Local Ensemble Weighting in the Context of Time Series Forecasting Using XCSF

Matthias Sommer, Anthony Stein, and Jörg Hähner
Organic Computing Group, University of Augsburg, 86159 Augsburg, Germany
[matthias.sommer|anthony.stein]@informatik.uni-augsburg.de

***Abstract*—Time series forecasting constitutes an important aspect of any kind of technical system, since the underlying stochastic processes vary over time. Extensive efforts for designing self-adaptive learning systems have been made, to take system designers out of the loop. One goal of such systems is to transfer design-time decisions, e.g. parametrisation, to the run-time. By means of forecasting the succeeding system state, the system itself is enabled to anticipate, how to reconfigure to handle upcoming conditions. Ensemble forecasting is a specific means of combining and weighting the forecasts of multiple independent forecast methods. This concept has proven successful in various domains today. In this work, we present our self-adaptive forecast module for ensemble forecasting of univariate time series and draw a picture of how the *eXtended Classifier System for Function approximation* (XCSF) can be utilised as a novel weighting approach in this context. We elaborate on the fundamental ideas and evaluate our proposed technique on the basis of several time series with different characteristics.**

I. Introduction

Current research activities focus on turning autonomic computing [1] and organic computing [2] systems from robust into resilient systems. The main purpose of adding adaptation and self-optimisation capabilities to technical systems in general, and organic and autonomic system in particular, is to allow for resilient and flexible solutions. We define the term *resilient* as *pro-active robustness* [3]. The control mechanisms encapsulating the self-adaptive and self-organising capabilities of the system do not only react to detected disturbances and dissatisfying system performance, but are trying to foresee upcoming problems. For the pure control tasks, this means to make forecasts that serve as additional input for the self-adaptation mechanism defined by the architecture. In particular, the system pro-actively changes the control strategies to predicted future states, also trying to become aware of the impact of the decisions taken.

Time series forecasting has been used to support the decision-making process, and to mitigate the uncertainty of future trends in real-world domains [4]–[6]. Linear combination of multiple forecasts from several forecast methods provides more accurate forecasts, than when relying only on one single forecast method [7]. Accordingly, the combination of different learners reduces the model selection risk. We adapt a evolutionary online machine learning technique, the *extended classifier system for function approximation* (XCSF), to multi-model ensemble forecasting [8]. XCSF learns the optimal weights for the combination of individual forecasts from independent forecast techniques at runtime via reinforcement and evolutionary algorithms.

We propose and evaluate a highly self-adaptive technique to relieve an engineer from complex, design-time-situated tasks related to forecast selection, and forecast combination. We demonstrate the capabilities of XCSF for multi-model ensemble forecasting of time series with several univariate time series from different real-world domains exhibiting different characteristics.

The remainder of this paper is structured as follows: First, we provide a brief overview of the related work in this field. We move on, mapping the formal concept of forecast combination to machine learning problems. Based on this theoretical concept, we present how learning classifier systems, in particular the XCSF, can be practically used to tackle this problem. On the basis of differently characterised time series, we compare the XCSF-based forecast combination to three alternative linear combination approaches, and against some individual forecast techniques. We conclude this work with a summary of our findings and an outlook on future work.

II. Related Work

Combining Forecasts: A comprehensive review of forecast combination strategies until 1989 is presented by Clemen [9]. Menezes et al. [10] give an overview over well-established ensemble methods and propose practical guidelines when to use which method. These combination strategies range from simple statistical ensembles, such as the naïve simple average, the trimmed mean, and the median [11], to more complex combination mechanisms. In order to improve the reliability of forecasts and to overcome the limitations and drawbacks of individual techniques, different approaches have been discussed in literature. These include, finding the best individual model from a set of forecast methods [12], [13], combining the forecasts from a given set of methods [9], [14], and finding the optimal set of candidates that improves the forecast precision the most [15]. A recent survey summarises state-of-the-art ensemble methods, such as bagging, boosting, random forest, decomposition methods, and many more [16].

Several researchers investigated the strength and weaknesses of the forecast combination approach. Hibon and Evgeniou [17] showed that the combination of several, individual forecasts on average results in lower forecast errors than when relying only on a single forecast, and that the performance among individual methods is significantly worse than the worst

performance of their possible combinations. A recent study by Adhikari and Agrawal [7] investigates linear combination strategies (e.g. trimmed mean, outperformance, and least square regression) in comparison to five individual forecast techniques. Their results indicate that the forecast accuracies of the individual forecast methods vary notably and that all combination methods significantly reduce the forecast error. Herbst et al. [12] propose a self-adaptive approach that selects suitable forecasting methods for a given context, based on a static decision tree. In contrast to ensemble forecasting, they only select the most promising one out of the set of available forecast methods instead of combining the individual forecasts.

Combining with Machine Learning Techniques: Supervised machine learning techniques learn patterns or functions from a training set of data [18]. They only rely on historical data to learn the stochastic dependency between a set of input and output variables. Prudencio and Ludermir [19] utilise a multi-layer perceptron trained with backpropagation, respectively with the Levenberg-Marquardt algorithm to find the optimal weights for two forecast methods. The input is based on the time series and certain characteristics, such as the length of the time series, and a trend in the data. Their results indicated that the perceptron performs better than the simple average of the two individual forecasts.

In contrast, our approach resembles a self-adaptive technique that learns the optimal combination of several individual forecasts at runtime, and therefore does not rely on training data and offline training.

