p	0.20	0.0543 ± 0.0070	0.0869 ± 0.0125	0.0869 ± 0.0125
f_{s3}	0.05	0.0300 ± 0.0083	0.0353 ± 0.0027	0.0347 ± 0.0025
f_{s3}	0.10	0.0510 ± 0.0045	0.0633 ± 0.0043	0.0618 ± 0.0032
f_{s3}	0.20	0.0831 ± 0.0075	0.1013 ± 0.0070	0.1024 ± 0.0080
f_{s4}	0.05	0.0321 ± 0.0056	0.0406 ± 0.0066	0.0387 ± 0.0050
f_{s4}	0.10	0.0547 ± 0.0048	0.0676 ± 0.0061	0.0669 ± 0.0041
f_{s4}	0.20	0.0863 ± 0.0069	0.1105 ± 0.0070	0.1132 ± 0.0084
f_{abs}	0.05	0.0190 ± 0.0020	0.0242 ± 0.0025	0.0243 ± 0.0026
f_{abs}	0.10	0.0349 ± 0.0027	0.0482 ± 0.0034	0.0481 ± 0.0031
f_{abs}	0.20	0.0867 ± 0.0048	0.1191 ± 0.0051	0.1208 ± 0.0052

over 50 runs. The results show that all three XCSF versions are accurate in that the final approximation error is lower than ϵ_0. We can also see that the error of XCSFrls is generally lower than the one of XCSFb and XCSFrb. This comes hand in hand both with a larger evolved population and with a lower average generality of the classifiers, as shown in Tables 3 and 4. These results are not surprising as a more reliable classifier error estimate can be expected to improve the classifier system's generalization capabilities, as confirmed by the outcome of this experiment.

The statistical analysis of the data reported in Tables 2, 3 and 4 reveals that the differences between XCSFrb and XCSFb are always *not significant* with a 99.9% confidence. A further analysis of Table 2 shows that the differences between XCSFrls, XCSFrb, and XCSFb are *significant* with a 99.9% confidence for almost all of the experiments, with the exception of cases where complex function are to be estimated with a low error, which prohibits generalization (for example, f_{s3} and f_{s4} with $\epsilon_0 = 0.05$). With respect to the population size (given in Table 3) the statistical analysis indicates that the differences are not always significant, especially on the most complex functions f_{s3} and f_{s4}. Concerning the fitness-weighted average generality, as reported in Table 4, the post-hoc analysis shows that the differences between XCSFrb and XCSFb are always *not significant* with a 99.9% confidence level, while the differences between XCSFb, XCSFrb, and XCSFrls are always significant.

In summary, the results suggest that improving the accuracy and the reliability of the classifier error estimate with the approaches introduced in Sections 4.2 and 4.3 allows XCSF to evolve more general and slightly more compact solutions (even if the size of the populations evolved by XCSFb and XCSFrb are not always significantly smaller than the ones evolved by XCSFrls).

Table 3. Average number of macroclassifiers evolved by XCSFrls, XCSFb, and XCSFrb applied to f_p, f_{s3}, f_{s4}, and f_{abs}. The action-set subsumption is not used. Statistics are averages over 50 runs.

$f(x)$	$\epsilon 0$	XCSFrls	XCSFrb	XCSFb
f_p	0.05	37.6200 ± 3.4922	33.0400 ± 3.6876	32.8200 ± 2.9509
f_p	0.10	33.6000 ± 3.3226	29.5000 ± 2.9883	29.1400 ± 3.0200
f_p	0.20	29.3600 ± 3.2970	27.0800 ± 3.2609	27.0800 ± 3.2609
f_{s3}	0.05	52.7600 ± 4.3523	52.0400 ± 4.1807	51.3200 ± 4.7516
f_{s3}	0.10	48.9400 ± 4.2020	45.9400 ± 4.1493	47.5800 ± 4.9883
f_{s3}	0.20	45.2800 ± 3.6989	42.5000 ± 3.1953	42.8000 ± 3.0331
f_{s4}	0.05	54.4200 ± 4.6047	52.8800 ± 4.4973	54.0000 ± 5.0794
f_{s4}	0.10	52.5800 ± 4.7248	50.0800 ± 4.0686	50.0800 ± 4.8820
f_{s4}	0.20	50.0400 ± 4.0594	47.1000 ± 3.6455	47.9200 ± 3.8773
f_{abs}	0.05	45.3600 ± 3.9180	44.1000 ± 4.4419	42.9000 ± 4.0062
f_{abs}	0.10	44.6800 ± 2.7162	41.6800 ± 3.8494	41.6800 ± 3.3849
f_{abs}	0.20	40.7800 ± 3.9104	35.6400 ± 3.3031	35.8600 ± 3.5497

