Learning Sensorimotor Control Structures with XCSF: Redundancy Exploitation and Dynamic Control

Martin V. Butz  
butz@psychologie.uni-wuerzburg.de

Gerulf K.M. Pedersen  
gerulf@psychologie.uni-wuerzburg.de

Patrick O. Stalph  
stalph@informatik.uni-wuerzburg.de

Department of Cognitive Psychology  
University of Würzburg  
97070 Würzburg, Germany

ABSTRACT

XCS has been shown to be an effective genetics-based classification, datamining, and reinforcement learning tool. The systems learns suitable, compact, maximally general problem solutions online. In the robotics and cognitive systems domains, however, applications of XCSF are very sparse and mostly restricted to small, symbolic problems. Recently, a sensorimotor XCSF system was applied to cognitive arm control. In this paper, we show how this XCSF-based arm-control mechanisms can be extended (1) to efficiently exploit redundant behavioral alternatives and (2) to guide the control of dynamic arm plants. The XCSF system encodes redundant alternatives in its inverse control representations and resolves the encoded redundancies dependent on current constraints—such as arm posture preferences—on the fly. An adaptive PD controller translates the XCSF-based direction and distance commands into actual motor commands for dynamic arm control. We apply the complete system to the control of a simulated, physical arm with three degrees of freedom in a two-dimensional environment and to a simulation of the industrial KR16 Kuka arm with ODE-based physics engine.

1. INTRODUCTION

The genetics-based machine learning system XCS [15, 16] combines gradient-descent mechanisms with evolutionary computation techniques. In order to avoid convergence to locally optimal representations a distributed evolutionary mechanism is applied. The gradient-based approach maximizes local approximation accuracy. Generally, XCS is designed to generate problem solutions that partition the problem space in order to generate locally optimal solution approximations with a maximally general problem partitioning that covers the complete problem domain [8]. In comparison to other machine learning techniques, XCS converts problem feedback into problem partitions by means of the evolutionary mechanism—avoiding (1) convergence to local optima and (2) the requirement that problem feedback can be directly converted to structural adaptations.

The XCSF system is the XCS version that is applied to real-valued domains in which real-valued inputs are converted to real-valued outputs—yielding an effective function-approximation technique [17]. It has been shown that XCSF learns to adjust its problem partitioning structures, that is, classifier conditions, in such a way as to approximate function output values maximally generally and maximally accurately. XCSF has been shown to outperform pure clustering techniques and perform comparably to statistical techniques that generate comparable problem solutions [4]. The advantage of XCSF, however, is its applicability to problem domains in which the condition prediction mapping is unknown.

Most recently, XCSF was successfully applied to a kinematic robot arm control task, in which the problem input was split into a context input, which was used to partition the problem space, and a prediction input, which was used to generate a forward model of the arm kinematics [2]. In this setting, the statistical techniques are not applicable but XCSF has still shown to yield reliable forward model predictions as well as to be able to generate redundancy-resolving inverse control commands. Two challenges, however, had not been further addressed so far: (1) the redundancy-resolving mechanism was realized by the imposition of somewhat arbitrary constraints and (2) the controlled arm was a kinematic arm, leaving the problem of controlling a dynamic system aside. These two challenges are addressed in the present work. Particularly, we show that redundancy can be resolved in such a way that current behavioral constraints are flexibly taken into account and
that the XCSF control system can be effectively coupled with an adaptive PD controller in order to control dynamic arm plants.

The remainder of the paper is structured as follows. First, we give an overview of the XCSF arm control system as introduced elsewhere [2]. Next, we provide the mathematical framework to resolve redundancies on the fly and show how to impose additional constraints on the system. We evaluate the resulting system with several differing constraints on the kinematic robot control task. In Section 4 we couple the XCSF system with dynamic arm control plants and validate continued reliable and flexible control.