III. Problem Statement

From the point of view of the machine learning domain, each individual forecast method can be seen as a learner figuring out how to learn the underlying model of a given time series. In general, a time series describes a time-ordered sequence of data points $X_1, ..., X_t$, derived from a system in discrete time intervals of successive measurements. Based on this internal model, each learner gives his independent vote F_i (or forecast in terms of time series forecasting). The outputs of each individual forecast technique $F_1, \ldots, F_n$ are considered with varying impact, weighted according to their expected accuracy. Let W_i $(i = 1 \ldots n)$ be the weight of forecaster i, and F_i its respective forecast. Then, the combined forecast $y(t)$ for time step t can be calculated as

$$y(t) = f(F_1, \ldots, F_n, W_1, \ldots, W_n) = \frac{\sum_{i=1}^{n} F_i * W_i}{\sum_{i=1}^{n} W_i} \quad (1)$$

where F_i is the forecast value and W_i its assigned weight with $W_i \geq 0, \forall i$. Usually, the value range of each weight is restricted to $[0, 1]$, and the sum of all weights equals 1. The problem of finding the optimal weights can be formulated as a minimisation problem:

$$\underset{\vec{W}}{\operatorname{argmin}}(|x(t) - y(t)|) \quad (2)$$

with $x(t)$ being the actual value at time step t, and $y(t)$ being the combination of forecasts at time step t, weighted by $\vec{W} := (W_1, \ldots, W_n)$. The optimisation criterion is the minimisation of the forecast error. This can be achieved by exploring the search space of all possible weight combinations. The key problem is that these weights are not static among a certain set of forecast methods, but are dependent on the current situation within the time series, and the time series itself. In general, the forecast combination problem can be expressed as finding a vector of the optimal weights for an input of forecasts from different forecasting models, to obtain the optimal combination of these forecasts.

In the following, we give a brief introduction to the XCSF [20]. We explain how we utilise an evolutionary, rule-based, online machine learning technique, the XCSF, to evolve rules that encode the optimal weights for the combination of individual forecasts F_i from independent forecast techniques at runtime.

IV. XCSF in a Nutshell

In 1995, Wilson introduced a revealing LCS derivative – the extended classifier system (XCS), an evolutionary rule-based machine learning technique. XCS gained a majority of attention in the broader research field of michigan-style *learning classifier systems* (LCS). XCS was initially utilised for function approximation in 2002 [21]. In his article, Wilson presented XCSF (XCS for function approximation) that aims at approximating an underlying function by means of combining *gradient-based* local learning and a *steady-state niche genetic algorithm* (GA) [20]. More precisely, XCSF attempts to partition the problem space into several subspaces and learns linear approximations of the targeted function within the corresponding niches. Therefore, it generates and maintains a certain set of rules (in this context usually termed classifiers). This set is called classifier *population* $[P]$.

A single classifier cl_j comprises a couple of attributes: 1) The condition C that determines a certain subspace by encoding a geometric structure, 2) an action a that defines a reaction that can be executed on the environment (in the case of XCSF, only a single *dummy action* a_d is used), 3) a weight vector $\vec{w}_j = (w_0, w_1, \ldots, w_n)$ that represents the coefficients for the linear approximation, 4) an error estimate ϵ that reflects the mean absolute prediction error of the linear approximation, 5) a fitness value ϕ that can be roughly interpreted as an inverse of ϵ, which represents the accuracy of the linear approximation.

As illustrated by Figure 1, a single iteration through XCSF's main loop can be described as follows:

At each time step t, XCSF retrieves an n-dimensional situation/feature vector $\sigma(t)$. Next, the population $[P]$ is scanned for any matching classifiers cl_j, which in turn constitute the match set $[M]$. Afterwards, the so-called *system prediction*[1] $P(\sigma(t))$ is calculated by means of a fitness-weighted sum of all linear approximations at the problem space's site $\sigma(t)$,

[1]We note that the term 'system prediction' hails from the XCS terminology and that we clearly differ this from the term 'forecast' that is used throughout the paper to denote a calculated time series value at time $t + 1$.

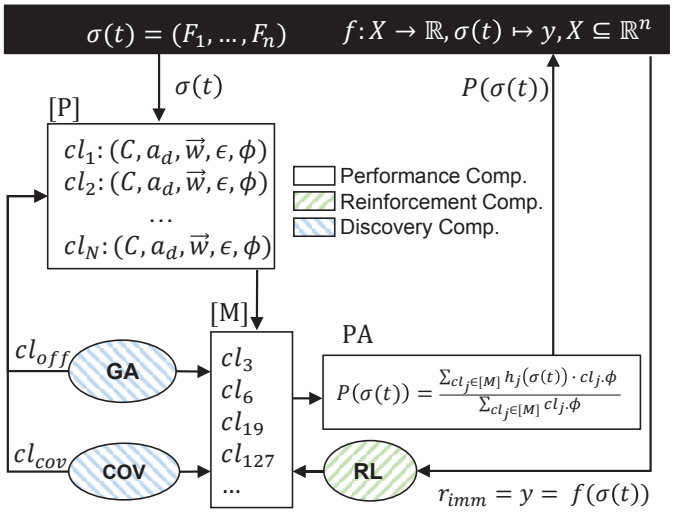


Figure 1: Schematic illustration of conventional XCSF

where the linear approximations of a classifier cl_j is calculated as follows:

$$h_j(\sigma(t)) = w_0 + \sum_{i=1}^{n} w_i \cdot \sigma_i \tag{3}$$

As can be seen, XCSF approximates the underlying problem space in a piece-wise linear fashion and combines overlapping subspaces represented by classifiers cl_j in the same match set $[M]$ (i.e. *environmental niche*) according to their fitness values $cl_j.\phi$. This combination, more precisely the system prediction $P(\sigma(t))$, is then used as output value. All aforementioned steps through the main loop are actualised by XCSF's *performance component*. The actual time series value is set to be the immediate reward r_{imm}. The reward is eventually used to refine the attributes $\vec{w}, \epsilon, \phi$, etc. of each classifier $cl_j \in [M]$ with gradient-based techniques. For the adaptation of the coefficients $\vec{w}$ of the linear approximation, the *recursive least squares* (RLS) algorithm [20] is used. The refinement of the matching classifiers is the task of XCSF's *reinforcement component*.