Table 4. Average generality of classifiers evolved by XCSFrls, XCSFb, and XCSFrb applied to f_p, f_{s3}, f_{s4}, and f_{abs}. The action-set subsumption is not used. Statistics are averages over 50 runs.

$f(x)$	$\epsilon 0$	XCSFrls	XCSFrb	XCSFb
f_p	0.05	0.2916 ± 0.1462	0.3272 ± 0.1693	0.3265 ± 0.1719
f_p	0.10	0.3389 ± 0.2106	0.3614 ± 0.2418	0.3649 ± 0.2430
f_p	0.20	0.3951 ± 0.2928	0.4361 ± 0.3288	0.4361 ± 0.3288
f_{s3}	0.05	0.0719 ± 0.0354	0.0819 ± 0.0405	0.0824 ± 0.0404
f_{s3}	0.10	0.1032 ± 0.0452	0.1155 ± 0.0476	0.1145 ± 0.0478
f_{s3}	0.20	0.1342 ± 0.0503	0.1454 ± 0.0570	0.1457 ± 0.0576
f_{s4}	0.05	0.0556 ± 0.0294	0.0635 ± 0.0329	0.0634 ± 0.0329
f_{s4}	0.10	0.0797 ± 0.0369	0.0892 ± 0.0387	0.0898 ± 0.0384
f_{s4}	0.20	0.1039 ± 0.0430	0.1172 ± 0.0458	0.1170 ± 0.0449
f_{abs}	0.05	0.1255 ± 0.0303	0.1340 ± 0.0363	0.1345 ± 0.0353
f_{abs}	0.10	0.1498 ± 0.0520	0.1637 ± 0.0619	0.1635 ± 0.0622
f_{abs}	0.20	0.2052 ± 0.1264	0.2480 ± 0.1445	0.2518 ± 0.1437

6.3 Classifier Error and Action-Set Subsumption

The action-set subsumption operator [11] is a powerful mechanism to improve the generalization capabilities of XCS. In practice, action-set subsumption is rarely used in XCSF [12,7]. In fact, the action-set subsumption relies heavily on the correctness of the classifier error estimate in order to identify accurate classifiers, and in XCSF this can easily result in evolving overgeneral solutions. This is mainly due to the noisy classifier error estimate computed by XCSF as shown in Figure 1. Thus,

Table 5. Performance of XCSFrls, XCSFb, and XCSFrb applied to f_p, f_{s3}, f_{s4}, and f_{abs}. The action-set subsumption is used with $\theta_{ASsub} = 25$. Statistics are averages over 50 runs.

$f(x)$	$\epsilon 0$	XCSFrls	XCSFrb	XCSFb
f_p	0.05	0.0655 ± 0.0080	0.0419 ± 0.0051	0.0429 ± 0.0049
f_p	0.10	0.1799 ± 0.0078	0.0834 ± 0.0134	0.0816 ± 0.0129
f_p	0.20	0.1863 ± 0.0001	0.1863 ± 0.0001	0.1862 ± 0.0001
f_{s3}	0.05	0.0550 ± 0.0107	0.0461 ± 0.0170	0.0495 ± 0.0295
f_{s3}	0.10	0.1027 ± 0.0230	0.0771 ± 0.0071	0.0794 ± 0.0092
f_{s3}	0.20	0.2314 ± 0.0297	0.1451 ± 0.0112	0.1439 ± 0.0100
f_{s4}	0.05	0.0609 ± 0.0221	0.0512 ± 0.0217	0.0470 ± 0.0109
f_{s4}	0.10	0.1024 ± 0.0108	0.0801 ± 0.0090	0.0844 ± 0.0335
f_{s4}	0.20	0.2246 ± 0.0277	0.1493 ± 0.0116	0.1482 ± 0.0120
f_{abs}	0.05	0.0527 ± 0.0074	0.0361 ± 0.0060	0.0357 ± 0.0046
f_{abs}	0.10	0.1343 ± 0.0235	0.0859 ± 0.0075	0.0825 ± 0.0076
f_{abs}	0.20	0.2899 ± 0.0002	0.1725 ± 0.0279	0.1661 ± 0.0182

