# Hyper-ellipsoidal Conditions In XCS: Rotation, Linear Approximation, and Solution Structure

Martin V. Butz
Department of Cognitive Psychology
University of Würzburg
97070 Würzburg, Germany
mbutz@psychologie.uni-wuerzburg.de

Pier Luca Lanzi
Dip. di Elettronica e Informazione
Politecnico di Milano
Milano 20133, Italy
pierluca.lanzi@polimi.it

Stewart W. Wilson
Prediction Dynamics
Concord, MA 01742, USA
wilson@prediction-dynamics.com

## ABSTRACT

The learning classifier system XCS is an iterative rule-learning system that evolves rule structures based on gradient-based prediction and rule quality estimates. Besides classification and reinforcement learning tasks, XCS was applied as an effective function approximator. Hereby, XCS learns space partitions to enable a maximally accurate and general function approximation. Recently, the function approximation approach was improved by replacing (1) hyperrectangular conditions with hyper-ellipsoids and (2) iterative linear approximation with the recursive least squares method. This paper combines the two approaches assessing the usefulness of each. The evolutionary process is further improved by changing the mutation operator implementing an angular mutation that rotates ellipsoidal structures explicitly. Both enhancements improve XCS performance in various non-linear functions. We also analyze the evolving ellipsoidal structures confirming that XCS stretches and rotates the evolving ellipsoids according to the shape of the underlying function. The results confirm that improvements in both the evolutionary approach and the gradient approach can result in significantly better performance.

## Categories and Subject Descriptors

I.2.8 [**Artificial Intelligence**]: Problem Solving, Control Methods, and Search; I.2.6 [**Artificial Intelligence**]: Learning; G.1.2 [**Numerical Analysis**]: Approximation

## General Terms

Algorithms

## Keywords

Function Approximation, Genetic Algorithms, LCS, XCS, Locally Weighted Learning, Recursive Least Squares


*GECCO'06,* July 8–12, 2006, Seattle, Washington, USA.


## 1. INTRODUCTION

The XCS learning classifier system was introduced by Wilson in 1995 [16]. Recent applications of the system to real-world datamining problems have confirmed the strong learning capabilities of the system [2, 3]. Moreover, theoretical advancements have led to a deeper understanding as well as to a theoretical confirmation that XCS is able to reliably evolve accurate problem solutions for a wide range of problems in polynomial time [6].

Recent system enhancements have targeted on improving the evolving solution structure (that is, the population of classifiers) on two ends of the spectrum: (1) Condition structures were modified to be maximally suitably applicable in real-valued and multi-valued problems [17, 2]. (2) Predictions were enhanced to linear and polynomial predictions [18, 14]. Most recently, it was shown that general ellipsoidal condition structures are more effective than hyperrectangular structures for many function approximation problems [5]. Moreover, the delta rule for the learning of (linear) rule predictions was effectively replaced with the pseudo-inverse method or recursive least squares (RLS) method [14, 13].

This paper combines the two approaches investigating if the RLS method is also useful in combination with the hyperellipsoidal condition structures. The results confirm the usefulness of the improved approximation method also within the representation of hyperellipsoidal conditions. Moreover, the paper improves the evolutionary algorithm for the evolution of rotated hyperellipsoidal conditions by mutating the angular orientation of the evolving hyperellipsoids explicitly instead of mutating the complete covariance matrix. Also this approach, independent of the approximation method taken, improves XCS's function approximation performance. Finally, this paper investigates the evolving ellipsoidal condition structures confirming that XCS stretches and rotates the ellipsoids most suitably for a maximally accurate function approximation.

The paper is structured as follows. The next section gives a short overview of the XCS system. Next, we progressively add the explicit rotation and the RLS structure to the hyperellipsoidal condition representation and evaluate the consequent function approximation performance. Section 5 analyzes the evolved condition structures. Summary and conclusions discuss the impact of this study from a broader perspective.

## 2. XCS BACKGROUND

XCS is a typical Michigan-style learning classifier systems (LCSs) [12, 4]. The following introduction of XCS introduces the enhanced XCS system for function approximation — often termed XCSF [17, 18]. Moreover, we introduce XCSF with general hyperellipsoidal conditions [5]. For further information on XCS the interested reader is referred to the cited literature as well as the algorithmic description of XCS [8].

