Filtering Sensory Information with XCSF: Improving Learning Robustness and Control Performance

Jan Kneissler
Institute of Cognitive Modeling
University of Tübingen
Sand 14, 72076 Tübingen, Germany
jan.kneissler@uni-tuebingen.de

Patrick O. Stalph
Institute of Cognitive Modeling
University of Tübingen
Sand 14, 72076 Tübingen, Germany
patrick.stalph@uni-tuebingen.de

Jan Drugowitsch
Institut National de la Santé et de la Recherche Médicale
Ecole Normale Supèrieure
29 rue d'Ulm, 75005 Paris, France
jan.drugowitsch@ens.fr

Martin V. Butz
Institute of Cognitive Modeling
University of Tübingen
Sand 14, 72076 Tübingen, Germany
butz@uni-tuebingen.de

ABSTRACT

It was previously shown that the control of a robot arm can be efficiently learned using the XCSF classifier system. So far, however, the predictive knowledge about how actual motor activity changes the state of the arm system has not been exploited. In this paper, we exploit the forward velocity kinematics knowledge of XCSF to alleviate the negative effect of noisy sensors for successful learning and control. We incorporate Kalman filtering for estimating successive arm positions iteratively combining sensory readings with XCSF-based predictions of hand position changes over time. The filtered arm position is used to improve both trajectory planning and further learning of the forward velocity kinematics. We test the approach on a simulated, kinematic robot arm model. The results show that the combination can improve learning and control performance significantly. However, it also shows that variance estimates of XCSF predictions may be underestimated, in which case self-delusional spiraling effects hinder effective learning. Thus, we introduce a heuristic parameter, which limits the influence of XCSF's predictions on its own further learning input. As a result, we obtain drastic improvements in noise tolerance coping with more than ten times higher noise levels.

Categories and Subject Descriptors

F.2.1 [**Analysis of Algorithms and Problem Complexity**]: Numerical Algorithms and Problems; G.1.2 [**Numerical Analysis**]: Approximation—*approximation of surfaces and contours, least squares approximation, nonlinear approximation*; I.2.6 [**Artificial Intelligence**]: Learning

General Terms

Algorithms, Theory, Performance

Keywords

Noise, Robotics, Function Approximation, Learning Classifier Systems, XCSF

1. INTRODUCTION

The learning classifier system for function approximation XCSF [27] – a derivative of the original XCS learning classifier system [23] – approximates functions by a population of locally-weighted piece-wise linear approximators. Each approximator is usually referred to as a *classifier*, whose condition determines its activity and thus its local influence given a current input. The classifier prediction is a linear approximation of the encountered input-output value combinations. Recently, it was shown [9, 21] that XCSF shares many similarities with the locally-weighted projection regression (LWPR) algorithm [22], which is well-known in the neuro-robotics literature [17, 18]. In this perspective, XCSF classifiers may be considered as neural structures that specify local receptive fields. The linear predictions of the currently active receptive fields of an XCSF population are combined and weighted dependent on their relative strength.

Facing robotic control problems, XCSF was successfully applied to arm control problems up to seven degrees of freedom [19]. XCSF's capabilities were evaluated based on the control performance of the system. In particular, XCSF was trained to learn the forward velocity kinematics of the controlled arm, mapping changes in joint angles to changes in hand locations given the current arm configuration. Its knowledge was only used to generate inverse control commands, inverting the locally linear mappings for generating directional hand movements. The forward knowledge, on the other hand, has not been further exploited in any way so far.

In this paper, we exploit the forward knowledge of the XCSF system. While learning with XCSF, we filter the successive perceptions of the simulated robot arm by means of Kalman filtering techniques. To do so, XCSF's knowledge has to be exploited for deriving its current predictions about action-effects, but also its current estimate of its certainty about its prediction. Thus, XCSF's error estimate needs to reflect the variance in its current prediction, as investigated elsewhere [12, 11, 16]. When implementing the filtering technique without any precautions, we show that the system may learn from over-filtered sensory information and thus may get stuck in a self-delusional loop, where XCSF over-believes its own predictions. When effectively limiting the estimated prediction confidence, however, we can show that the resulting XCSF system can deal with much larger measurement noise, still learning its bidirectional forward prediction and inverse control structures effectively.

We now first give an overview of XCSF for arm control. Next, we specify the necessary modifications and the new interaction cycle when XCSF applies Kalman filtering in the arm control scenario. We scrutinize the performance of XCSF on two simulated arm models with two and seven degrees of freedom, respectively. As the main result, we show that with proper settings XCSF can solve problems with a higher magnitude of sensory noise. Summary and future work considerations conclude the paper.

2. XCSF FOR ARM CONTROL

Learning Classifier Systems (LCSs) were originally introduced by John H. Holland [13]. The accuracy-based XCS learning classifier system was introduced by Stewart W. Wilson [23]. XCS was successfully applied in binary classification tasks [23, 4], data mining problems [2, 25], and function approximation problems [26, 27, 15, 7], among others [3]. In the real-valued function approximation case, XCS is termed XCSF, essentially constituting an iteratively learning regression system.

We focus on the XCSF system [27, 7] and its potential for learning the forward velocity kinematics of a robot arm for inverse control [5, 10, 19]. The considered XCSF setup is based on the available implementation [20] and the detailed descriptions available in the literature [5, 7, 19]. For introducing the novel aspects into the system, several modifications were necessary. We now first specify the relevant aspects and then detail the applied modifications.