Table 6. Average number of macroclassifiers evolved by XCSFrls, XCSFb, and XCSFrb applied to f_p, f_{s3}, f_{s4}, and f_{abs}. The action-set subsumption is used $\theta_{ASsub} = 25$. Statistics are averages over 50 runs.

$f(x)$	$\epsilon 0$	XCSFrls	XCSFrb	XCSFb
f_p	0.05	10.7800 ± 2.9277	7.8600 ± 2.5535	7.3800 ± 2.6449
f_p	0.10	4.5200 ± 1.8027	7.8400 ± 3.7966	7.9000 ± 2.9682
f_p	0.20	2.6000 ± 1.2329	2.1800 ± 1.0713	2.5200 ± 1.2528
f_{s3}	0.05	27.6400 ± 3.5820	29.8600 ± 3.7041	30.4000 ± 3.8158
f_{s3}	0.10	20.5200 ± 4.2391	20.5400 ± 3.5565	20.7800 ± 3.0678
f_{s3}	0.20	16.2800 ± 3.7151	14.6800 ± 3.5068	13.3600 ± 3.0382
f_{s4}	0.05	33.1000 ± 4.1049	38.1800 ± 3.4333	38.2800 ± 5.1109
f_{s4}	0.10	25.9400 ± 4.0812	27.5200 ± 4.1916	27.1600 ± 3.9767
f_{s4}	0.20	20.4000 ± 4.8497	17.7200 ± 2.9465	19.5200 ± 2.9205
f_{abs}	0.05	16.4800 ± 3.3301	16.3400 ± 3.4328	16.5000 ± 2.7514
f_{abs}	0.10	12.3600 ± 2.8549	15.5200 ± 4.4777	15.5800 ± 3.4761
f_{abs}	0.20	2.6600 ± 1.3800	8.8800 ± 3.3205	10.0400 ± 3.9036

Table 7. Average generality of classifiers evolved by XCSFrls, XCSFb, and XCSFrb applied to f_p, f_{s3}, f_{s4}, and f_{abs}. The action-set subsumption is used with $\theta_{ASsub} = 25$. Statistics are averages over 50 runs.

$f(x)$	$\epsilon 0$	XCSFrls	XCSFrb	XCSFb
f_p	0.05	0.6833 ± 0.0820	0.5109 ± 0.0896	0.5149 ± 0.0846
f_p	0.10	0.9861 ± 0.0739	0.7375 ± 0.0846	0.7301 ± 0.0847
f_p	0.20	0.9960 ± 0.0635	0.9971 ± 0.0542	0.9961 ± 0.0619
f_{s3}	0.05	0.0987 ± 0.0487	0.0842 ± 0.0403	0.0843 ± 0.0406
f_{s3}	0.10	0.1437 ± 0.0619	0.1231 ± 0.0506	0.1216 ± 0.0517
f_{s3}	0.20	0.2202 ± 0.1164	0.1732 ± 0.0650	0.1776 ± 0.0638
f_{s4}	0.05	0.0759 ± 0.0398	0.0637 ± 0.0331	0.0638 ± 0.0335
f_{s4}	0.10	0.1094 ± 0.0500	0.0927 ± 0.0413	0.0914 ± 0.0411
f_{s4}	0.20	0.1631 ± 0.0845	0.1338 ± 0.0510	0.1331 ± 0.0517
f_{abs}	0.05	0.1772 ± 0.0465	0.1506 ± 0.0339	0.1491 ± 0.0348
f_{abs}	0.10	0.3034 ± 0.1619	0.2294 ± 0.1049	0.2258 ± 0.1021
f_{abs}	0.20	0.9959 ± 0.0639	0.3469 ± 0.1839	0.3377 ± 0.1683

in the last set of experiments we test whether the new classifier error updates can improve the performance of XCSF when action-set subsumption is used.