2. XCSF ROBOT ARM CONTROL SYSTEM

The XCSF control system [17] evolves a population of condition-prediction classifiers, which yields piecewise linear function approximation. Essentially, the system clusters the problem input domain by means of its condition structures in order to develop maximally accurate linear approximations.

2.1 XCSF in a Nutshell

XCSF is the function approximation version of the XCS system [15]. Instances of the to-be learned function are sampled iteratively and the system learns from the sampled problem instances. Given a particular problem instance, the classifier population is scanned for the matching classifiers—those whose conditions are satisfied by the given input. Given the resulting match set of matching classifiers, a prediction of the output of this problem instance can be generated. The comparison between predicted and actual output yields the training signal, which is used to directly adjust the predictions of the classifiers by means of gradient-descent mechanisms. Moreover, the error in the prediction value is used to estimate the actual accuracy of each matching classifier and to derive a fitness estimate of each classifier, which reflects the scaled relative accuracy.

Besides the prediction and fitness adjustments, a steady-state, niche genetic algorithm is applied: Two classifiers are selected for reproduction in the match set. If the number of resulting classifiers in the population exceeds the maximal population size \( N \), classifiers are deleted from the whole population. Due to the niche-based selection but population-based deletion, a generalization pressure is added apart from the fitness pressure, which stresses the reproduction of more accurate classifiers. Overall, the system is designed to yield classifiers that yield maximally accurate predictions while covering maximally general problem subspaces. The population as a whole represents the problem solution in the form of piece-wise linear, overlapping problem sub-solutions. Further information on the XCS mechanism and the XCSF system in particular can be found elsewhere [4, 17].

Recent enhancements of XCSF have shown that improvements in the prediction representations and approximation mechanisms can yield improved system performance [10]. Moreover, modifications in the condition structures can yield improvements in the learned problem space partitions and thus improvements in the system’s scalability [4, 11]. The base system we are using here is XCSF with rotating, ellipsoidal condition structures and linear predictions that are approximated by means of recursive least squares (RLS) [4]. The Java code of the applied system is available online [14].

2.2 XCSF Control System

To apply XCSF as a control system, condition and prediction mechanisms were split such that the conditions partition the current state of the controlled system in order to be able to accurately predict motor-induced system changes. The original work on XCSF as a sensorimotor control system also emphasized the relation to cognitive structures and neural control mechanisms [2]. In fact, the resulting XCSF system learns sensorimotor control structures in that it correlates motor-induced posture with location changes. To optimize these correlations, XCSF partitions the relevant posture space in such a way as to optimize the precision in the sensorimotor correlations. This encoding scheme complies with neuroscientific findings, which suggest that the brain structures spatial population codes (such as population codes of body postures) highly pro-actively, optimizing correlated sensorimotor control structures [6].

In the present arm control applications, XCSF partitions the arm posture space, which is encoded in the form of joint angles, in order to optimize the predictions of arm position changes given arm posture changes. That is, it correlates posture-based movements, which may be induced by motor control commands, with hand location changes. During learning, given a certain posture state \((\delta_1, \delta_2, \delta_3)\), XCSF creates its match set as usual. However, to determine its prediction the system requires movement information in the form of changes in joint angles \((\delta_1, \delta_2, \delta_3)\), which are converted into changes in hand locations \((\Delta_x, \Delta_y)\). Thus, in this setting, XCSF for control learns two linear approximations for the two changes in hand locations, essentially adapting six weights \(w_{x1}, w_{x2}, w_{x3}, w_{y1}, w_{y2}, w_{y3}\).  

Adaptation is done by the usual recursive least squares (RLS) mechanism [10]. Error and fitness estimates are adapted based on the average error in the two linear predictions. The evolutionary mechanism works as usual (cf. [15, 10, 4]).