2.1 Learning Forward Velocity Kinematics

In general, XCSF learns to approximate multi-dimensional functions using piece-wise, linear models. It evolves a population of classifiers, where each classifier covers a particular subspace of the input space, which may be termed the receptive field of a classifier. Moreover, each classifier learns a linear model for approximating the function surface in its subspace. The linear model of a classifier is typically adapted by means of recursive least squares [7, 14]. The receptive field, that is, the condition part of a classifier, is evolved over time by a steady-state genetic algorithm (GA). We use general Gaussian kernel-based receptive fields, yielding ellipsoidal regions of influence for a classifier. The population of classifiers overall yields a function approximation surface. The goal of XCSF is to minimize the absolute or squared difference between this surface and the encountered function.

XCSF is easily modifiable to learn the velocity kinematics of a robot arm. A robot arm may be characterized by its *configuration space* $\mathcal{C} \subset \mathbb{R}^n$ and its *task space* $\mathcal{T} \subset \mathbb{R}^m$ of the end-effector. While the task space is usually encoded in a Cartesian coordinate system, the configuration space may be encoded by joint angles. Due to the arm kinematics, a configuration $\mathbf{q} \in \mathcal{C}$ uniquely determines the corresponding end-effector location in task space $\boldsymbol{x} \in \mathcal{T}$. This *forward kinematics* mapping can be expressed as a typically non-linear function

$$\boldsymbol{x} = f(\mathbf{q}). \tag{1}$$

In the literature, XCSF has been mainly applied for learning the *forward velocity kinematics* of a robot arm. Given the current joint angle configuration $\mathbf{q}$, the velocity kinematics can be written as

$$\dot{\boldsymbol{x}} = \mathbf{J}(\mathbf{q})\dot{\mathbf{q}} \tag{2}$$

where $\mathbf{J}(\mathbf{q}) = \partial f / \partial \mathbf{q}$ is the $m \times n$ Jacobian matrix. When XCSF learns the velocity kinematics, it thus partitions the configuration space $\mathcal{C}$ with its receptive fields to approximate the locally linear Jacobians $\mathbf{J}(\mathbf{q})$ given the specified configuration $\mathbf{q}$.

To control the robot arm, a goal direction needs to be translated into necessary control commands. Given an arm with redundant degrees of freedom, that is, $n > m$, the inverse is not uniquely defined. That is, the inverse of $\mathbf{J}$ is under-determined. Here, we pick the *pseudo-inverse* or *Moore-Penrose* matrix [1], which represents the solution with minimum norm of this inverse kinematics problem. However, it was shown elsewhere [8, 19] that other redundancy resolution techniques can be applied, such as applying additional movement constraints.

To sum up, XCSF learns the mapping from configuration space velocities $\dot{\mathbf{q}}$ to task space velocities $\dot{\boldsymbol{x}}$, depending on the current configuration $\mathbf{q}$. XCSF is well-suited for this task, since the algorithm is able to cluster a context space while learning a function that operates on a different space, but depends on the context. In our task the current configuration is the context and XCSF clusters the configuration space with rotating hyper-ellipsoidal RFs [7]. In turn, each classifier approximates the Jacobian matrix using linear recursive least squares approximations. Given a particular configuration, the classifiers whose conditions overlap with the current configuration are considered; their linear Jacobians are combined in a weighted manner; finally, the resulting Jacobian matrix is inverted for generating a control signal.

2.2 Error and Confidence Estimation

While XCSF traditionally estimates the mean absolute deviation of its prediction, a variance estimate is necessary to combine the prediction with other, e.g., sensory information. Thus, XCSF is modified by estimating the standard deviation of a classifier prediction, rather than the mean absolute deviation. This was proposed and investigated elsewhere for the XCSF system in function approximation problems [16], where it was shown that the performance of XCSF does generally not suffer from this system change.

In general, given k samples (q, x) a classifier has learned from, the variance of the error of a single classifier in XCSF may be estimated by using the sequence of values x_i and respectively generated predictions x_i^{p}:

$$\hat{\sigma}_k^2 = \frac{1}{k} \sum_{i=1}^{k} (x_i - x_i^{\mathrm{p}})^2. \tag{3}$$

Note that x_i^{p} specifies the prediction that was generated when the sample was encountered. It is thus based on the knowledge of the classifier available until that point in time. A disadvantage of this approach is that the estimate tends to be pessimistic, in that the true σ_k^2 is likely to be smaller because the improvement of the linear model due to later samples is not taken into account. The advantage, however, is that it is not necessary to calculate the prediction using all previous samples in every time step. In consequence, it is not required to keep a history of the previous samples. Rather, the variance estimate can be updated iteratively online:

$$\hat{\sigma}_{k+1}^2 = \frac{k\,\hat{\sigma}_k^2 + (x_{k+1} - x_{k+1}^{\mathrm{p}})^2}{k+1}. \tag{4}$$

With this variance estimation, we are then able to combine different sources of information to generate more state estimations. In particular, we will combine the predictions with the sensory feedback in an information-theoretic fashion. We expect that the filtered sensor information will (i) generate more accurate learning signals and will thus (ii) improve behavioral control.

2.3 XCSF Settings

In all experiments reported in this paper, we used the XCSF setup that was reported in [7]. In particular, we use XCSF with rotating, hyper-ellipsoidal conditions, that is, general Gaussian kernels, which were first introduced in [6]. Moreover, each classifier learns a linear prediction by means of recursive least squares approximation. In particular, the parameters of XCSF were set as follows: $N = 2000$, $\alpha = 1$, $\beta = 0.1$, $\delta = 0.1$, $\nu = 5$, $\chi = 1$, $\theta_{\mathrm{del}} = 20$, $\theta_{\mathrm{sub}} = 20$. The GA application threshold $\theta_{GA} = 200$. The target error is set to $\varepsilon_0 = 10^{-4}$. The mutation probability for each element of the condition is set to one over the number of alleles, e.g., $\mu = 1/5$ for the two-dimensional case and $\mu = 1/35$ for the seven degree of freedom arm. If mutated, center values are mutated within the receptive field bounds, the stretches are decreased or increased in size maximally doubling or halving their current size; the angles are uniformly changed by maximally 45°. The initial radius of receptive fields is taken uniformly random from $[0.01, 1]$. Uniform crossover, GA subsumption, and tournament selection with $\tau = 0.4$ are applied. Condensation [24] commences after 80% of the learning iterations.