We again apply XCSFrls, XCSFb, and XCSFrb to the four functions in Table 1, using the same parameters setting as in the previous experiment, except for the action-set subsumption that is now active with $\theta_{ASsub} = 25$. The performance of XCSFrls, XCSFb, and XCSFrb is reported in Table 5 computed as the average RMSE of the evolved solutions over 50 runs. The results show that XCSFb and XCSFrb are always able to evolve accurate solutions while the solutions evolved by XCSFrls are never accurate except for the simplest function,

f_p, with the highest error threshold, $\epsilon_0 = 0.2$, that allows the evolution of a completely general solution. As we expected, the results suggest that in XCSFrls the action-set subsumption operator may have disruptive effects on the evolved population by considering overgeneral classifiers to be accurate. On the other hand, the more reliable error estimates used in XCSFrb and in XCSFb avoid such a problem. The statistical analysis of the data reported in Table 5 reveals that the differences between XCSFrb and XCSFb are always *not significant* with a 99.9% confidence. A further analysis of Table 5 shows that the differences between XCSFrls and the variants XCSFrb and XCSFb are *significant* with a 99.9% confidence for all the experiments except when the function is of low complexity and generalization is straightforward (e.g. f_p with $\epsilon_0 = 0.2$).

The analysis of the size and generality makes sense only if the the evolved population is accurate. For this reason we have only analyzed the results of XCSFrb and XCSFb, as XCSFrls is almost never accurate. The statistical analysis of the data reported in Table 6 shows that the differences between XCSFrb and XCSFb are always *not significant* with a 99.9% confidence. On the other hand, the same analysis applied to the data in Table 7 shows that XCSFrb evolves slightly more general populations than XCSFb and this difference is significant for most of the experiments. In addition, a comparison with the data reported in the previous section (Tables 3 and 4), shows that by using the action-set subsumption operator it is possible to evolve a more compact and general population (differences are always *significant* with a 99.9% confidence), confirming the results obtained by applying the action-set subsumption to XCS [11].

In summary, the experimental results confirm our hypotheses: the classifier error updates used in XCSFb and in XCSFrb offer a more reliable estimate and therefore allow the action-set subsumption to perform as intended. In fact, the populations evolved by XCSFb and XCSFrb are always accurate and they are also significantly smaller and more general than the ones evolved without using action-set subsumption.

7 Conclusions

In this paper we have proposed a new classifier error definition that is not only more consistent with the XCSF performance component but can also be estimated more effectively. For this purpose, we have introduced the Bayes Linear Analysis framework to compute both the optimal classifier weight vector and the classifier error. In particular, within this framework, we have provided an insight into the relationship between the expected classifier error and the statistical properties of the problem variables, that is, the target payoff and the input vector. Additionally, we have provided two update rules for updating the classifier error more accurately. We have also discussed the similarities between the proposed approach and the Least Squares one that was successfully applied to extending XCSF in [7]. Finally, the classifier error update rules presented in this paper have been empirically compared on several function approximation tasks. Our results suggest that the new error updates do not only compute a more

reliable and accurate estimate, but are also able to improve the performance and the generalization capabilities of XCSF. In particular, the new error update rules (i) allow XCSF to evolve a more compact and general population and (ii) prevent XCSF from evolving inaccurate overgeneral approximations when the action-set subsumption operator is used. On the other hand, improving the error estimate with the usual Widrow-Hoff rule requires the tuning of the learning rate parameters and may significantly slow down the error estimate convergence. However, it is still not clear whether a slower convergence may affect the performance in more complex problems than the ones considered here.

In conclusion, it is important to say that the update rules introduced in this paper have been derived assuming that all the past experiences collected by the system are equally important for solving the problem. Unfortunately this does not hold in multistep problems, where recent experience is usually more important. Therefore, the approach introduced here needs to be extended for multistep problems, possibly with some mechanism of recency-weighting of the collected experience.