### 2.1 Representation

XCSF is applied to real-valued function approximation problems. The system learns iteratively receiving an input vector $\vec{x_t} = (x_1, ..., x_n) \in \mathcal{S} \subseteq \Re^n$, determining its function value prediction $P$ and receiving the actual function value $y_t$. XCSF represent its function approximation values in a population of classifiers, where each classifier specifies in its condition part its applicability and in its prediction part its linear function approximation. Essentially, XCSF realizes function approximation by a locally weighted learning approach [1]. Classifier conditions partition the input space and classifier predictions (linearly) approximate the function value within the partitions.

A classifier in XCSF consists of a condition $C$, a prediction $R$, a prediction error $\varepsilon$, and a fitness value $F$. The general hyperellipsoidal condition $C$ is formally represented as follows:

$$C = (\vec{m}, \Sigma) = ((m_1, m_2, ..., m_n)^T, (\sigma_{1,1}, \sigma_{1,2}..., \sigma_{n,n-1}, \sigma_{n,n})), \quad (1)$$

where $\vec{m}$ specifies the center of the ellipsoid, the notation $^T$ denotes the transpose of a (column) vector or matrix, and $\Sigma$ defines the fully weighted Euclidean distance metric of the ellipsoid, also termed Mahalanobis distance [1]. This *transformation matrix* essentially determines the stretch of the ellipsoid and the rotation in the $n$-dimensional problem space. Classifier activity is determined by

$$cl.ac = \exp\left(-\frac{(\vec{x} - \vec{m})^T \Sigma^T \Sigma (\vec{x} - \vec{m})}{2}\right), \quad (2)$$

which essentially determines the distance from the current input and then applies the Gaussian kernel on the distance. A classifier is active (that is, it *matches*), if its current activity is above some threshold $\theta_m$.

The linear function *prediction* $R$ is specified by a weight vector

$$R = \vec{w} = (w_0, w_1, ..., w_n)^T, \quad (3)$$

where $w_0$ is the offset weight. The prediction is then determined by the inner product $(\vec{x^*} - \vec{m^*})^T \vec{w}$ where vectors $\vec{x}$ and $\vec{m}$ are enhanced by a trailing one and zero, respectively. The *prediction error* $\varepsilon$ estimates the mean absolute error of the linear predictions. The *fitness* $F$ specifies the accuracy of the classifier relative to competing, that is, overlapping classifiers. The three estimates are iteratively updated by the following gradient-based techniques.

### 2.2 Rule Parameter Update

Each learning iteration $t$, XCS receives a problem instance $\vec{x_t}$ with the corresponding actual function value $y_t$. XCS forms a match set $[M]$ of all classifiers whose conditions are active. The match set is where XCS updates its classifier parameters.

XCS uses the error between prediction $||(\vec{x_t^*} - \vec{m^*})^T \vec{w}||$ and actual function value $y_t$ to update the reward prediction, prediction error, and fitness estimate of all classifiers in match set $[M]$ (Since we do function approximation, we ignore actions so that there are no action sets). Hereby, we use the usual delta update rule for the weight vector $R$, that is,

$$w_i \leftarrow w_i + \eta(y_t - (\vec{x^*} - \vec{m^*})^T \vec{w})(x_i^* - m_i^*), \quad (4)$$

where $\eta$ denotes the learning rate. Reward prediction error and fitness are updated by the usual XCS procedure using the inner product $||(\vec{x_t^*} - \vec{m^*})\vec{w}^T||$ as the prediction. Essentially, the reward prediction error approximates the averaged absolute error of the generated prediction minus the received actual function value. Fitness is derived from the error and estimates the averaged relative accuracy of a classifier in comparison with other, overlapping, competing classifiers. Fitness is used for offspring selection. After rule updates and possible GA invocation in the current match set, the next iteration starts.

### 2.3 Rule Structure Evolution

XCSF is initialized with an empty population. Initial classifiers are generated by a covering mechanism that creates a matching condition given a problem instance $\vec{x}$ for which no classifier matches. The center of the hyperellipsoid ($\vec{m}$) is set to the current input ($\vec{x}$). Only the diagonal entries in the transformation matrix $\Sigma$ are initialized to the squared inverse of the uniformly randomly chosen number between zero and the parameter $r_0$. All other matrix entries are set to zero. In this way, covering creates axis-parallel hyperellipsoidal condition parts.