3. INVERSE KINEMATIC CONTROL

For the inverse control process, the predictions learned by XCSF have to be inverted in order to yield desired posture changes given currently desired hand location changes. Since XCSF learns two linear equations, given a desired \((\Delta_x, \Delta_y)\) there is no unique solution for the corresponding posture changes \((\delta_1, \delta_2, \delta_3)\). To determine the actually invoked posture change, somewhat arbitrary constraints were used previously by either setting iteratively randomly one posture change to zero or by resolving the redundancy with a NN-based forward-backward approximation approach [2]. However, as shown elsewhere [3], redundant behavioral alternatives should actually be exploited to increase behavioral flexibility—as they are exploited in animals and humans. Thus, we show that the developed XCSF coding can be used to exploit redundancy by employing actual behavioral constraints, such as joint location preferences.

3.1 Redundancy Resolution Mechanism

The redundancy resolution problem is defined by the two linear equations, which specify the expected change in hand location \((\Delta_x, \Delta_y)\) given changes in joint angles \((\delta_1, \delta_2, \delta_3)\). In essence, an over-specified linear system needs to be solved.
in this case. Given the current hand location \((h_x, h_y)\) and a given goal location \((g_x, g_y)\), the desired hand direction can be set to \((\Delta_x = g_x - h_x, \Delta_y = g_y - h_y)\). Thus, the two linear predictions for each change in hand location are specified as follows:

\[
\begin{align*}
wx_1 \delta_1 + wx_2 \delta_2 + wx_3 \delta_3 &= \Delta_x = (g_x - h_x), \\
wy_1 \delta_1 + wy_2 \delta_2 + wy_3 \delta_3 &= \Delta_y = (g_y - h_y).
\end{align*}
\]

Ignoring the special case of parallel hyperplanes, given \(\Delta_x\) and \(\Delta_y\), the solution to this set of equations is a line in three dimensional delta posture space.

To determine the actual line, we fix one variable (any \(\delta_i\)) with two arbitrary values (for computational simplicity we use zero and one), which yields two points on the desired line. Given these two points \(p_1, p_2\) in the posture space the line is specified by the vector equation

\[f(\mu) = p_1 + \mu(p_2 - p_1).\]

This line essentially specifies all postures that are expected to suitably yield the desired hand location change.

Given now a secondary goal \(s\), such as a particularly desirable change in the actual posture, the point \(p\) on this line can be determined that is closest to this secondary goal \(s\). Thus, we have to minimize the distance from \(s\) to the line. Point \(p\) may be specified by \(\mu_{goal}\), which can be determined by the following projection calculation:

\[\mu_{goal} = \left[ (s - p_1) \cdot (p_2 - p_1) \right]/\left[ ||p_2 - p_1||^2 \right],\]

which yields the actual posture change vector \(p = f(\mu_{goal})\). Given the predictions are accurate, the determined inverse change in posture space will yield the desired change in hand location while adhering to the imposed constraint \(s\) — minimizing the difference in change in joint angles in comparison to the desired change in joint angles specified in \(s\). The resulting command may now be either sent directly to a kinematic arm plant or it may be further processed by a PD controller, which in turn yields control torques, given a dynamic arm plant. Before turning to the dynamic control challenge, we first evaluate learning and the specified redundancy resolution mechanism on a kinematic arm plant.

### 3.2 Experimental Evaluation

We test the system on a kinematic arm plant with three joints. The limbs (starting from the shoulder joint) have lengths 6, 5 and 4 spatial units and can each rotate from \(-3\) up to \(+3\) radians. For XCSF, postures and locations are normalized to \([0, 1]\) range. In contrast to [2], learning was done on the actual simulation moving the arm iteratively to randomly generated goals in posture space and learning from the observed movements. Once a goal posture is reached, a new random one is generated and learning continues with the previous goal posture used as initial posture. To prevent learning from zero movements and ensure a more uniform problem space sampling, only every tenth instance is used for learning and the instances are further filtered using only those that yield a change in hand location of at least \(0.04\) units.\(^2\) Figure 1 shows the typical prediction performance curve of XCSF in this setting: The system learns a good function approximation mapping fast and continues to improve the approximation over time. After 108k iterations, condensation with closest classifier matching is applied [4] showing that the system significantly decreases its population size (number of distinct classifiers) while maintaining an accurate mapping.