References

1. Drugowitsch, J., Barry, A.: A formal framework and extensions for function approximation in learning classifier systems. Machine Learning 70(1), 45–88 (2008)
2. Glantz, S.A., Slinker, B.K.: Primer of Applied Regression & Analysis of Variance, 2nd edn. McGraw Hill, New York (2001)
3. Goldstein, M.: Bayes linear analysis. In: Kotz, S., Read, C.B., Banks, D.L. (eds.) Encyclopedia of Statistical Sciences, vol. 3, pp. 29–34. Wiley, New York (1999)
4. Lanzi, P.L., Loiacono, D., Wilson, S.W., Goldberg, D.E.: Extending XCSF beyond linear approximation. In: Genetic and Evolutionary Computation – GECCO-2005. ACM Press, Washington (2005)
5. Lanzi, P.L., Loiacono, D., Wilson, S.W., Goldberg, D.E.: XCS with Computed Prediction for the Learning of Boolean Functions. In: Proceedings of the IEEE Congress on Evolutionary Computation – CEC-2005, Edinburgh, UK. IEEE Computer Society Press, Los Alamitos (2005)
6. Lanzi, P.L., Loiacono, D., Wilson, S.W., Goldberg, D.E.: Prediction update algorithms for XCSF: RLS, kalman filter, and gain adaptation. In: GECCO 2006: Proceedings of the 8th annual conference on Genetic and evolutionary computation, pp. 1505–1512. ACM Press, New York (2006)
7. Lanzi, P.L., Loiacono, D., Wilson, S.W., Goldberg, D.E.: Generalization in the XCSF classifier system: Analysis, improvement, and extension. Evolutionary Computation 15(2), 133–168 (2007)
8. Loiacono, D., Marelli, A., Lanzi, P.L.: Support vector regression for classifier prediction. In: GECCO 2007: Proceedings of the 9th annual conference on Genetic and evolutionary computation, pp. 1806–1813. ACM Press, New York (2007)
9. Weisstein, E.W.: Sherman-morrison formula. From MathWorld–A Wolfram Web Resource, `http://mathworld.wolfram.com/Sherman-MorrisonFormula.html`
10. Widrow, B., Hoff, M.E.: Neurocomputing: Foundation of Research. In: Adaptive Switching Circuits, pp. 126–134. MIT Press, Cambridge (1988)

11. Wilson, S.W.: Classifier Fitness Based on Accuracy. Evolutionary Computation 3(2), 149–175 (1995)
12. Wilson, S.W.: Classifiers that approximate functions. Journal of Natural Computing 1(2-3), 211–234 (2002)

A Linear Bayes Analysis

Linear Bayes Analysis defines the *optimal* classifier weight vector $\mathbf{w}$ as the one that minimises the objective function

$$\begin{aligned} J &= E\left[(P - \mathbf{w}^T\boldsymbol{\phi})(P - \mathbf{w}^T\boldsymbol{\phi})^T\right] \\ &= E\left[P^2\right] - E\left[P\boldsymbol{\phi}^T\right]\mathbf{w} - \mathbf{w}^T E\left[P\boldsymbol{\phi}\right] + \mathbf{w}^T E\left[\boldsymbol{\phi}\boldsymbol{\phi}^T\right]\mathbf{w}. \end{aligned} \tag{25}$$

Solving $\partial J/\partial \mathbf{w} = 0$ for $\mathbf{w}$ results in

$$\mathbf{w} = E\left[\boldsymbol{\phi}\boldsymbol{\phi}^T\right]^{-1} E\left[P\boldsymbol{\phi}\right]. \tag{26}$$

The classifier's squared error estimate can be computed by substituting Eq. 26 into Eq. 25, resulting in

$$\begin{aligned} \varepsilon^2 = E\left[P^2\right] - E\left[P\boldsymbol{\phi}^T\right] E\left[\boldsymbol{\phi}\boldsymbol{\phi}^T\right]^{-1} E\left[P\boldsymbol{\phi}\right] - \left(E\left[\boldsymbol{\phi}\boldsymbol{\phi}^T\right]^{-1} E\left[P\boldsymbol{\phi}\right]\right)^T E\left[P\boldsymbol{\phi}\right] + \\ + \left(E\left[\boldsymbol{\phi}\boldsymbol{\phi}^T\right]^{-1} E\left[P\boldsymbol{\phi}\right]\right)^T E\left[\boldsymbol{\phi}\boldsymbol{\phi}^T\right] E\left[\boldsymbol{\phi}\boldsymbol{\phi}^T\right]^{-1} E\left[P\boldsymbol{\phi}\right] = \\ = E\left[P^2\right] - E\left[P\boldsymbol{\phi}^T\right] E\left[\boldsymbol{\phi}\boldsymbol{\phi}^T\right]^{-1} E\left[P\boldsymbol{\phi}\right] \end{aligned} \tag{27}$$