XCS applies a steady-state, niched GA for rule evolution. The GA selects two parental classifiers from the current match set $[M]$ using set-size relative tournament selection based on the classifier's fitness estimates [7]. Two offspring are generated from the selected parents and crossover and mutation is applied. We utilize uniform crossover, in which any corresponding values in the two classifier conditions are exchanged with a probability of .5. Mutation alters each entry in the condition part with a probability of $\mu$. Mutation moves the center of the hyperellipsoid uniformly randomly within its current interval. A matrix entry is mutated by maximally decreasing (increasing) the value by 50%. If the value was set to zero, it is initialized to a randomly chosen value as in covering for the diagonal matrix entries but considering parameter $\mu_0$. Before the offspring is inserted in the population two classifiers may be deleted to keep a fixed population size $N$. Classifiers are deleted from $[P]$ with probability proportional to an estimate of the size of the match sets that the classifiers occur in. If the classifier is sufficiently experienced and its fitness $F$ is significantly lower than the average fitness of classifiers in $[P]$, its deletion probability is further increased.

### 2.4 Learning Bias in XCS

The evolutionary algorithm is the main component that searches for better problem space partitions with respect to the achieved predictive accuracy. GA selection propagates the currently most accurate classifiers. Mutation searches in the neighborhood for better space partitions. Crossover combines previously successful sub-partitions. Selection in action sets in combination with deletion in the whole pop-

ulation stresses generalization. Moreover, the mechanism has a niching effect biasing the evolutionary process towards evolving a classifier population that covers the whole problem space.

While the evolutionary mechanism is designed to evolve partitions in which linear approximations are maximally accurate, the gradient descent-based methods estimate the suitability of the current partitions. Thus, XCS applies a distributed, local search mechanism combining evolutionary techniques with gradient learning techniques to find a global problem solution. As a whole, XCS strives to evolve a complete, maximally accurate, and maximally general function approximation.

# 3. EFFICIENT SEARCH IN ROTATION SPACE

This section first investigates parameter dependencies of XCSF. Next, the mutation operator is improved resulting in higher parameter independence and more reliable learning performance. We then replace the iterative delta rule update of the predictions with the RLS method. The combination of both improvements yields the strongest performance.

The investigations focus on the following function:

$$f_1(x, y, z) = sin(8\pi(x + y + z)) \quad (x, y, z \in [0, 1]).$$

Note that the sine lies obliquely in the three dimensional problem space. Twelve full sinus waves need to be approximated in the three dimensional space. If not stated differently, all experiments are averages over 20 experimental runs and parameters were set as follows: $N = 6400$, $\beta = \eta = 0.5$, $\alpha = 1$, $\varepsilon_0 = .01$, $\nu = 5$, $\theta_{GA} = 50$, $\tau = .4$, $\chi = 1.0$, $\mu = 0.05$, $r_0 = .5$, $\theta_{del} = 20$, $\delta = 0.1$, $\theta_{sub} = 20$.

## 3.1 Parameter Dependencies

We first investigate the impact of learning rate $\beta$ and mutation control parameter $\mu_0$ on performance. Figure 1 shows that the function is highly challenging for XCSF with hyperellipsoids in the chosen settings. Best performance is achieved when the learning rate $\beta$ is set to the lower value .1 and value $\mu_0$ is set to the lower value .5. The learning rate influence indicates that XCSF relies on stable parameter estimates. The high rate $\beta = .5$ appears to introduce too much noise in the error and fitness estimates making it hard for XCS to detect more suitable classifier structures. The mutation influence $\mu_0$ suggests that small changes in classifier structure are advantageous in the solution structure evolution. Thus, small iterative improvements are better than larger jumps in the condition structure space.

Both observations are confirmed in the successive sections improving first condition structure search and then linear approximation.