3. FILTERING SENSORY INFORMATION

With the capability of generating predictions and estimating the confidence in these predictions by means of variance estimates, we now proceed in specifying how the sources of information are combined online in the proposed setup. We first provide general background about combining stochastic information. Next, we detail Kalman filtering and particularize how it is applied in the XCSF setting. Finally, we specify the exact iterative processing of sensory information, predictions, and feedback over time.

3.1 Combining Stochastic Information

First, let us recapitulate a well known methodology for extracting a higher fidelity signal by linear combination of several low-fidelity input signals. Let us therefore assume that we are given n random variables $(X_i)_{i=1\ldots n}$ that all share the same mean $\langle X_i \rangle = \bar{x}$ and have variances $V(X_i) = \sigma_i^2$. We further assume that the variances σ_i^2 are known to us, but the mean $\bar{x}$ is not and shall therefore be estimated by a linear combination

$$\hat{x} = \frac{\sum w_i\,X_i}{\sum w_i},$$

with some suitable weights w_i, with the hope that $\hat{x}$ gives us a better (i.e. lower variance) representation of the true value $\bar{x}$.

It is obvious that $\langle \hat{x} \rangle = \bar{x}$, but we can also calculate the variance of $\hat{x}$ in the extreme case of independent inputs:

$$V_{\mathrm{ind}}(\hat{x}) = \frac{\sum w_i^2\,\sigma_i^2}{(\sum w_i)^2}. \tag{5}$$

In cases where the independence requirement does not hold, we propose to use the weighted average of the variances instead (it can be shown that this always is an upper bound on the value of the true combined variance):

$$V_{\mathrm{avg}}(\hat{x}) = \frac{\sum w_i\,\sigma_i^2}{\sum w_i}. \tag{6}$$

A natural choice for the weights is to set them proportional to the information content of the corresponding random variable, that is, $w_i = \frac{1}{\sigma_i^2}$. In this case, the two formulas become identical up to an additional factor n:

$$V_{\mathrm{ind}}(\hat{x}) = \frac{1}{\sum \frac{1}{\sigma_i^2}} \tag{7}$$

$$V_{\mathrm{avg}}(\hat{x}) = \frac{n}{\sum \frac{1}{\sigma_i^2}} \tag{8}$$

XCSF predicts function values as *weighted* linear combinations of predictions of the subset of classifiers that match the current context. In our implementation, we apply as weight the ones given by the inverse of the variance estimate of each classifier. The joint error variance for its prediction is then computed according to (8).

3.2 Kalman Filtering

Filtering can be seen as a procedure for transforming a noisy time series $(\tilde{x}_n)$ to a smoothed time series $(\bar{x}_n)$ that (hopefully) is closer to the sequence of "true" values (x_n). Kalman filtering does that by elegantly combining the following two sources of information:

- the filtered value $\bar{x}_{n-1}$ from the previous time step, corrected by the predicted change Δx_n^{p} between steps, and
- the sensor readings $\tilde{x}_n$ for the current time step.

The calculation of the new estimates $\bar{x}_n$ and variance V_n can be split into two steps.

Prediction step:

$$\bar{x}_{n|n-1} = x_{n-1} + \Delta_x^{\mathrm{p}} \tag{9}$$

$$V_{n|n-1} = V_{n-1} + \sigma_{n-1}^2 \tag{10}$$

Update step:

$$\bar{x}_n = \bar{x}_{n|n-1} + \beta\,(\tilde{x}_n - \bar{x}_{n|n-1}) \tag{11}$$

$$V_n = (1-\beta)\,V_{n|n-1} \tag{12}$$

For a proper combination of the sensory feedback and the predicted internal state estimation, the smoothing parameter β is used. It can be specified by weighing the respective information contents. Thus, it is given by:

$$\beta = \frac{V_{n|n-1}}{V_{n|n-1} + \sigma^2_{\text{sensor}}}, \qquad (13)$$

where σ_{sensor} denotes the standard deviation of the sensory noise. For $\beta = 1$, there is no filtering at all, while for $\beta = 0$ the new measurement is completely ignored. If the variance of several consecutive delta position estimates $\sigma^2_{n-k}, \ldots, \sigma^2_n$ are all very small, the variance estimate V_n converges quickly towards a value close to 0.

3.3 Filtering with XCSF

It has to be remembered that the prediction and prediction variance estimates of XCSF are always in flux during learning. Thus, these estimates need to be handled with care. In particular, it may well happen that XCSF temporarily over-estimates its confidence in its own predictions. If this occurs, XCSF may enter a rather vicious circle: due to its low variance estimates in its predictions, it may completely over-rule the actual sensory feedback. In effect, learning may stall and XCSF may believe in its own predictions – consequently preventing further learning. To avoid this effect, we introduce a threshold θ_β, which constitutes a lower-bound on β in (13). The overall flow of information is thus modified as follows.