Given that in XCSF we have $\boldsymbol{\phi}^T = [x_0 \quad \mathbf{s}^T]$, and decomposing the weight vector into $\mathbf{w}^T = [w_0 \quad \mathbf{w}'^T]$, we can rewrite $\partial J/\partial \mathbf{w} = 0$ as the following coupled equations

$$E\left[w_0 x_0^2 + \mathbf{w}'\mathbf{s}^T\right] = E\left[P\right], \tag{28}$$

$$E\left[w_0 x_0 \mathbf{s} + \mathbf{w}'\mathbf{s}\mathbf{s}^T\right] = E\left[P\mathbf{s}\right], \tag{29}$$

that, when solved for w_0 and $\mathbf{w}'$, result in,

$$\mathbf{w} = \begin{bmatrix} w_0 \\ \mathbf{w}' \end{bmatrix} = \begin{bmatrix} x_0^{-1}(E\left[P\right] - \mathrm{cov}(P,\mathbf{s})\mathrm{cov}(\mathbf{s},\mathbf{s})^{-1} E\left[\mathbf{s}^T\right]) \\ \mathrm{cov}(P,\mathbf{s})\mathrm{cov}(\mathbf{s},\mathbf{s})^{-1} \end{bmatrix}, \tag{30}$$

where we have

$$\begin{aligned} \mathrm{cov}(P,\mathbf{s}) &= E\left[(P - E\left[P\right])(\mathbf{s} - E\left[\mathbf{s}\right])\right], \\ \mathrm{cov}(\mathbf{s},\mathbf{s}) &= E\left[(\mathbf{s} - E\left[\mathbf{s}\right])(\mathbf{s} - E\left[\mathbf{s}\right])^T\right]. \end{aligned}$$

By substituting the above classifier weight vector definition into Eq. 25, we can rewrite the classifier squared error as follows:

$$\varepsilon^2 = \mathrm{cov}(P,P) - \mathrm{cov}(P,\mathbf{s})^T \mathrm{cov}(\mathbf{s},\mathbf{s})^{-1} \mathrm{cov}(P,\mathbf{s}), \tag{31}$$

where $\mathrm{cov}(P,P) = E\left[(P - E\left[P\right])^2\right]$.

B Bayes Linear Analysis and Least Squares

Let $\mathbf{\Phi}_t$ and $\mathbf{\Pi}_t$ denote respectively the feature and payoff matrix, after t updates, and given by

$$\mathbf{\Phi}_t = \begin{bmatrix} \boldsymbol{\phi}_1^T \\ \vdots \\ \boldsymbol{\phi}_t^T \end{bmatrix}, \qquad \mathbf{\Pi}_t = \begin{bmatrix} P_1 \\ \vdots \\ P_t \end{bmatrix}. \tag{32}$$

The Least Squares algorithm find the weight vector $\mathbf{w}_t$ that minimises the square error $\sum_{i=i}^{t}(P_i - \mathbf{w}_t^T \boldsymbol{\phi}_i)^2$, by solving the normal equation

$$\mathbf{\Phi}_t^T \mathbf{\Phi}_t \mathbf{w}_t = \mathbf{\Phi}_t^T \mathbf{\Pi}_t. \tag{33}$$

By definition, we have

$$\mathbf{\Phi}_t^T \mathbf{\Phi}_t = \sum_{i=1}^{t} \boldsymbol{\phi}_i \boldsymbol{\phi}^T, \tag{34}$$

$$\mathbf{\Phi}_t^T \mathbf{\Pi} = \sum_{i=1}^{t} P_i \boldsymbol{\phi}_i. \tag{35}$$