## 3.2 Adding Explicit Rotations via Mutations

In its current form, the full transformation matrix determines the stretch and orientation of ellipsoidal conditions. This is clearly a highly redundant encoding of the actual necessary structure. Although redundant encodings are known to be potentially helpful in the GA literature [9], redundancy in the case of the ellipsoidal encoding may not be as helpful due to several reasons: (1) Mutations in the transformation matrix may cause strong alterations in the underlying ellipsoidal structures. (2) The redundant encoding may

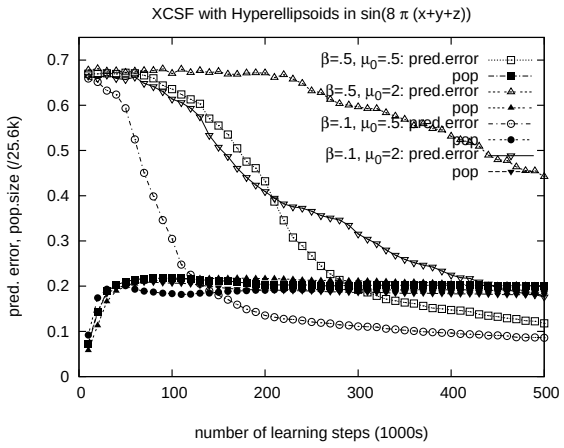


**Figure 1: The $sin(8\pi(x + y + z))$ function is a hard approximation task.**

lead to disruptive crossover operations. Essentially, a redundant encoding of two similarly oriented ellipsoidal structures may actually lead to a completely differently oriented ellipsoid. Thus, recombined offspring may often reflect random guesses rather than effective structural recombination. (3) The redundant representation may lead to unnecessary diversity and consequent evolutionary stagnation in XCS. Since successful evolutionary search relies on reproductive opportunities of classifiers before likely deletion [7], higher diversity can prevent the detection of relevant classifiers before deletion significantly delaying evolutionary progress.

To conquer these constraints, we now replace the covariance matrix representation with an angular representation. It is know, that in three (two) dimensions, three (one) angles are sufficient for an exact representation of any orientation of an object. In our case, we represent hyperellipsoids. Given the Euler angles $\gamma_i$, we can easily compute the corresponding transformation matrix [15]. Thus, instead of evolving the full transformation matrix explicitly, we now evolve the Euler angles $\gamma_i$ and use the resulting matrix to determine classifier activity as described above.

In the evolutionary process, the Euler angles are initialized to zero upon covering. Mutation alters the angles by a uniform random number in the range $[-\pi\mu_0, \pi\mu_0]$. The angles are constrained to lie in the range of $(-2\pi, 2\pi]$. Note that this representation still allows redundant encodings: For example, in two dimensions an ellipse with stretch $m_1 = 2$ and $m_2 = .1$ and angle 0 (or $2\pi$) is equivalent to another ellipse with stretch $m_1 = .1$ and $m_2 = 2$ and angle $\pi/2$ (or $-3\pi/2$).

Figure 2 shows that the angular mutation improves XCSF's performance significantly. In fact, with a lower learning rate, regardless of the chosen $\mu_0$ value, XCSF evolves an accurate function representation. With a larger learning rate, the fluctuations in parameter estimates persist and continue to delay learning progress. Nonetheless, also in this setting XCS evolves a similar accurate solution after $500k$ learning iterations. Compared to the previous mutation approach, XCSF reaches a higher accuracy level in all four settings. Also learning speed is improved. However, learning rate influences stay significant. Thus, we now add an improved linear prediction method to the system, which was introduced elsewhere [14, 13].

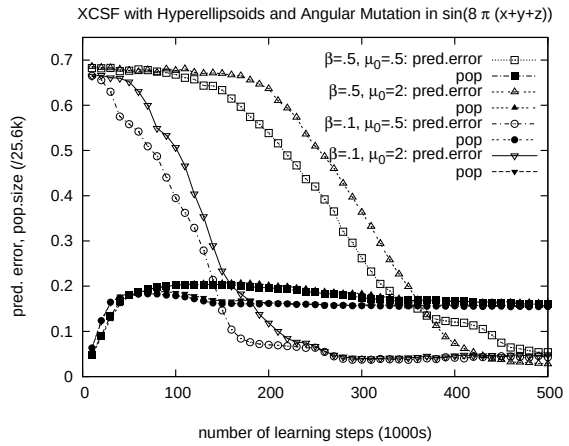


**Figure 2: The benefit of rotation becomes very obvious in the $sin(8\pi(x+y+z))$ function. Still, a good approximation takes time to evolve and stays parameter dependent.**