As Figure 1 depicts, there remains a further part – the *discovery component*, which is responsible for exploring the problem space, and to guarantee immediate response to any input vector $\sigma(t)$ at any time t. Whenever $[P]$ contains no cl_j whose condition C encompasses the current situation $\sigma(t)$, a so-called *covering mechanism* (COV) is activated. This assures that at least one classifier cl_{cov} is generated ad-hoc to cover the current stimuli. The attributes of cl_{cov} are initialised using predefined initial values for $\vec{w}, \epsilon$, and ϕ. The concrete geometric shape of the condition C is generated probabilistically to a certain degree. In our work, we used the general hyper-ellipsoidal condition representation introduced by Butz et al. in [22].

Besides the covering mechanism, a *steady-state niche genetic algorithm* (GA) comes into operation, to refine the geometric shapes during the learning process. Thus, it tries to find the most suitable local structures for any subspace in the function to be learned. It therefore recombines and mutates the conditions C of two selected and copied parental classifiers, resulting in two newly generated offspring classifiers cl_{off} to be inserted in [P]. The remaining classifier attributes such as $\vec{w}, \epsilon, \phi$ are inherited from their parents, and partially adapted according to predefined reduction values. The selection of the parental classifiers can be achieved for instance by *roulette-wheel* (fitness proportionate) or *tournament* selection [23]. This offspring classifier is subsumed by a more general one, if a sufficiently experienced classifier with a prediction error ϵ smaller than the target error ϵ_0 exists in $[M]$ that additionally encompasses the condition of the offspring classifier entirely.

For the sake of brevity, we assume a certain degree of familiarity with the standard XCSF. For a more detailed view on XCS(F) and all relevant algorithmic parts and parameters, we refer the reader to [21], [24], [25].

V. XCSF for Time Series Forecast Combination

In the following paragraphs, we elaborate on the idea of utilizing XCSF as a forecast combination method. Therefore, we first have to introduce some formalisms and interpretations:

As already stated in Section III, a time series can be formalized as follows: $ts = X_1, \ldots, X_t$, where t determines the length of ts. Forecasting strives to look beyond the current value of ts and to predict the value m steps in the future X_{t+m}. We focus on forecasting exactly one step in the future. Thus, we define $\hat{F}_t$ as forecast for the future value X_{t+1}. With XCSF as ensemble time series forecast technique, we further define F_i for $i = 1, \ldots, k$ as the calculated value of the i-th forecast method.

Accordingly, the situation vector $\sigma(t)$ that is retrieved by XCSF is now defined by $\sigma(t) = (F_1, \ldots, F_n)$, i.e. XCSF is presented the individual forecast values of each utilised method in the ensemble. In the case of covering, and to provide XCSF with a sufficient initial prediction, the offset weight w_0 of the initialised weight vector $\vec{w}$ for the newly created classifier cl_{cov} is set to the mean value of the multiple forecasts delivered in the situation vector $\sigma(t)$. For the combination of n forecast methods, $\vec{w} = \{w_0, w_1, ...w_n\}$ resembles the coefficients of the linear prediction using RLS, calculated as:

$$P(\sigma(t)) = w_0 + w_1 * F1 + \cdots + w_n * F_n \tag{4}$$

This leads to an approximated linear function $h(\sigma(t))$ for the region covered by the novel classifier that intersects the ordinate approximately at the actual value's height in the problem function (see Figure 2). Naturally, the proximity to the actual value X_{t+1} depends on the quality of the selected ensemble of forecast methods.

The reward r_{imm} that is retrieved by XCSF after forecasting the value for time step $t+1$ ($\hat{F}_t$) is set to the actual value X_{t+1}. Accordingly, the absolute error can be calculated and further incorporated to update the classifier attributes of all $cl_k \in [M]$.

The time series can be interpreted as a function, where the sampling of the domain is determined by the time a certain value appears. Thus, instead of a uniform sampling, the time series is, conceptually spoken, ordered by time. In each

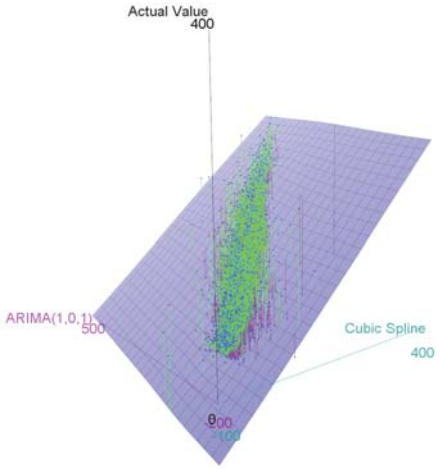


Figure 2: Scatter plot showing the problem space for the combination of two forecast methods (SUNSPOTS data set).

learning step, XCSF is presented the values $\hat{F}_t$ of multiple forecast methods for time step $t+1$.

To provide the reader with an idea of how the problem function that XCSF has to approximate may look like, the plot in Figure 2 depicts a fitted regression surface. The surface is determined by the individual forecast values F_1 and F_2, as well as the time series value X_{t+1}. By approximating such a function, XCSF learns implicitly how much influence each feature in $\sigma(t)$ (i.e. $F_1, \ldots, F_k$) has.

VI. Evaluation

In order to demonstrate the benefit of our approach, we evaluate our combination strategy with several time series with different characteristics, i.e. nonlinearity, trends, seasonality. First, we apply linear scaling, normalising each time series to the value range of $[0; 1]$. Second, we make one-step forecasts for every time step of each time series. Third, based on these forecasts and several forecast accuracy measures, we evaluate the forecast errors, and compare several combination strategies to our XCSF approach.