Multiplying the left and the right hand of Eq. 33 by $1/t$ gives together with Eqs. 34 and 34,

$$\frac{1}{t}\sum_{i=1}^{t} \boldsymbol{\phi}_i \boldsymbol{\phi}^T \mathbf{w}_t = \frac{1}{t}\sum_{i=1}^{t} P_i \boldsymbol{\phi}_i, \tag{36}$$

which, according to the definition introduced by Eqs. 16, 17, and 18, can be written as

$$\hat{E}_{\boldsymbol{\phi}\boldsymbol{\phi},t} \mathbf{w}_t = \hat{E}_{P\boldsymbol{\phi},t}, \tag{37}$$

which results in the classifier weight vector update,

$$\mathbf{w}_t = \hat{E}^{-1}_{\boldsymbol{\phi}\boldsymbol{\phi},t} \hat{E}_{P\boldsymbol{\phi},t}. \tag{38}$$

The squared prediction error (minimised by the above equation) is defined by

$$\varepsilon_t^2 = \frac{1}{t}\sum_{i=1}^{t} \left(P_i - \mathbf{w_t}^T \boldsymbol{\phi}_i\right)^2, \tag{39}$$

and can be expanded to

$$\varepsilon_t^2 = \frac{1}{t}\sum_{i=1}^{t} P_i^2 + \mathbf{w_t}^T \left(\frac{1}{t}\sum_{i=1}^{t} \boldsymbol{\phi}_i \boldsymbol{\phi}_i^T\right) \mathbf{w_t} - \mathbf{w_t}^T \left(\frac{2}{t}\sum_{i=1}^{t} \boldsymbol{\phi}_i\right), \tag{40}$$

which, together with Eqs. 16, 17,18, and 38, gives

$$\varepsilon_t^2 = \hat{E}_{PP,t} - \hat{E}^T_{P\boldsymbol{\phi},t} \hat{E}^{-1}_{\boldsymbol{\phi}\boldsymbol{\phi},t} \hat{E}^T_{P\boldsymbol{\phi},t}. \tag{41}$$

C Recursive Least Squares

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

$$\sum_{t=1}^{t} \left(\mathbf{w}_t^T \boldsymbol{\phi}_t - P_t\right)^2 + \frac{1}{\delta^{\mathrm{RLS}}} \|\mathbf{w}_t\|^2, \tag{42}$$

and satisfies the equality

$$\left(\boldsymbol{\Phi}_t^T \boldsymbol{\Phi}_t + \frac{1}{\delta^{\mathrm{RLS}}} \mathbf{I}\right) \mathbf{w}_t = \boldsymbol{\Phi}_t^T \mathbf{P}_t, \tag{43}$$

where $\mathbf{I}$ denotes the identity matrix and δ^{RLS} is a large positive constant. Let $\mathbf{V}_t^{-1} = \boldsymbol{\Phi}^T \boldsymbol{\Phi}_t$ denote the feature autocorrelation matrix estimate, that satisfies the relation

$$\mathbf{V}_t^{-1} = \mathbf{V}_{t-1}^{-1} + \boldsymbol{\phi}_t^T \boldsymbol{\phi}_t, \tag{44}$$

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

$$\boldsymbol{\Phi}_t^T \mathbf{P}_t = \boldsymbol{\Phi}_{t-1}^T \mathbf{P}_{t-1} + \boldsymbol{\phi}_t P_t, \tag{45}$$

which, together with Eqs. 43 and 44 allows us to derive

$$\mathbf{V}_t^{-1} \mathbf{w}_t = \mathbf{V}_t^{-1} \mathbf{w}_{t-1} + \boldsymbol{\phi}_t (P_t - \mathbf{w}_{t-1}^T \boldsymbol{\phi}_t). \tag{46}$$

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

$$\mathbf{w}_t = \mathbf{w}_{t-1} + \mathbf{V}_t \boldsymbol{\phi}_t (P_t - \mathbf{w}_{t-1}^T \boldsymbol{\phi}_t). \tag{47}$$