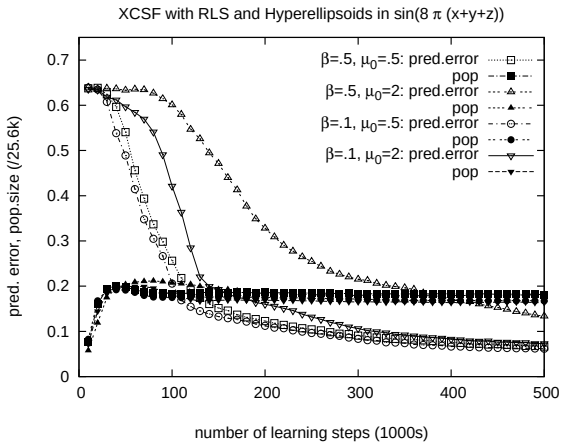


**Figure 3: With the RLS approximation, the $sin(8\pi(x+y+z))$ performance somewhat stabilizes yielding a more smooth performance progress. However, large mutation rates as well as large learning rates continue to cause performance disruption.**

## 3.3 Using Recursive Least Squares Approximation

The results so far showed that XCS suffers from a larger learning rate $\beta$ indicating that the error and fitness estimates, which depend on the generated function value predictions, fluctuate and consequently partially mislead the evolutionary progress. Recently, the delta rule update of the prediction part (Equation 4) was replaced by the pseudoinverse method [14] and RLS [13]. RLS is known to yield faster and very stable parameter approximations [11].

To implement (linear) RLS in XCS classifiers, a matrix $V$ (of size $|n| \times |n|$) needs to be added to each classifier. The update of XCSF with RLS is done as follows. Given the current input $x$ and the target value $y$, RLS updates the weight vector $\vec{w}$ as

$$\vec{w} \leftarrow \vec{w} + \vec{k}[(y_t - (\vec{x^*} - \vec{m^*}))^T \vec{w}],$$

where, $\vec{k}$ is the *gain vector* computed as

$$\vec{k} = \frac{V^T(\vec{x^*} - \vec{m^*})}{1 + (\vec{x^*} - \vec{m^*})^T V^T (\vec{x^*} - \vec{m^*})}, \quad (5)$$

while matrix $V$ is updated recursively by,

$$V^T = \left[I - \vec{k}(\vec{x^*} - \vec{m^*})^T\right] V^T. \quad (6)$$

RLS implements least squares rigorously. For the initialization (avoiding the often used inverse computation), we set matrix V to the scaled identity matrix as suggested elsewhere [11]:

$$V = \delta_{rls} \times I, \quad (7)$$

where $I$ is the identity matrix of dimension $n+1$ and $\delta_{rls} > 0$ ($\delta_{rls} = 1$ in the experiments herein). Note that RLS is a special case of Kalman filtering for the case of a fixed target state and no control signal applied to the state variables (that is, the weight vector $\vec{w}$).

Performance of XCSF with RLS is reported in Figure 3. Clearly, RLS stabilizes parameter estimates making the system more independent from the chosen learning rate $\beta$. In fact, with a sufficiently low $\mu_0$ value, learning stays nearly independent from learning rate $\beta$. However, when $\mu_0$ is chosen too high, also the higher learning rate continues to cause disruption. This indicates that RLS is able to yield better and faster approximations but cannot directly help in the structural search for maximally effective space partitions.

## 3.4 Angle Representation Plus RLS

Combining RLS with the introduced angular mutation yields most effective performance. Figure 4 shows that now all four parameter settings converge quickly and reliably to an accurate function approximation solution. The achieved approximation error is smaller than in all previous settings indicating that (1) the angular mutation enables a more directed evolution to maximally suitable space partitions, and (2) the RLS yields better and faster function value approximations in the evolving partitions. In the next section, we investigate the performance on several other three dimensional functions assessing the general usefulness of RLS and the angular mutations in the evolution of hyperellipsoidal condition structures.