Every 2500 iterations, the XCSF-based control capabilities are tested using the following setup. The posture space is iterated uniformly such that step \(n\) and \(n + 1\) have a small distance in posture space (moving only one of the joints by a certain fraction). A start-goal combination consists of

\[\text{start} = \text{posture}(n) \text{ and goal} = \text{handlocation}(\text{posture}(n+1))\]

Thus, the greedy XCSF-based control mechanism does not face deadlock situations if the shortest hand movement is chosen. Since many start-goal combinations are near the borders of the posture space, which is less frequently sampled during learning, we test the inverse control with different border fraction settings, where, for example, a border fraction of 0.1 means that 10% of the reachable space of each joint on both extreme sides (flexed and bent) is not tested. For the evaluations, we monitor the number of start-goal combinations bridged successfully as well as path efficiency, which is determined by dividing the Euclidean distance from start to goal hand location by the actual distance traveled by the hand. The path efficiency reported is the average of all ten independent runs of all 124 test cases—testing five equally distributed postures for each joint with the next corresponding hand location as described above.

We first test XCSF without any particular additional constraint in the described setting. The default constraint to resolve redundancy is the null vector, which corresponds to a constraint that always prefers minimal joint angular movement. Figure 2 shows performance of XCSF on the kinematic arm with border fraction 0.0, 0.1, and 0.2. The border regions pose a difficulty for XCSF due to the unequally distributed sampling caused by the online iterative instance sampling approach. Nonetheless, particularly for the inner \(0.5, \alpha = 1, \epsilon_B = 0.1, \nu = 5, \theta_{Q_A} = 50, \tau = .4, \chi = 1.0, \mu = 0.05, \eta = 5, \theta_{del} = 20, \delta = 0.1, \theta_{sub} = 20.\)

\(^2\)All reported results are generated from ten independent runs. If not stated differently, the XCSF parameter settings are those used elsewhere [2], which are \(N = 6400, \beta = \eta = 0.5, \alpha = 1, \epsilon_B = 0.1, \nu = 5, \theta_{Q_A} = 50, \tau = .4, \chi = 1.0, \mu = 0.05, \eta = 5, \theta_{del} = 20, \delta = 0.1, \theta_{sub} = 20.\)
areas, XCSF reaches 100% performance and executes efficient execution paths to the goal, that is, the hand moves nearly on a straight line from start to goal. Thus, XCSF can generally learn suitable control commands in the arm control task. Next, we evaluate if and how additional movement constraints may alter the behavior of the arm.

We impose two types of constraints on the redundancy resolution mechanism: a posture constraint and a joint angle constraint. In the first case, a particular posture is the most desirable state of the system. Thus, upon redundancy resolution, the system compares the current system posture with the preferred posture and uses the difference between preferred and current posture as its constraint vector $s$. In the second case, if one joint angle of a particular joint $i$ is preferred without further constraints, the difference between preferred and current joint angle is used as the corresponding entry $s_i$, whereas the other entries in $s$ are set to zero. Thus, a movement towards the joint angle is preferred, while the other joint angles are preferably not moved at all.

Figure 3 compares the performance of XCSF with zero movement constraint (No Const) with the performance of XCSF with preferred posture $(0, 0, 0)$ (Const 000) and with preferred joint angle 0 of the shoulder joint (Const 0J) in the borderfraction=0.1 setting. The graph shows that reachability as well as path efficiency are not significantly affected. This confirms that the redundancy resolution mechanism works flexibly without affecting the actual goal pursuance mechanism. Figure 4, however, shows that the constraint mechanism does affect the behavior of the arm. When comparing the distance of the final posture reached with posture $(0, 0, 0)$, it can be seen that posture constraint yields final postures that are closer to posture $(0, 0, 0)$ compared to the joint constraint runs, while those are still closer compared to the base constraint run. With respect to the distance of the constraint shoulder joint, posture and shoulder constraints consistently yield closer final joint angles as well. The results confirm that XCSF triggers arm trajectories that move the arm towards a preferred posture or joint angle where possible, while not affecting the performance to reach the hand goal.