Fig. 1 specifies the information flow in XCSF without filtering, as it was previously employed, for example, in [5, 8, 19]. XCSF interacts with a "controller" module, which reads sensory information from and executes motor commands in the associated arm model. Motor noise and sensor noise may be added in this process. Given the current joint angle state of the arm q_{n-1}, XCSF generates a set of matching classifiers. Using this set and a given desired direction of the hand $\Delta x^*_n = g - \tilde{x}_{n-1}$ towards goal g, XCSF generates a control command Δq^*_n by means of its locally linear, inverse velocity kinematics model. Motor noise is added to this motor command, which is then sent to the arm model. In return, the next angular state q_n is perceived. Note that noise may be added to the perceived angular states, which may also be filtered. However, since the changes in angular states directly depend on the Δq_n motor control signal, simple linear filtering may be applied in this case, independent of XCSF. Thus, we do not add noise to the angular states in this work. Additionally, the consequent location of the end-effector x_n is perceived and noise is added, yielding $\tilde{x}_n$. The information is thus complete to learn from the described interaction using q_{n-1} as the context signal for matching, and $\Delta q_n = q_n - q_{n-1}$ as well as $\Delta\tilde{x}_n = \tilde{x}_n - \tilde{x}_{n-1}$ for the prediction and resulting error and fitness updates.

The described original XCSF setup for learning and goal-directed behavior control is modified as shown in Fig. 2. The added Kalman filtering process essentially filters the successive perceptions of end-effector locations $\tilde{x}_n$ by means of XCSF's forward velocity kinematics model. Previously, the change in end effector state was directly determined by the noisy location signal, that is, $\Delta\tilde{x}_n = \tilde{x}_n - \tilde{x}_{n-1}$. Now the change in the end effector location is calculated by filtering the location feedback $\tilde{x}_n$ with the state prediction generated by the locally linear forward velocity kinematics model of XCSF, that is, $\bar{x}_{n-1} + \Delta x^p_n$, additionally considering the

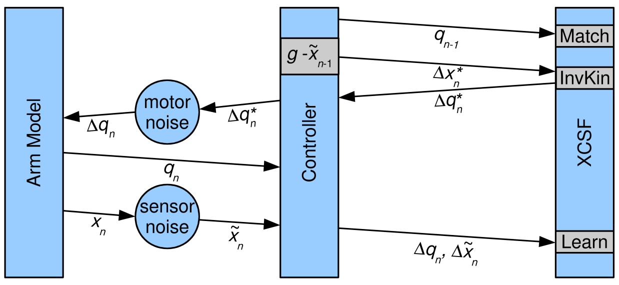


Figure 1: Classic setup for XCSF-based arm control and learning.

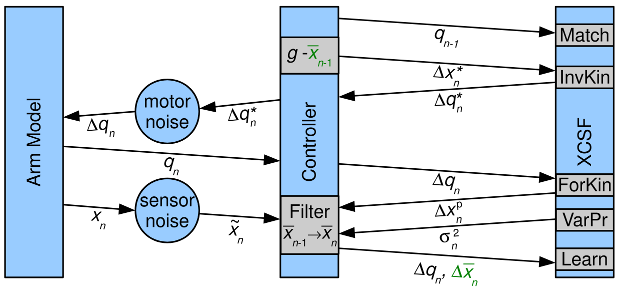


Figure 2: Setup with Kalman filtering to improve robustness against noise on the hand position sensors, and thus to improve the noise-robustness of XCSF during learning and arm control.

variance estimate σ^2_n of the local model. In consequence, XCSF can exploit its predictive knowledge controlling the arm based on the filtered internal hand location estimate $\bar{x}_n$ and learning from the filtered location changes $\Delta\bar{x}_n$ in each iteration n. Thus, we expect to reap the benefits of the filtering process for both, learning and movement planning.

4. RESULTS

To evaluate the approach and scrutinize the dependency on the threshold θ_β, which controls the maximum influence of XCSF's predictive knowledge as detailed above, we tested performance on an exemplary two degree of freedom robot arm system, working in a two-dimensional plane. Finally, we also provide results on an anthropomorphic seven degree of freedom robot arm. All the reported results were carried out on a kinematic arm simulation.

4.1 Experimental Setup

The planar 2-DoF arm with limb lengths $l_1 = 1.4$, $l_2 = 0.9$ and angular ranges $q_1, q_2 \in [-\pi, \pi]$ was used to explore the potential of the filtering approach. The maximum rotation velocity per joint is restricted to 0.01 radians per iteration. The kinematic specifications of the 7-DoF anthropomorphic arm are illustrated in Fig. 3. The arm has a total length of 100 cm. Rotation axes $q_1, \ldots, q_7$ are drawn as dashed lines; the two rotary joints are depicted with a circle. Joint angles are restricted to $q_1 \in [-1.0, 2.9]$, $q_2 \in [-1.5, 1.5]$, $q_3 \in [-1.0, 1.0]$, $q_4 \in [-0.0, 2.8]$, $q_5 \in [-0.7, 1.0]$, $q_6 \in [-1.5, 1.5]$,

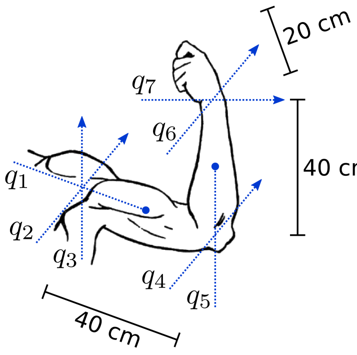


Figure 3: The seven degree of freedom anthropomorphic arm setup used in our simulation.

$q_7 \in [-0.5, 0.7]$. Similar to a human arm, the shoulder has three DoFs, the elbow allows for two DoFs, and the wrist offers another two DoFs.

During learning, the involved controller generates goal locations uniformly randomly within the workspace of the arm. From the beginning, XCSF utilizes its internal model to strive for the given goal. A goal is assumed to be reached if the arm end-effector reaches a distance smaller than 5% of its arm length to the goal. If the goal is reached or 200 iterations have passed without reaching the goal, a new goal location was generated. In this way, well-balanced exploration was achieved, as proposed elsewhere [19]. Moreover, 0.05 standard deviations in radians of angular motor noise were added to the motor commands, as also done elsewhere [19]. This is particularly necessary early during learning to foster sufficient exploration.