# 4. PERFORMANCE SUITE

Despite of the achieved improvements, it remains unclear when rotations are actually useful. This section investigates performance in several other challenging three dimensional functions evaluating the usefulness of RLS and angular mutations in them.

## 4.1 Functions with Independent Dimensions

It was hypothesized before [5] that rotations of the hyperellipsoidal conditions are only useful when the dimensions are nonlinearly dependent on each other. This is the case in the sinusoidal curve since the sine function is applied to the sum of all dimensions. Thus, the influence of each dimension is combined nonlinearly yielding the function value.

The dimensions are linear independent when the contribution to the target value of each dimension is considered separately and then combined linearly. In this case, space partitioning can be axis-parallel so that also the most suit-

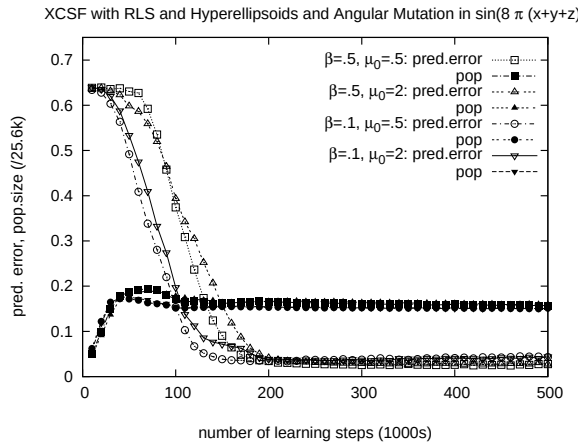


**Figure 4: With the general hyperellipsoidal condition structure and RLS approximation, most accurate approximations are learned in the $sin(8\pi(x+y+z))$ function. Learning also gains speed, stability, and parameter independence.**

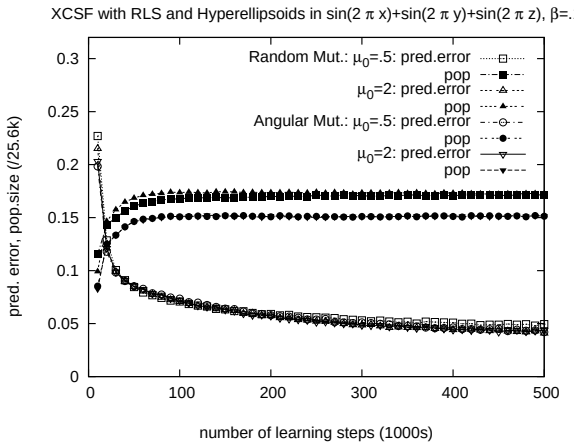


**Figure 5: In a nonlinear three dimensional function with independent problem dimensions, rotation does not improve performance.**

able orientation of the ellipsoids is axis-parallel. Thus, rotations should not be advantageous in functions in which each dimension is combined linearly.

One example of such a function is:

$$f_2(x, y, z) = sin(2\pi x) + sin(2\pi y) + sin(2\pi z).$$

Figure 5 shows performance of XCSF with hyperellipsoids and RLS in this function. As predicted, angular mutation has no influence on the learning performance. Also the learning rate has only a slight influence on performance. Population size is larger in the case of matrix mutations since the condition structure is expressed by more parameters letting additional room for mutations and additional condition representations. The function is generally quite challenging, though, seeing that three sine waves interact in the production of the overall function value.

## 4.2 Superimposed Sine Functions

When we superimpose several sinus functions, the required space partitions reflecting nonlinearity become even more important. We tested XCSF without and with RLS as well as without and with angular mutation on several superimposed sinuses. Due to the complexity of the function, we double the maximum population size to $N = 12800$.

Figure 6 shows XCSF's performance on the following superimposed sine function:

$$f_3 = \quad sin(2\pi(x+y+z)) + sin(4\pi(x+y+z)) + sin(6\pi(x+y+z))$$

We can see that angular mutation helps to evolve an optimal classifier condition structure—regardless if without or with RLS. Surprisingly, performance degrades slightly after about $200k$ learning iterations—most likely an effect of disrupting interactions between overlapping classifier structures. This effect needs to be investigated further.