A. Experimental Setup

a) Time series: To evaluate our combination strategy, we use ten time series from real-world domains. These daily data sets are taken from the Quandl data library[2], and from the FRED economic data library[3]. Their time plots can be seen in Figure 3, and further description is given in Table I. These time series exhibit different characteristics, such as trends, seasonal patterns, or non-stationary behaviour.

b) Parametrisation: Our implementation makes use of the *XCSF-Ellipsoids Java* project [20] that was made available by Patrick Stalph and Martin Butz in 2008. XCSF was mostly parametrised with the default values as suggested by the authors of the project, except of the N parameter determining the maximum number of classifiers in the population [P], and the r_0 parameter defining the initial condition size during the covering process. Therefore, XCSF parameters were set as follows: $N = 200$, $\alpha = 1$, $\beta = 0.1$, $\delta = 0.1$, $r_0 = 0.5$, $\delta_{RLS} = 1000$, $\lambda = 1$, $\nu = 5$, $\epsilon_0 = 0.01$, $\theta_{GA} = 50$, $\theta_{del} = 20$, $\theta_{sub} = 20$, $\mu = 0.05$, $\chi = 1.0$, $\phi_{ini} = 0.1$, $\epsilon_{ini} = 0.0$, $fitnessReduction = 0.1$, $predictionErrorReduction = 1.0$. Only *GA subsumption* was activated. *Compaction* and *condensation* was not used. *Tournament selection* was selected for the choice of parents within the GA.

c) Individual Forecast Methods: We compare our multi-model combination with XCSF against the performance of some individual forecast methods [26], and against three linear combination strategies (Simple average, optimal weights, and outperformance [10]).

Moving average (MA) equals an ARIMA(0,0,1) model. It has the advantage of quick low cost updates and accurate short-term forecasts. However, it does not cope well with trend or seasonality.

Cubic spline's smoothing (CS) model is equivalent to an ARIMA(0,2,2) model, but with a restricted parameter space. Its advantage over the full ARIMA model is that it provides a smooth historical trend, as well as a linear forecast function.

ARIMA(1,0,1) resembles a first-order autoregressive model with one order of a moving average process.

The *exponential smoothing state space* model (ETS) is fully automatic and estimates the model based on the given time series only.

The *random walk with drift* model (RW) is calculated as

$$Y_t = c + Y_{t-1} + Z_t \tag{5}$$

where Z_t is a normal error.

Within this evaluation, we define the input of CS as the last ten time series values. ETS and ARIMA make their forecasts based on the last 25 observations. The MA method considers the last three values. The random walk method estimates the drift based on the last ten observations.

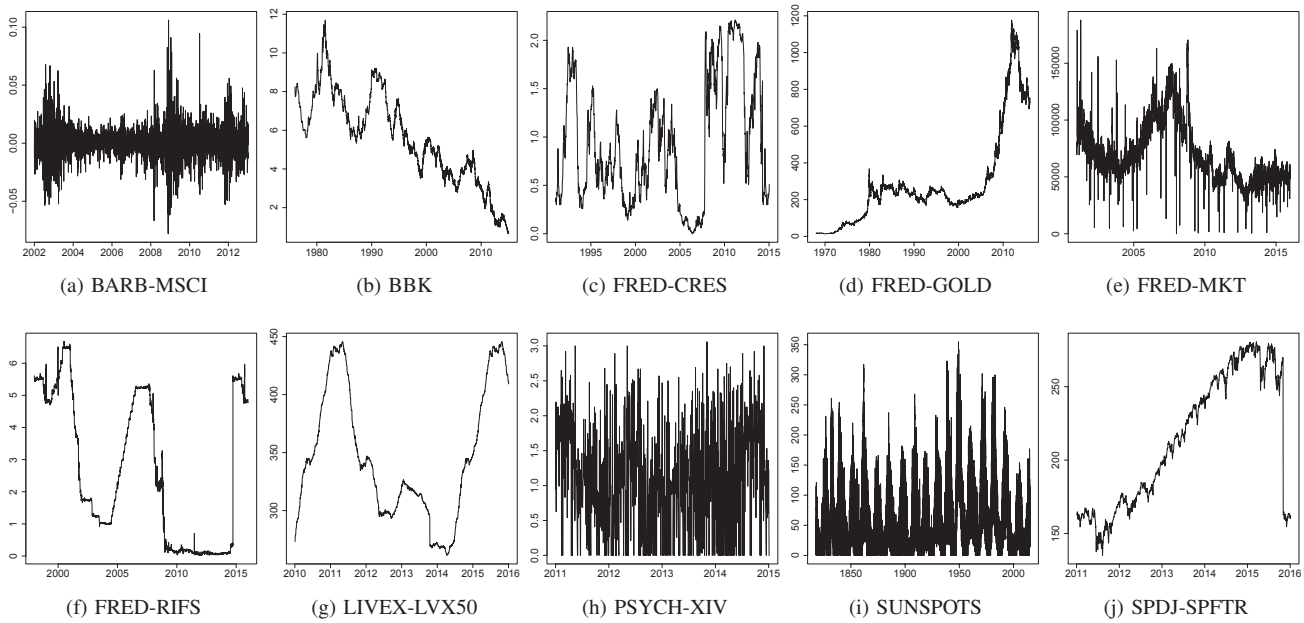
d) Combination Strategies: The *simple average* (SA) simply combines the individual forecasts by calculating their average.

Optimal weights (OW) estimates the linear weights to minimise the error variance of the combination (assuming unbiasedness for each individual forecast). The sum of the resulting weights equals one and no individual weight can be outside the interval $[0, 1]$. The weights are estimated bases on a short history of n forecast errors for each forecast model. A parameter study with $n = \{5, 10, 15, 20, 25\}$ indicated that $n = 5$ offers the best results.

Outperformance (OP) calculates each individual weight as the probability that its respective forecast will perform the best (in the smallest absolute error sense) on the next iteration. Each weight is estimated as the fraction of occurrences in which its respective forecasting model has performed the best in the latest k executions. We conducted a brief parameter study for $k = \{5, 10, 15, 20, 25\}$. The best performance was achieved with $k = 20$. We therefore use this parameter setting in the following evaluation.