4.2 Quality Criterion

To track learning performance, offline testing phases were inserted. In an offline testing phase, XCSF-learning was switched off, but filtering was performed as usual. Each testing phase contained 100 episodes with randomly chosen start/goal-pairs (the same in all testing phases). The Kalman filter was set to its initial state (infinite variance) at the beginning of each episode. Each episode consisted of 30 time steps. During this testing period, we evaluated the angle between the direction towards the target and the actually performed change in hand position as a quality criterion. The measure includes the effect of motor noise but excludes uncertainty due to sensor noise. We report the mean of this angle (averaged over all steps of all episodes of a testing phase) as a measure for the quality of the system's path planning capability.

4.3 Validity of Error Variance Estimation

First, we investigated how much our conservative online prediction error measure overestimates the true value. As demonstrated in Fig. 4, during the main phase of learning up to about 110 k learning steps, the online estimate agrees very well with the true error variance (which was measured in the offline phase). Thus, XCSF estimates its prediction error rather accurately, when compared to the sample real error between filtered estimate and actual real value.

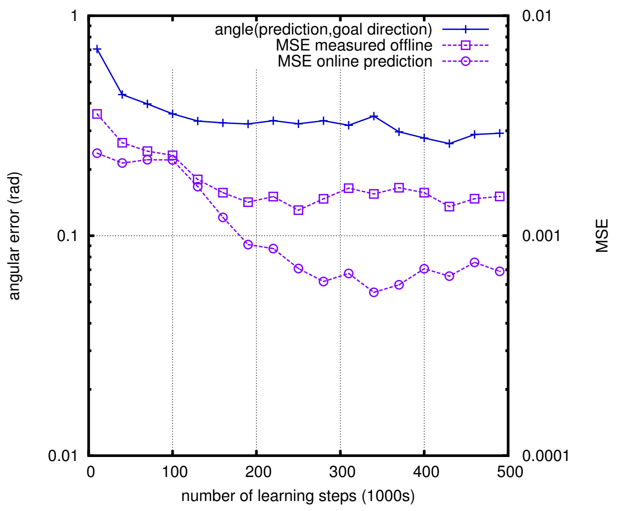


Figure 4: The online predicted and offline measured error variance (noise level $\sigma_{\text{sensor}} = 0.01$) shows that XCSF is able to correctly approximate this value.

4.4 Effect of Threshold Parameter θ_β

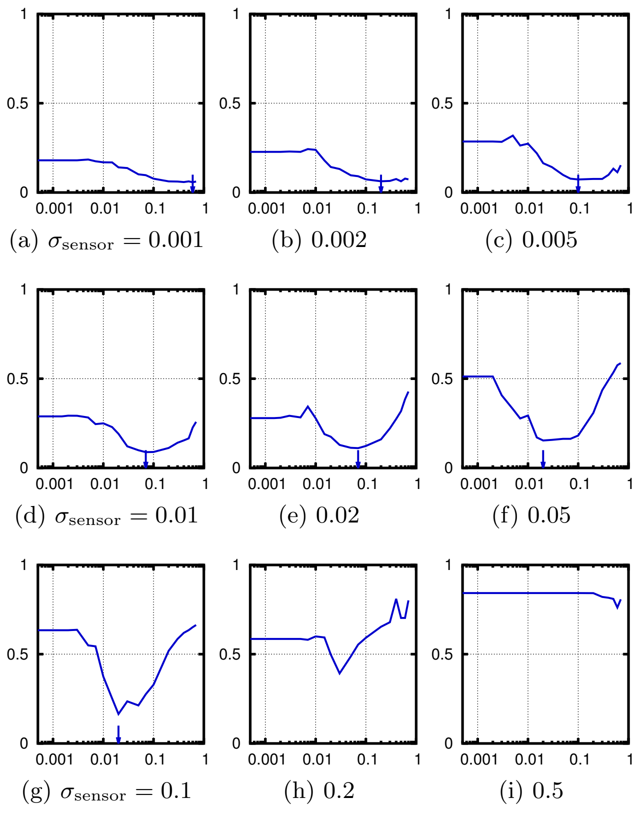
Next, we measured the effect of the proposed threshold θ_β on performance for different levels of position sensor noise. In the plots of Fig. 5 the mean angular quality measure (averaged over ten independent experiments and over the offline phases between 300 k and 400 k learning steps) is shown as a function of θ_β for different levels of sensor noise σ_{sensor}. We find that if the noise is almost negligible, values of θ_β close to 1 are advantageous. For higher levels of sensor noise, however, there is a clearly distinguishable optimum, which moves gradually to smaller values as σ_{sensor} increases (highlighted with arrows). When further increasing the noise level above a critical value (around 0.05), the place of the optimum ceases shifting and, instead, the depth of the valley starts reducing. For even higher noise levels, the performance finally completely collapses for all choices of θ_beta.

In effect, these results show that XCSF benefits from Kalman filtering with appropriate θ_β threshold, but may also suffer from delusional cycles, in which it starts to believe too strongly in its own, inaccurate predictions (small θ_β values in Fig. 5). The smaller the actual noise in the system, the higher the need to prevent XCSF from believing too strongly in its own predictions. Moreover, the results show that XCSF can deal with a considerable amount of noise when incorporating Kalman filtering principles. Only when adding more than 20% standard deviation of the actual arm length noise to the location sensor, hardly any performance improvement is deducible any longer.

4.5 Noise Robustness

In Fig. 6, we give the time course of XCSF performance for typical cases. Fig. 6(a) is the baseline case without filtering. At a noise level 0.01, learning of the baseline system already fails completely. With Kalman filtering (Fig. 6(b)) and optimal θ_β (from Fig. 5) XCSF still learns well at this noise level and even when the noise level is increased by a factor of 5.