To get the update for $\mathbf{V}$, we apply the Sherman-Morrison formula [9] to Eq. 44, resulting in

$$\mathbf{V}_t = \mathbf{V}_{t-1} - \frac{\mathbf{V}_{t-1} \boldsymbol{\phi}_t \boldsymbol{\phi}_t^T \mathbf{V}_{t-1}}{1 + \boldsymbol{\phi}_t^T \mathbf{V}_{t-1} \boldsymbol{\phi}}, \tag{48}$$

which can be written as

$$\beta^{\mathrm{RLS}} = 1 + \boldsymbol{\phi}_t^T \mathbf{V}_{t-1} \boldsymbol{\phi}_t, \tag{49}$$

$$\mathbf{V}_t = \mathbf{V}_{t-1} - \frac{1}{\beta^{\mathrm{RLS}}} \mathbf{V}_{t-1} \boldsymbol{\phi}_t \boldsymbol{\phi}_t^T \mathbf{V}_{t-1}, \tag{50}$$

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 $\mathbf{V}_0^{-1} = (1/\delta^{\mathrm{RLS}}) \mathbf{I} > 0\mathbf{I}$. This introduces a bias that is kept small by setting δ^{RLS} to a large value.

D Tracking Mean Square Error

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. 43 at t satisfies the normal equation

$$\left(\mathbf{\Phi}_t^T \mathbf{\Phi}_t\right) \mathbf{w}_t = \mathbf{\Phi}_t^T \mathbf{P}_t, \tag{51}$$

which can also be written as

$$\mathbf{w}_t^T \mathbf{\Phi}_t^T \left(\mathbf{\Phi}_t \mathbf{w}_t - \mathbf{P}_t\right) = 0. \tag{52}$$

Our aim is to find an incremental update equation for the MSE, $f_t(\mathbf{W}_t)$, that, following Eq. 3, is in matrix notation given by

$$t f_t(\mathbf{w}_t) = \|\mathbf{\Phi}_t \mathbf{w}_t - \mathbf{P}_t\|^2, \tag{53}$$

Using $-\mathbf{P}_t = -\mathbf{\Phi}_t \mathbf{w}_t + (\mathbf{\Phi}_t \mathbf{w}_t - \mathbf{P}_t)$ and Eq. 52, we can derive

$$\begin{aligned} \mathbf{P}_t^T \mathbf{P}_t &= \mathbf{w}_t^T \mathbf{\Phi}_t^T \mathbf{\Phi}_t \mathbf{w}_t - 2\mathbf{w}_t^T \mathbf{\Phi}_t^T (\mathbf{\Phi}_t \mathbf{w}_t - \mathbf{P}_t) + (\mathbf{\Phi}_t \mathbf{w}_t - \mathbf{P}_t)^T (\mathbf{\Phi}_t \mathbf{w}_t - \mathbf{P}_t) \\ &= \mathbf{w}_t^T \mathbf{\Phi}_t^T \mathbf{\Phi}_t \mathbf{w}_t + \|\mathbf{\Phi}_t \mathbf{w}_t - \mathbf{P}_t\|^2, \end{aligned} \tag{54}$$

and thus we can express the sum of squared errors by

$$\|\mathbf{\Phi}_t \mathbf{w}_t - \mathbf{P}_t\|^2 = \mathbf{P}_t^T \mathbf{P}_t - \mathbf{w}_t^T \mathbf{\Phi}_t^T \mathbf{\Phi}_t \mathbf{w}_t. \tag{55}$$

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. 44, 45 and 55, and use $\mathbf{V}_t^{-1} \mathbf{w}_t = \mathbf{\Phi}_t^T \mathbf{P}_t$ after Eq. 51 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

$$t f_t(\mathbf{w}_t) = (t-1) f_{t-1}(\mathbf{w}_{t-1}) + (\mathbf{w}_{t-1}^T \boldsymbol{\phi}_t - P_t)(\mathbf{w}_t^T \boldsymbol{\phi}_t - P_t), \tag{56}$$

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

$$\varepsilon_t^2 = \varepsilon_{t-1}^2 + \frac{1}{t}\left((\mathbf{w}_{t-1}^T \boldsymbol{\phi}_t - P_t)(\mathbf{w}_t^T \boldsymbol{\phi}_t - P_t) - \varepsilon_{t-1}^2\right). \tag{57}$$