[2]http://www.quandl.com, last access: 2016/03/14

[3]https://research.stlouisfed.org/fred2/, last access: 2016/03/15



(a) BARB-MSCI (b) BBK (c) FRED-CRES (d) FRED-GOLD (e) FRED-MKT

(f) FRED-RIFS (g) LIVEX-LVX50 (h) PSYCH-XIV (i) SUNSPOTS (j) SPDJ-SPFTR

Figure 3: Plots of the time series used for the evaluation.

Table I: Description of the time series data sets.

Time series	Description	Type	Length
BARB-MSCI	MSCI Equity Index	stationary, non-seasonal	2,918
BBK	Bundesbank: Yields on debt securities outstanding issued by residents	non-stationary, trend, seasonal	9,925
FRED-CRES	Cleveland Financial Stress Index	non-stationary, seasonal	5,810
FRED-GOLD	LBMA Gold Price: Daily Prices	non-stationary, trend, non-seasonal	12,118
FRED-MKT	Total Commercial Paper Issues with a Maturity Between 1&4 Days	stationary, non-seasonal	3,813
FRED-RIFS	15-Day AA Financial Commercial Paper Interest Rate	non-stationary, seasonal	4,183
LIVEX-LVX50	Daily prices of 50 Fine Wine Index	non-stationary, seasonal	1,633
PSYCH-XIV	Sentiment volume ratios for stocks	stationary, non-seasonal	921
SUNSPOTS	Daily total number of sunspots	stationary, seasonal	68,873
SPDJ-SPFTR	S&P Dow Jones 500 Futures Index	non-stationary, trend, seasonal	1281

e) Evaluation Measures: Theil's U-statistic is a relative accuracy measure, comparing the RMSE of the proposed method against the RMSE of a naive one-step ahead forecast:

$$U = \sqrt{\frac{\sum_{t=1}^{n-1} (\frac{F_{t+1} - Y_{t+1}}{Y_t})^2}{\sum_{t=1}^{n-1} (\frac{Y_{t+1} - Y_t}{Y_t})^2}} \tag{6}$$

where Y_t is the actual value of a point for a given time period t, F_t its respective forecast, and n is the number of data points.

The symmetric mean absolute percentage error (SMAPE) is an accuracy measure based on percentage (or relative) errors. It is calculated as:

$$SMAPE = \frac{1}{n} \sum_{t=1}^{n} \frac{|F_t - Y_t|}{|F_t| + |Y_t|}. \tag{7}$$

The mean absolute scaled error (MASE) [27] compares the forecast accuracy with the average forecast error of the one-step naive forecast method:

$$MASE = \frac{\sum_{t=1}^{n} |Y_t - F_t|}{\frac{n}{n-1} \sum_{i=2}^{n} |Y_i - Y_{i-1}|}. \tag{8}$$

Its lower bound is 0, and the method with the lowest MASE offers the best accuracy. MASE allows to compare forecast accuracy between time series.

B. Experimental Results

The following section presents the results of the evaluation for an ensemble of two forecast methods, ARIMA(1,0,1) and cubic spline. Table III shows the SMAPE, MASE, and U-statistic values for each time series, forecast method, and combination strategy. The average ranking for each strategy is also given in the last column, labelled "Avg. rank". For each forecast accuracy measure, the strategies are ranked according to the "Avg. rank", i.e. the last column. As we can see, ARIMA(1,0,1) provided rather accurate forecasts, whereas the forecasts of cubic spline resulted in rather high forecast errors. A good combination strategy should therefore tend to weight ARIMA(1,0,1) higher than cubic spline.

Table II lists the most influential classifiers out of a population of 83 classifiers, evolved after 9000 time steps for the BBK time series. For each classifier, its condition, its coefficient vector $\vec{w}$, its fitness ϕ, its experience, and

Table II: XCSF population showing the most important classifiers with the highest fitness for the BBK time series (sorted by fitness ϕ desc.).

Cond.: Center / Stretch	**Coefficients** $\vec{w}$	ϕ	**Exp.**	ϵ^4
0.516, 0.516 / 0.39, 0.79	0.516, 0.760, 0.240	0.272	4530	4.2
0.660, 0.516 / 0.20, 0.79	0.624, 0.745, 0.255	0.216	5327	1.6
0.517, 0.520 / 0.39, 0.79	0.517, 0.740, 0.259	0.154	3526	4.2
0.661, 0.516 / 0.20, 0.79	0.621, 0.724, 0.275	0.151	2935	2.4
0.661, 0.520 / 0.23, 0.79	0.628, 0.775, 0.221	0.151	922	2.4

its prediction error ϵ^4 are presented. As we can see, these classifiers give ARIMA(1,0,1) an average weight $w_1 = 0.75$, and CS an average weight $w_2 = 0.25$. This result confirms that XCSF is able to learn to assign higher weights to forecast methods with higher accuracy.

Looking at the individual conditions, it can be clearly seen that each of the five listed classifiers cover approximately the same region of the problem space. In other words, the center and stretch values of the hyper-ellipsoids are rather similar. We attribute this effect to the smaller amount of opportunities of the GA to refine the condition structures, since the learning task only comprised ≈ 9000 steps. Another reason hails from the shape of the underlying problem function as depicted in Figure 2. Indeed, it seems to be possible to only create one single hyper-ellipsoid with a linear approximation to resemble the regression surface. However, XCSF needs more than one classifier to allow for exploratory behavior exerted by the GA, to find a nearly optimal condition structure. The classifier that fits this particular problem function best is expected to outperform any competing classifiers in terms of its fitness, its numerosity, and its experience. Nonetheless, such in-depth investigations regarding the classifier evolution have to handled in future works.