Finally in Fig. 7 we have assembled the summary of all



(a) $\sigma_{\text{sensor}} = 0.001$ (b) 0.002 (c) 0.005

(d) $\sigma_{\text{sensor}} = 0.01$ (e) 0.02 (f) 0.05

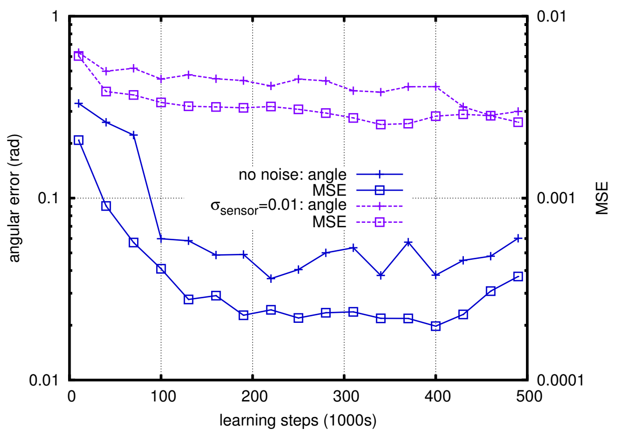
(g) $\sigma_{\text{sensor}} = 0.1$ (h) 0.2 (i) 0.5

Figure 5: The angular error measure (vertical axis) is plotted against parameter θ_β (horizontal axis) for different noise levels σ_{sensor}. (a)-(g) Optimal θ_β values are marked with arrows.

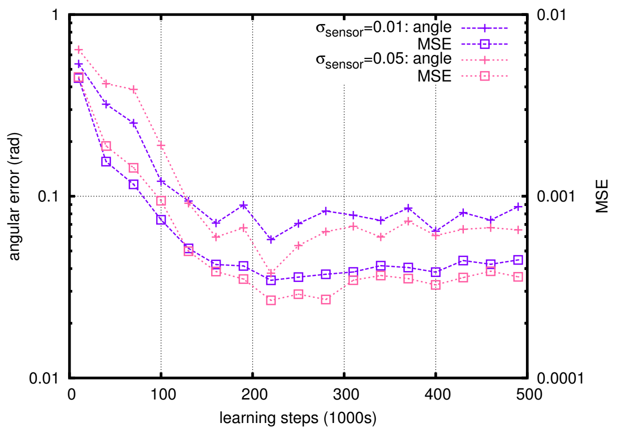
experiments with the two degree of freedom arm in a single graph. It shows the angular performance for different noise levels for the baseline system as well as the Kalman filtering systems with and without threshold θ_β. It can be seen that pure Kalman filtering reduces performance in the regime of moderate sensor noise levels. Only for higher noise levels it shows some benefit, but only when its performance has already become unacceptably bad. The introduction of a filtering threshold θ_β completely changes the picture. For small σ_{sensor}, the baseline is reproduced and the performance stays acceptable for a much wider range. Taking an angular quality of 0.2 as acceptance limit, we see that our proposed system can cope with a noise level of 0.1 while the baseline system starts failing at 0.05 already.

4.6 7-DoF Anthropomorphic Arm

We now take the analysis one step further and run the same experiments on the anthropomorphic, seven degree of freedom arm that acts in a three-dimensional space. The increased dimensionality not only increases the learning complexity, but also affects the control performance. While it is comparably simple to reach high precision for a two joint planar arm, the 7D task generally shows higher error measures. Nonetheless, the XCSF learning classifier system is able to learn a suitable representation that is sufficient for accurate control.



(a) Baseline



(b) Kalman Filtering

Figure 6: Learning progress during the course of a full training run. (a) XCSF without filtering and noise levels $\sigma_{\text{sensor}} = 0$ and 0.01; noise deteriorates performance. (a) Kalman filtering with optimal β bounds; good performance is achieved for $\sigma_{\text{sensor}} =$ 0.01 and even higher noise magnitude of 0.05.

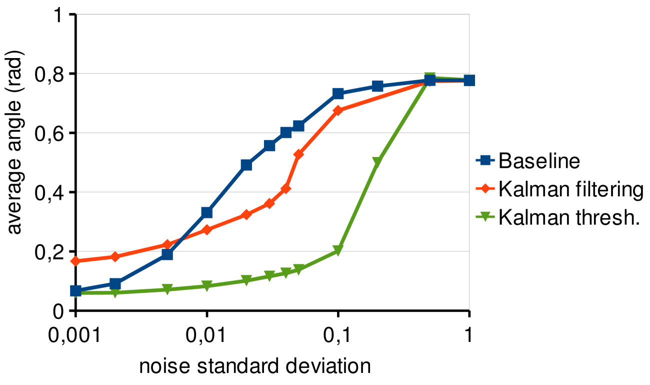


Figure 7: Comparison of baseline performance (no filtering), Kalman filtering, and Kalman filtering with optimal θ_β. The latter approach allows for accurate control with more than ten times higher noise levels.