4. DYNAMIC ARM CONTROL

Applying the output from XCSF to a dynamic arm plant requires that either the output of XCSF is a torque that can be used directly in the plant or that a controller is used to translate the output to a usable torque. In this paper the latter approach is used.

The purpose of the controller is to find a suitable output $u(t)$ to the joint actuator given an error signal $e(t)$. This is accomplished by the use of an extension to the classical PD-control structure. For continuous time the normal PD-controller structure is given as

$$u(t) = K_p (e(t) + T_d \dot{e}(t)) \ ,$$

where $K_p$ and $T_d$ are constants representing gain and rate time, respectively, and $\dot{e}(t)$ is the time-derivative of the error $e(t)$ [5]. As the output of XCSF is a movement direction $dir$, the error for a given joint $e_i(t)$ can be directly set to the $dir_i$.

In order to implement the controller in a digital system it is necessary to use the digital equivalent of the controller equation (1). To avoid derivative disruptions when changing desired positions, an implementation that avoids taking the derivative of the reference signal is used:

$$u[k] = K_p \left( e[k] - \frac{T_d}{T_s} (y[k] - y[k-1]) \right) \ ,$$

Figure 2: Results for the inverse kinematic tests with three arm models. The higher the border fraction, the better the results reaching near perfect reachability and path efficiency.
where the square brackets indicate discrete-time indices and \( y[k] \) denotes the actual joint posture at time index \( k \). This implementation is symmetric with regard to the sign of the error.

### 4.1 Gravity Extension

Due to the effect of gravity on the system and the fact that the normal controller implementation does not counteract the pull of gravity, an extension to the normal controller implementation is used.

Extensions to gravity have been previously proposed in different papers (cf. [7]). Those methods add a variable amount of torque to the controller output based on how much gravity induces additional load on the individual joints. The drawback with these methods is that not only the kinematics but also part of the dynamics must be known to determine the added value. For the XCSF-based control system we intended to minimize the influence of knowledge about the dynamics on the parameters, so that a method that only depends on the kinematics was preferred. One way to realize a dependency on the kinematics is gain scheduling [1], which continuously adapts the parameters of the controller based on the system’s kinematics.

In our case, gain scheduling adapts parameters \( K_p \) and \( T_d \) based on the status of the system. By adapting the parameters based on the direction of gravity compared to the configuration as well as the desired movement of the system, it is possible to obtain the desired corrections of the controller to counter the gravitational effect.

The basic principle is best described in 2D, but can also be extended to 3D when required. In such a system it is possible to determine the angle between a horizontal baseline vector \( \vec{b} \), that is perpendicular to the gravity vector \( \vec{g} \), and the vector pointing along the direction of the link for which the joint is to be controlled \( l \). This angle can be used along with the direction of movement to determine a correction factor \( c_i[k] \):

\[
c_i[k] = \begin{cases} 1 + a_i \cos(\theta_i[k]) & \text{for } e_i \cos(\theta_i[k]) \geq 0 \\ 1 - w_i \cos(\theta_i[k]) & \text{otherwise} \end{cases} ,
\]

where parameters \( a_i \) and \( w_i \) determine the amount of correction for moving against or with gravity, respectively. This correction factor will have the highest value when movement is against gravity for \( \theta_i \)-angles \( 0 \) and \( 2\pi \) and the lowest value when moving along gravity at those same angles. When gravity is perpendicular to the movement direction the correction factor will have a value of 1.