Concerning the U-statistic and the MASE measure, a forecast should only be considered in case its value is lower than 1. Table III shows that, in most cases, XCSF is better or similar to this reference value. All other combination strategies are mostly above this boundary. The results indicate that in the case of stationary time series, XCSF performs similar or slightly superior to the best reference technique. XCSF outperforms the two individual forecast techniques for every time series. Time series that are characterised by trends seem to be challenging for XCSF. For now, we attribute this observation to the fact that XCSF gets no chance to reinforce already evolved classifiers, since they are not, or seldom, triggered again, due to the trend. Put differently, when a time series follows a trend, the already covered niches within the approximated problem function will not, or rarely, be sampled again. Thus, XCSF is not able to learn an adequate weighting for these situations. Nonetheless, we deem XCSF as promising candidate for ensemble time series forecasting, but more efforts regarding the parametrisation have to be done in future work.

[4] ϵ values are to be multiplied by 10^{-3}

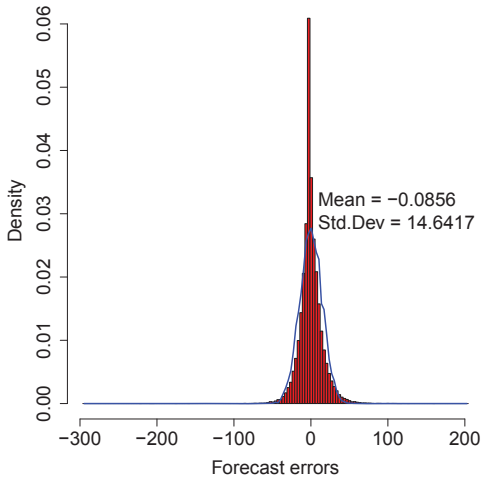


Figure 4: Forecast error histogram for XCSF (SUNSPOTS data set).

a) Different Number of Forecast Methods: : Table IV evaluates the combination strategies for ensembles of size 2, 3, 5 and 7. The ensembles comprise the following forecast methods: set size 2 = ARIMA + CS; 3 = ARIMA + CS + MA; 5 = ARIMA + CS + MA + RW + ETS. The results are averaged over all ten time series. Consequently, set size 2 resembles the average results of Table III. As stated before, a combination of more forecast methods increases the forecast accuracy. The lowest MASE, U-measure, and SMAPE values were achieved using five forecast methods. Considering MASE and U-measure, XCSF has the lowest values. In contrast, for the SMAPE measure, it ranks last. However, the difference to the other strategies is rather small.

b) Accuracy of the Predictive Model: To confirm the reliability of our model, we check both the correlations between errors for successive forecasts, and that the errors are normally distributed with a mean of zero. The representative histogram of the forecast errors in Figure 4 supports the observation that the distribution of forecast errors is centred around zero, and approximately normally distributed. Additionally, we carried out a Ljung-Box test to determine whether there is significant evidence for non-zero correlations at lags 1 to 20. The Ljung-Box test returns a p-value of ≈ 0.13, indicating that there is little evidence for non-zero autocorrelations in the forecast errors for lags 1 to 20. These observations hold for the other time series as well. Therefore, we can assume that XCSF provides an adequate predictive model, which probably cannot be substantially improved upon.

Figure 5 presents the results for the MASE forecast accuracy measure for several set sizes, averaged over all ten time series. The box plots show the statistical distribution of the MASE errors for all time series. The bottom and top of the box represent the first and third quartiles, and the band inside the box represents the median. Outliers are indicated by separate points. The plot shows that XCSF has the lowest average MASE, while also having the lowest lower quartile.

Table III: Results per time series for an ensemble with two forecast methods (ARIMA(1,0,1) and Cubic spline) (bold values highlight the best performances).

	MSCI	BBK01	CRES	GOLD	MKT14	RIFS	LVX50	PSYCH	SUNSPOTS	SPFTR	Avg. rank
U-measure											
XCSF	**1.00**	**0.92**	**0.98**	1.31	**0.52**	**1.02**	**0.19**	**0.98**	**1.07**	**0.38**	1.3
MA	1.01	1.13	1.05	1.23	1.52	1.14	1.45	1.20	1.14	1.14	2.5
ARIMA(1,0,1)	1.02	1.15	1.08	1.23	0.99	1.03	1.93	1.50	**1.07**	1.13	2.8
OW	1.10	1.18	1.09	1.28	1.25	1.30	1.84	1.46	1.22	1.17	4.0
OP	1.22	1.15	1.10	**1.21**	2.02	1.56	1.92	1.50	1.38	1.15	4.2
CS	1.22	1.34	1.32	1.48	2.07	1.64	1.54	1.97	1.53	1.38	5.6
SMAPE											
ARIMA(1,0,1)	76.34	**0.38**	1.53	**0.47**	4.94	**3.35**	**0.11**	35.00	26.72	**0.40**	2.2
SA	74.93	**0.38**	**1.49**	0.49	5.80	3.81	**0.11**	41.34	27.86	**0.40**	2.3
XCSF	88.57	**0.38**	1.51	0.62	**4.75**	3.50	0.13	**34.60**	**25.93**	**0.40**	2.7
OW	**74.90**	0.39	**1.49**	0.51	5.38	4.19	0.12	45.18	28.70	0.42	3.4
OP	75.11	0.39	1.51	0.48	7.45	4.75	0.13	36.20	30.09	**0.40**	3.7
CS	75.19	0.46	**1.49**	0.60	7.80	5.11	0.13	48.68	29.49	0.50	4.9
MASE											
XCSF	**0.69**	**1.05**	**1.02**	**1.13**	**0.99**	1.16	1.15	**0.78**	**1.07**	**0.98**	1.2
ARIMA(1,0,1)	0.73	1.07	1.08	1.15	1.01	**1.12**	1.39	0.79	1.09	1.11	2.5
SA	0.90	1.07	1.03	1.19	1.18	1.18	1.38	0.97	1.14	1.14	3.2
OW	1.06	1.10	1.04	1.25	1.09	1.26	**0.84**	1.12	1.20	1.17	3.8
OP	1.24	1.08	1.07	1.16	1.55	1.41	1.59	0.81	1.34	1.13	4.2
CS	1.28	1.28	1.14	1.47	1.64	1.52	1.59	1.35	1.44	1.41	5.8

Table IV: Average SMAPE, U-measure, and MASE for forecast method ensembles with different size (bold values highlight the best performances).