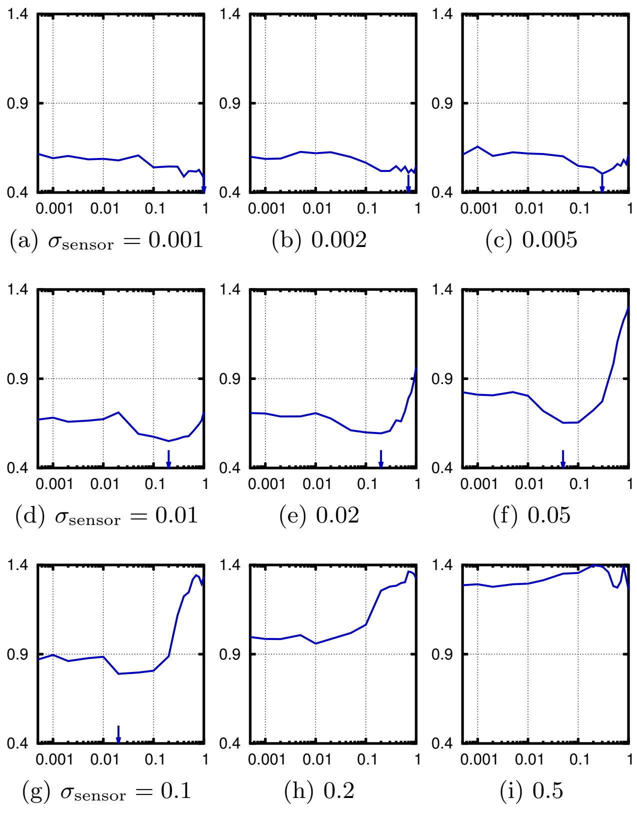


Figure 8: The angular error measure on the 7-DoF arm is plotted against parameter θ_β for different noise levels σ_{sensor}. (a)-(g) Optimal θ_β values are marked with arrows.

Again, the optimal θ_β bounds are unknown and intense search reveals similar results than previous experiments for the 2-DoF arm. For very small sensor noise $\sigma_{\text{sensor}} = 0.001$ in Fig. 8(a) the filtering approach cannot improve performance, which is confirmed by an optimal $\theta_\beta = 1$. With increasing sensor noise, the valley of optimal filtering shifts to the left until performance collapses for noise levels beyond 0.1 in Fig. 8(g)-(i). The general trend of the optimal θ_β bound is summarized in Fig. 9 for the 2-DoF arm and the 7-DoF arm.

The Comparison of baseline XCSF, XCSF with Kalman filtering, and XCSF with Kalman filtering including the approximately optimal bound β in Fig. 10 reveals that successful control is achieved with large magnitudes of sensor noise when XCSF's knowledge is used to refine subsequent sensor readings.

5. SUMMARY AND CONCLUSION

The goal of this work was to improve the performance of XCSF in arm control scenarios. To do so, we have introduced information content based weighting of classifier predictions. Moreover, the prediction error variance needed to be estimated online, as was proposed previously in [16]. To incorporate Kalman filtering, the evolving forward velocity kinematics model of XCSF was used to generate state predictions and thus filter the incoming sensory information. The performance results have shown that XCSF may run into a self-delusional spiral, in which it may over-filter the incoming sensory information. With the setting of an appropriate threshold to prevent this effect, it was shown that XCSF can face partially more than ten-times higher noise levels, dependent on the considered performance criterion. Similar results for the anthropomorphic, seven degree of freedom arm model confirmed the advantage of this approach.

At the moment, we study options of replacing the threshold θ_β with more inherent XCSF measures, by estimating the variance even more pessimistically than at the moment. Also, the relation to Bayesian information processing with XCSF is explored further. We expect that this approach will be applicable also to other prediction scenarios involving noisy sensors and possibly actuators. The setup, however, will generally be the same and the message of this paper ap-

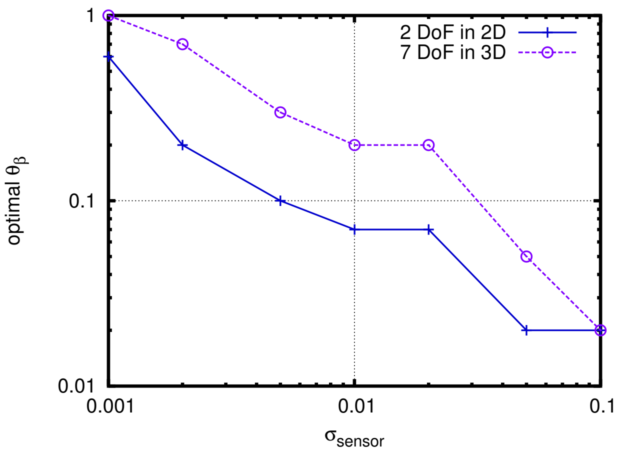


Figure 9: Optimal threshold parameter θ_β as function of noise level σ_{sensor} for the 2DoF robot arm in 2D and the 7DoF arm in 3D.

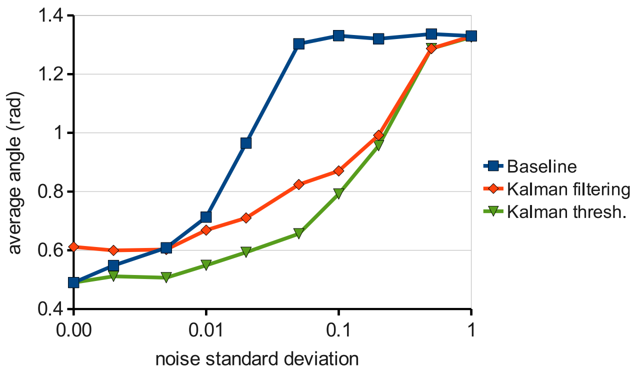


Figure 10: The control error is plotted against different levels of sensor noise for the three systems under consideration: baseline XCSF, XCSF with Kalman filtering, and XCSF with bounded Kalman filtering. With noisy sensors, the latter approach still allows for accurate control where the baseline system fails.

pears to apply in a general sense: given XCSF is predicting sensory information changes over time, its learning performance, its predictive capabilities, and possibly (dependent on the setup) its inverse control capabilities can be improved by exploiting the predictive knowledge of XCSF, filtering the incoming sensory information online.

Acknowledgments

The authors acknowledge here the funding from the Emmy Noether program (German Research Foundation, DFG, BU1335/3-1) and thank the Cognitive Modeling team.