The correction factor is then used to adapt gain \( K_p \) and rate time \( T_d \). The corrections made for a given joint \( i \) are

\[
K_p[k] = c_i[k]K_p[k] \quad T_d[k] = \frac{T_d[k]}{c_i[k]} .
\]

Thus, the gain is amplified with the correction factor and the rate time is attenuated accordingly. The reason for also correcting the rate time is to prevent the overall amplification of the derivative term, which from equation (2) can be found to be \( K_pT_d/T_s \).

### 4.2 Velocity Adaptation

For dynamic systems, interactions between different links can have a significant effect on the performance of the simple controllers that might be used. The major challenge is to account for transfers of momentum. The problem arises from the fact that simple controllers usually are implemented independently of each other and thus cannot adequately counter effect link interactions. However, if transfer of momentum is not taken into account, large overshoots of the outer links can occur before they settle in their desired positions.

One way to counteract this problem is by creating an adaptation scheme that limits the speed of the individual links and thus also the amount of momentum that they may transfer. The way such an adaptation scheme has been used to limit the overshoot of the system in this paper is similar to the gravity extension described previously. The only difference lies in the calculation of the correction factor \( c_i[k] \), which for the velocity adaptation is given as

\[
c_i[k] = \max \left\{ 1 - v_i \left| \dot{\theta}_i \right| , f_i \right\} ,
\]

where \( v_i > 0 \) is a coefficient that specifies how fast the correction coefficient should drop proportional to the absolute angular velocity of the joint and \( f_i \), which is confined to the range \([0,1]\], is a lower limit, which ensures that the correction factor does not become too small. Once the correction factor has been calculated, it can be applied to the gain and rate time in the same manner as described in equation (4). It is even possible to use the velocity adaptation in connection with the gravity extension. Since both influences are generally independent, the correction factors can simply be multiplied before the new gains and rate times are determined.

### 4.3 Dynamic Arm Experiments

With a powerful adaptive controller at hand, it remains to be shown if it can be suitably coupled with the XCSF’s learning and inverse control mechanisms. In essence, we apply the same mechanisms and control structures as in the kinematics setting but now extend these to the dynamic control setting. To do so, we extended the kinematic arm with a simulated dynamics engine in order to face the challenge...
During XCSF-based control, hand location goals are given, goal posture and current posture as the error signal which are then fed into the adaptive PD controller. The controller was used to invoke torques that lead the arm to the activated goal posture using the current difference between controller yield much smaller iterative posture and hand location changes in hand locations. Figure 5 shows the improvement of prediction accuracy over time with these settings. XCSF learns increasingly accurate predictions. Once condensation is applied (after 180k learning iterations), a similar accuracy is maintained while the population size of distinct macro classifiers decreases highly significantly.

The question is, however, if the predictions are sufficiently accurate to yield suitable inverse directional control commands and if these commands can be suitably processed by the adaptive PD controller to reach targets accurately. To test the dynamic control performance of the Dynamic XCSF control system, we used the same test setup as in the kinematics case. Figure 6 shows performance of XCSF in the respective border fraction settings. Compared to the purely kinematics control mechanism, control reliability slightly decreases but in the inner area (borderfraction=0.2) still nearly all goals are reached reliably—even after the application of the condensation mechanism. Also path efficiency improves over time and reaches a value of about 0.9, indicating that the hand path to the goal locations is on average 11% longer than the direct path of the hand to the goal. Considering the necessarily elaborate arm control mechanisms, this result indicates reliable and effective control of a dynamic arm plant, which has never been shown to be possible with any XCS(F) system before.

While effective dynamic control is confirmed by the above experiments, we still had to evaluate the constraint resolution mechanism. Thus, we conducted experiments similar to the kinematic arm setting imposing constraints of posture (0, 0, 0) and joint angle 0 on the redundancy resolution mechanism. Figure 7 shows that the redundancy resolution mechanism equally applies in the dynamic XCSF control setting. While the zero movement constraint (No Const) yields final postures furthest away from posture (0, 0, 0), the joint angle constraint yields closer final postures and the posture goal posture (0, 0, 0) and joint angle 0 on the redundancy resolution mechanism.