Measure	**Strategy**	**Set size**			**Avg. rank**
		2	3	5	
U-measure	XCSF	**0.84**	**0.77**	**0.75**	1.00
	SA	1.20	1.04	1.00	2.00
	OW	1.29	1.08	1.01	3.00
	OP	1.42	1.23	1.20	4.00
SMAPE	SA	**15.57**	**15.05**	14.97	1.33
	OW	16.01	15.16	**14.90**	2.00
	OP	**15.57**	17.43	15.59	2.67
	XCSF	16.04	15.68	15.71	3.67
MASE	XCSF	**1.00**	**0.95**	**0.93**	1.00
	SA	1.12	1.02	0.98	2.00
	OW	1.18	1.05	0.98	2.67
	OP	1.24	1.16	1.10	4.00

Independent of the number of forecast methods combined, XCSF offers the lowest MASE. For larger set sizes, MASE is reduced for all combination strategies. Based on our findings, we assume that additional forecast methods do not increase the forecast accuracy.

c) XCSF's Learning Behaviour: Figure 6 depicts the development of two XCSF-specific metrics, i.e. the averaged system error that corresponds to the mean absolute (forecast) error as well as the averaged size of the population [P]. Each data point is the average of 700 XCSF-combined forecasts. The curves show the development of the corresponding means over 30 i.i.d. experimental runs with different random seeds to face the bias that occurs due to the randomized operators (e.g. crossover, mutation, covering) XCSF relies on.

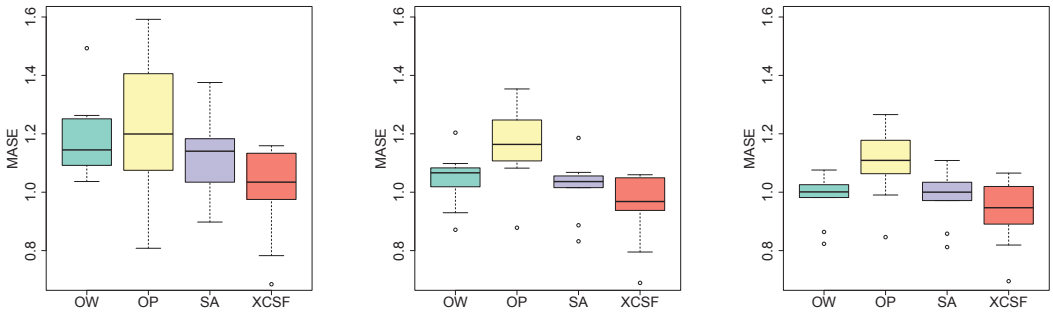
As you can see, the chosen population size restriction of $N = 200$ micro-classifiers is sufficient to approximate the problem function illustrated in Figure 2. Higher values (up to 6400) did not lead to smaller forecast errors. The number of physically stored classifiers (macro-classifiers with numerosity $cl.num > 1$) converges after approximately 10.000 steps to an amount of about 140.

Obviously, the forecast error continuously fluctuates between approximately 0.04 and 0.005. The number of peaks and valleys correlates with the seasons in the SUNSPOTS time series (cf. Figure 3). We attribute the fluctuation effect to the particular learning task XCSF is confronted with. The goal XCSF is striving for is not to forecast the next time series value directly, but to learn which of the individual forecast methods shall get a higher weight in which situation. Thus, when both single methods are forecasting values with a high error, XCSF also outputs a combined value that leads to an error with a similar (nonetheless mostly smaller) amount.

VII. Conclusion & Further Research

We applied the extended classifier system for function approximation XCSF to the challenging task of multi-model ensemble time series forecasting. Our presented approach was compared against two individual forecast methods, and three established combination strategies. The evaluation was done with real-world time series exhibiting different characteristics. Using this setup, we demonstrated that the utilisation of XCSF as forecast combination approach is competitive in comparison to the reference methods whenever the time series is stationary.

An in-depth investigation of the proposed concept with regard to XCSF-specific metrics and issues, such as the classifier evolution, and the formalisation of parametrisation guidelines constitute a top priority on our research agenda. We further plan to evaluate our combination with XCSF for longer forecast horizons. Additionally, standard XCSF as utilised in the present work will be compared to an interpolation-assisted variant as proposed in [25].



(a) Set size 2 (b) Set size 3 (c) Set size 5

Figure 5: Boxplots showing the MASE error averaged over all ten time series (lower values are better).

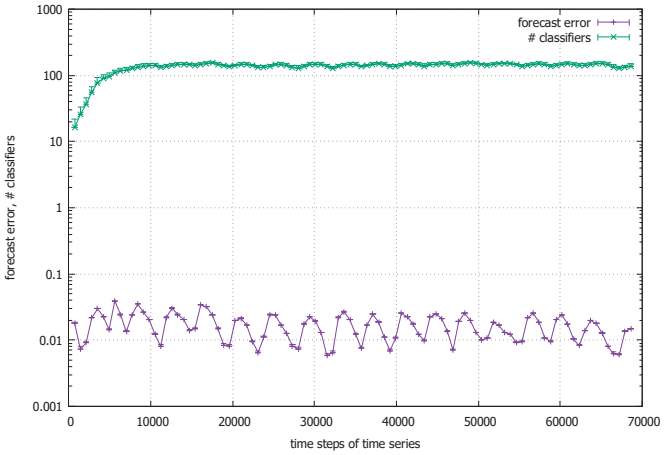


Figure 6: Illustration of XCSF's learning process in terms of forecast error and population size for the ensemble with ARIMA(1,0,1) and cubic spline. The SUNSPOTS data set was chosen since it contains the longest series (68873 time steps).