6. REFERENCES

[1] A. Ben-Israel and T. N. Greville. *Generalized Inverses: Theory and Applications.* Springer, 2003.

[2] E. Bernadó-Mansilla and J. M. Garrell-Guiu. Accuracy-based learning classifier systems: Models, analysis, and applications to classification tasks. *Evolutionary Computation*, 11:209–238, 2003.

[3] L. Bull, editor. *Applications of Learning Classifier Systems.* Springer-Verlag, 2004.

[4] M. V. Butz. *Rule-Based Evolutionary Online Learning Systems: A Principled Approach to LCS Analysis and Design.* Springer-Verlag, Berlin Heidelberg, 2006.

[5] M. V. Butz and O. Herbort. Context-dependent predictions and cognitive arm control with XCSF. *Genetic and Evolutionary Computation Conference, GECCO 2008*, pages 1357–1364, 2008.

[6] M. V. Butz, P. L. Lanzi, and S. W. Wilson. Hyper-ellipsoidal conditions in XCS: Rotation, linear approximation, and solution structure. *Genetic and Evolutionary Computation Conference, GECCO 2006*, pages 1457–1464, 2006.

[7] M. V. Butz, P. L. Lanzi, and S. W. Wilson. Function approximation with XCS: Hyperellipsoidal conditions, recursive least squares, and compaction. *IEEE Transactions on Evolutionary Computation*, 12:355–376, 2008.

[8] M. V. Butz, G. K. M. Pedersen, and P. O. Stalph. Learning sensorimotor control structures with XCSF: Redundancy exploitation and dynamic control. *Genetic and Evolutionary Computation Conference, GECCO 2009*, pages 1171–1178, 2009.

[9] M. V. Butz, K. L. Reif, and O. Herbort. Bridging the gap: Learning sensorimotor-linked population codes for planning and motor control. *International Conference on Cognitive Systems*, CogSys 2008, 2008.

[10] M. V. Butz and P. O. Stalph. Modularization of xcsf for multiple output dimensions. *Genetic and Evolutionary Computation Conference, GECCO 2011*, GECCO 2011, 2011.

[11] J. Drugowitsch and A. Barry. A formal framework and extensions for function approximation in learning classifier systems. *Machine Learning*, 70:45–88, 2008.

[12] J. Drugowitsch and A. M. Barry. Mixing independent classifiers. *Proceedings of the 9th annual conference on Genetic and evolutionary computation*, GECCO 2007:1596–1603, 2007.

[13] J. H. Holland and J. S. Reitman. Cognitive systems based on adaptive algorithms. In D. A. Waterman and F. Hayes-Roth, editors, *Pattern directed inference systems*, pages 313–329. Academic Press, New York, 1978.

[14] P. L. Lanzi, D. Loiacono, S. W. Wilson, and D. E. Goldberg. Prediction update algorithms for XCSF: RLS, Kalman filter and gain adaptation. *Genetic and Evolutionary Computation Conference, GECCO 2006*, pages 1505–1512, 2006.

[15] P. L. Lanzi, D. Loiacono, S. W. Wilson, and D. E. Goldberg. Generalization in the XCSF classifier system: Analysis, improvement, and extension. *Evolutionary Computation*, 15:133–168, 2007.

[16] D. Loiacono, J. Drugowitsch, A. Barry, and P. L. Lanzi. Analysis and improvements of the classifier error estimate in XCSF. In J. Bacardit, E. Bernadò-Mansilla, M. V. Butz, T. Kovacs, X. Llorà, and K. Takadama, editors, *Learning Classifier Systems*, volume 4998 of *Lecture Notes in Computer Science*, pages 117–135. Springer Berlin / Heidelberg, 2008.

[17] O. Sigaud and J. Peters, editors. *From Motor Learning to Interaction Learning in Robots.* Springer, 2010.

[18] O. Sigaud, C. Salaun, and V. Padois. On-line regression algorithms for learning mechanical models of robots: a survey. *Robotics and Autonomous Systems*, 59(12):1115–1129, December 2011.

[19] P. Stalph and M. V. Butz. Learning local linear jacobians for flexible and adaptive robot arm control. *Genetic Programming and Evolvable Machines*, pages 1–21, 2011. 10.1007/s10710-011-9147-0.

[20] P. O. Stalph and M. V. Butz. Documentation of JavaXCSF. COBOSLAB Report Y2009N001, Cognitive Bodyspaces: Learning and Behavior, University of Würzburg, Germany, 2009.

[21] P. O. Stalph, J. Rubinsztajn, O. Sigaud, and M. V. Butz. A comparative study: Function approximation with LWPR and XCSF. In *Proceedings of the 12th annual conference on genetic and evolutionary computation*, pages 1863–1870. ACM, 2010.

[22] S. Vijayakumar, A. D'Souza, and S. Schaal. Incremental online learning in high dimensions. *Neural Computation*, 17:2602–2634, 2005.

[23] S. W. Wilson. Classifier fitness based on accuracy. *Evolutionary Computation*, 3(2):149–175, 1995.

[24] S. W. Wilson. Generalization in the XCS classifier system. *Genetic Programming 1998: Proceedings of the Third Annual Conference*, pages 665–674, 1998.

[25] S. W. Wilson. Get real! XCS with continuous-valued inputs. In P. L. Lanzi, W. Stolzmann, and S. W. Wilson, editors, *Learning classifier systems: From foundations to applications (LNAI 1813)*, pages 209–219. Springer-Verlag, Berlin Heidelberg, 2000.

[26] S. W. Wilson. Function approximation with a classifier system. *Genetic and Evolutionary Computation Conference, GECCO 2001*, pages 974–981, 2001.

[27] S. W. Wilson. Classifiers that approximate functions. *Natural Computing*, 1:211–234, 2002.