During training, again random arm postures were generated but the PD controller was used to invoke torques that lead the arm to the activated goal posture using the current difference between goal posture and current posture as the error signal $\varepsilon[K]$. During XCSF-based control, hand location goals are given, which are translated into desired posture changes by XCSF, which are then fed into the adaptive PD controller.

Initially, random arm mechanisms equally apply in the dynamic XCSF control case. Figure 6 shows performance of XCSF in the respective border fraction settings. Compared to the purely kinematics control mechanism, control reliability slightly decreases but in the inner area (borderfraction=0.2) still nearly all goals are reached reliably—even after the application of the condensation mechanism. Also path efficiency improves over time and reaches a value of about 0.9, indicating that the hand path to the goal locations is on average 11% longer than the direct path of the hand to the goal. Considering the necessarily elaborate arm control mechanisms, this result indicates reliable and effective control of a dynamic arm plant, which has never been shown to be possible with any XCS(F) system before.

![Figure 5: XCSF for control is able to learn an accurate functional mapping in the specified arm control task.](image)

![Figure 6: Results for the inverse kinematic tests with three arm models. The higher the border fraction, the better the results reaching near perfect reachability and path efficiency.](image)

Maximum torques applicable were 350, 250, and 150 Nm, respectively for the three joints, which had masses of 12, 10, and 8 kg. The sampling time was set to 1 ms. The initial control parameters were set to $K_{p1} = 4000$, $K_{p2} = 2000$, $K_{p3} = 1000$, $T_{d1} = T_{d2} = T_{d3} = .15$. Adaptation parameters for the PD controller were set to $a_1 = 1.5$, $w_1 = 0.5$, $f_1 = 0.1$, $v_1 = 0.01$. Elements the posture space is explored much less uniformly. Elsewhere [13] it was shown that XCS learning suffers from unbalanced data sets but the detrimental effect can be counteracted upon by increasing the GA application threshold $\theta_{CA}$ with the effect of decreasing learning speed while distributing evolutionary learning effort more uniformly over the problem space. Thus, we increased learning to 200k iterations while setting $\theta_{CA}$ to 200. Moreover, we decreased the error threshold $\varepsilon_0$ to 0.0005 to account for the smaller changes in hand locations. Figure 6 shows the improvement of prediction accuracy over time with these settings. XCSF learns increasingly accurate predictions. Once condensation is applied (after 180k learning iterations), a similar accuracy is maintained while the population size of distinct macro classifiers decreases highly significantly.

The question is, however, if the predictions are sufficiently accurate to yield suitable inverse directional control commands and if these commands can be suitably processed by the adaptive PD controller to reach targets accurately. To test the dynamic control performance of the Dynamic XCSF control system, we used the same test setup as in the kinematics case. Figure 6 shows performance of XCSF in the respective border fraction settings. Compared to the purely kinematics control mechanism, control reliability slightly decreases but in the inner area (borderfraction=0.2) still nearly all goals are reached reliably—even after the application of the condensation mechanism. Also path efficiency improves over time and reaches a value of about 0.9, indicating that the hand path to the goal locations is on average 11% longer than the direct path of the hand to the goal. Considering the necessarily elaborate arm control mechanisms, this result indicates reliable and effective control of a dynamic arm plant, which has never been shown to be possible with any XCS(F) system before.