ACKNOWLEDGEMENT

The authors would like to thank Michael Vogt for his contribution in the scope of his master's thesis. Further on, we greatly appreciate the provision of the XCSF-Ellipsoids Java project that was developed by Patrick O. Stalph and Martin V. Butz.

REFERENCES

[1] J. O. Kephart and D. M. Chess, "The Vision of Autonomic Computing," *IEEE Computer*, vol. 36, no. 1, pp. 41–50, 2003.

[2] C. Müller-Schloer, H. Schmeck, and T. Ungerer, *Organic Computing - A Paradigm Shift for Complex Systems*, 1st ed. Springer, 2011.

[3] E. Hollnagel, D. D. Woods, and N. Leveson, *Resilience Engineering: Concepts And Precepts*. Ashgate Pub Co, 2006.

[4] D. Kramer and W. Karl, "Realizing a Proactive, Self-Optimizing System Behavior within Adaptive, Heterogeneous Many-Core Architectures," in *Self-Adaptive and Self-Organizing Systems (SASO), 2012 IEEE Sixth Intern. Conference on*, 2012, pp. 39–48.

[5] S. VanSyckel, D. Schafer, G. Schiele, and C. Becker, "Configuration Management for Proactive Adaptation in Pervasive Environments," in *Self-Adaptive and Self-Organizing Systems (SASO), 2013 IEEE 7th Intern. Conference on*, 2013, pp. 131–140.

[6] M. Sommer, S. Tomforde, and J. Hähner, *Autonomic Road Transport Support Systems*. Springer International, 2016, ch. An Organic Computing Approach to Resilient Traffic Management, pp. 113–130.

[7] R. Adhikari and R. K. Agrawal, "Performance evaluation of weights selection schemes for linear combination of multiple forecasts," *Artif. Intell. Rev.*, pp. 529–548, 2014.

[8] J. Armstrong, *Principles of Forecasting: A Handbook for Researchers and Practitioners*, ser. Operat. Res. & Manag. Science. Springer, 2001.

[9] R. T. Clemen, "Combining forecasts: A review and annotated bibliography," *Int. Journal of Forecasting*, vol. 5, no. 4, pp. 559 – 583, 1989.

[10] L. M. de Menezes, D. W. Bunn, and J. W. Taylor, "Review of guidelines for the use of combined forecasts." *European Journal of Operational Research*, vol. 120, no. 1, pp. 190–204, 2000.

[11] V. R. R. Jose and R. L. Winkler, "Simple robust averages of forecasts: Some empirical results," *Intern. Journal of Forecasting*, vol. 24, no. 1, pp. 163–169, 2008.

[12] N. R. Herbst, N. Huber, S. Kounev, and E. Amrehn, "Self-adaptive Workload Classification and Forecasting for Proactive Resource Provisioning," in *Proc. of the 4th ACM/SPEC International Conference on Performance Engineering*, ser. ICPE '13. ACM, 2013, pp. 187–198.

[13] R. Prudêncio and T. Ludermir, "Meta-learning approaches to selecting time series models." *Neurocomputing*, vol. 61, pp. 121–137, 2004.

[14] R. L. Winkler and S. Makridakis, "The combination of forecasts," *Journal of the Royal Statistical Society. Series A*, pp. 150–157, 1983.

[15] E. Alpaydın, *Maschinelles Lernen*. Oldenbourg, 2008, ISBN: 9783486581140.

[16] Y. Ren, L. Zhang, and P. N. Suganthan, "Ensemble classification and regression-recent developments, applications and future directions," vol. 11, no. 1, pp. 41–53, 2016.

[17] M. Hibon and T. Evgeniou, "To combine or not to combine: selecting among forecasts and their combinations," *Intern. Journal of Forecasting*, vol. 21, no. 1, pp. 15–24, 2005.

[18] T. Hastie, R. Tibshirani, and J. Friedman, *The Elements of Statistical Learning*. Springer, 2001.

[19] R. Prudencio and T. Ludermir, "Learning Weights for Linear Combination of Forecasting Methods," in *Proc. of Ninth Brazilian Symposium on Neural Networks*, 2006, pp. 113–118.

[20] M. Butz, P. Lanzi, and S. Wilson, "Function Approximation With XCS: Hyperellipsoidal Conditions, Recursive Least Squares, and Compaction," *Evolut. Comput., IEEE Trans. on*, vol. 12, no. 3, pp. 355–376, 2008.

[21] S. W. Wilson, "Classifiers that approximate functions," *Natural Computing*, vol. 1, no. 2, pp. 211–234, 2002.

[22] M. V. Butz, P. L. Lanzi, and S. W. Wilson, "Hyper-ellipsoidal Conditions in XCS: Rotation, Linear Approximation, and Solution Structure," in *GECCO*, 2006, pp. 1457–1464.

[23] M. V. Butz, K. Sastry, and D. E. Goldberg, *Genetic and Evolutionary Computation Conference Chicago*. Springer, 2003, ch. Tournament Selection: Stable Fitness Pressure in XCS, pp. 1857–1869.

[24] M. Butz and S. Wilson, "An algorithmic description of XCS," *Soft Computing - A Fusion of Foundations, Methodologies and Applications*, vol. 6, pp. 144 – 153, 2002.

[25] A. Stein, C. Eymüller, D. Rauh, S. Tomforde, and J. Hähner, "Interpolation-based Classifier Generation in XCSF," in *Proc. of Congress on Evolutionary Computation (CEC 2016)*, 2016.

[26] R. Hyndman and Y. Khandakar, "Automatic time series forecasting: the forecast package for R," *J. of Stat. Softw.*, vol. 26, no. 3, pp. 1–22, 2008.

[27] R. J. Hyndman and A. B. Koehler, "Another look at measures of forecast accuracy," *International Journal of Forecasting*, pp. 679–688, 2006.