While effective dynamic control is confirmed by the above experiments, we still had to evaluate the constraint resolution mechanism. Thus, we conducted experiments similar to the kinematic arm setting imposing constraints of posture (0, 0, 0) and joint angle 0 on the redundancy resolution mechanism. Figure 7 shows that the redundancy resolution mechanism equally applies in the dynamic XCSF control setting. While the zero movement constraint (No Const) yields final postures furthest away from posture (0, 0, 0), the joint angle constraint yields closer final postures and the posture goal posture (0, 0, 0) and joint angle 0 on the redundancy resolution mechanism.
constraint yields the closest (Const 000). Meanwhile, the mean absolute differences in the shoulder joint angle to angle 0 are smallest when applying the joint angle constraint, nearly equally small when applying the posture constraint, and much farther on average when applying the zero movement constraint. Thus, redundancy resolution works also in the dynamics setting. Neither path efficiencies nor reaching reliabilities are affected compared to the performance in the zero constraint setting.

4.4 Kuka KR16 Arm Control

As the final test scenario, we applied the dynamic XCSF-based control mechanism on a simulation of the commercially available Kuka KR16 robot [9]. We learn to control three joints of the arm that all work on a two dimensional vertical plane. To train the Kuka arm, we generated a learning scenario similar to the one used in the self-generated dynamic arm setting. Random posture goals were generated and the adaptive PD controller was used to guide the Kuka arm to the goal. We used the commercially available Webots [12] simulation package for the Kuka simulation (cf. Figure 8a). Again, we filtered the data to assure sufficiently of the imposed constraints.

Figure 7: XCSF reaches postures closer to the constraint posture or joint angle confirming the influence of the imposed constraints.

Figure 8a). Again, we filtered the data to assure sufficiently.

The highest potential of XCSF, however, may lie in its sensorimotor structures and its applicability to sensory data that is not directly computable or transferable by the knowledge of the system’s inverse kinematics. For example, pneumatic robot systems, in which muscle power is defined by air pressure, pose a much more human-like dynamic control challenge, which is not solvable with traditional inverse kinematic-based control approaches. The potential of XCSF in such more cognitive control tasks awaits future research efforts as does the application of the system to full three dimensional control tasks. Nonetheless, the available results as well as the knowledge that XCSF is applicable in problems with at least seven input dimensions [4] suggest that XCSF learning is scalable to three dimensional control tasks with a, for example, seven degree or freedom human-like arm system.

Acknowledgments

The authors acknowledge funding from the Emmy Noether program of the German research foundation (grant BU1335/3-1) and like to thank their colleagues at the department of psychology and the COBOSLAB team.

5. CONCLUSIONS

The system evaluations of the XCSF control system have shown that the encoded redundancies may actually not pose a problem during redundancy resolution but allow the flexible imposition of additional task constraints, such as posture or joint angle preferences. Moreover, it is easy to impose a zero movement constraint, stressing the execution of posture commands that yield minimal posture movements. It seems imaginable that such constraints may even be further adapted in order to yield even more efficient path to hand goal locations, improving, for example, the energy efficiency of the system by imposing a least effort constraint. Moreover, the coupling with the adaptive PD controller, which self-adapts dependent on velocity and gravity knowledge, showed that control applications to commercial robotic systems are within our reach, although further optimizations seems to be necessary to ensure 100% performance reliability.

The system evaluations of the XCSF control system have shown that the encoded redundancies may actually not pose a problem during redundancy resolution but allow the flexible imposition of additional task constraints, such as posture or joint angle preferences. Moreover, it is easy to impose a zero movement constraint, stressing the execution of posture commands that yield minimal posture movements. It seems imaginable that such constraints may even be further adapted in order to yield even more efficient path to hand goal locations, improving, for example, the energy efficiency of the system by imposing a least effort constraint. Moreover, the coupling with the adaptive PD controller, which self-adapts dependent on velocity and gravity knowledge, showed that control applications to commercial robotic systems are within our reach, although further optimizations seems to be necessary to ensure 100% performance reliability.

6. REFERENCES

Figure 8: XCSF is also able to learn suitable predictions (b) to predict hand position changes given posture movements for the Kuka KR16 (a) robot arm. The predictive structures can be suitably used to effectively control the Kuka KR16 robot arm when coupled with the adaptive PD controller (c).