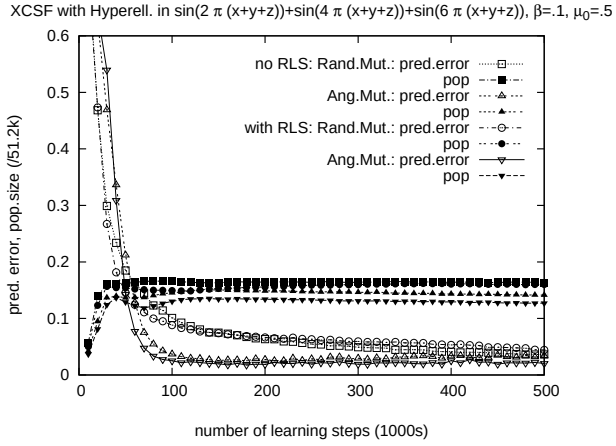


**Figure 6: Angular mutation improves learning speed significantly in $f_3$. RLS has only an additional, but small positive influence on performance.**

## 4.3 Non-continuously Differentiable Function

Another function in which space partitions need to be even more abrupt are non-continuously differentiable functions. At the point of the non-continuity in the differentiation, the linear approximation becomes quickly severely incorrect, so that this point should be partitioned well. XCSF needs to find these partitions efficiently in order to evolve accurate function approximations.

One example of such a function is the following:

$$f_4 = |sin(2\pi(x+y+z))| + |cos(2\pi(x+y+z))|$$

Figure 7 shows the performance of XCSF ($N = 6.4k$) with hyperellipsoids in this function. Angular mutation appears to be very useful in this case, yielding a faster learning performance and a more accurate performance after 500k learning steps in comparison to runs without angular mutation. Again, angular mutation yields higher payoff than RLS although RLS also improves learning speed and also increases accuracy—at least in the case without angular mutation.

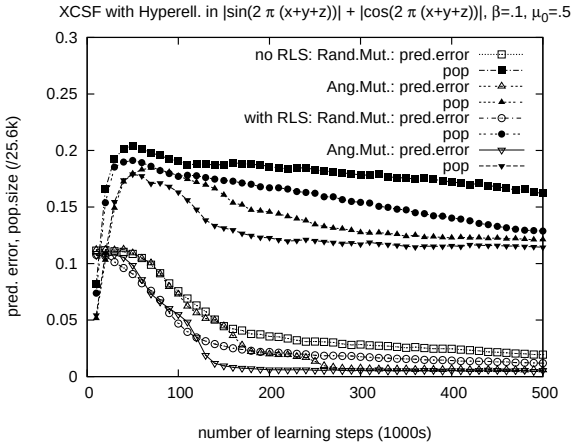


**Figure 7: Also in the case of a non-continuously differentiable, oblique function, angular mutation yields significant performance improvements.**

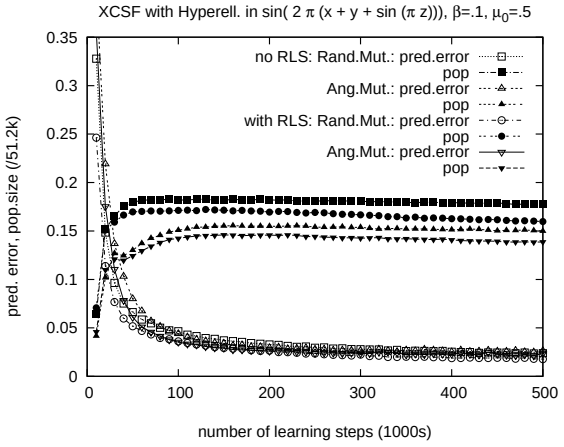


**Figure 8: The RLS update yields slightly stronger improvement than angular mutation in a function in which obliqueness changes over the problem space.**

## 4.4 Changing Obliqueness

As a final test function, we change the obliqueness of the function in question by applying a non-linear function, such as the sine, to one dimension before adding it to the other dimensions and then applying an overall function, such as another sine. Thus, we have two nested sine functions. One function of that kind is:

$$f_5 = sin(2\pi(x + y + sin(\pi z)))$$

In this case, the required most suitable rotations are now twisted inside each other and are changing continuously. Thus, rotations may not be very useful any longer. Figure 8 shows performance of XCSF ($N = 12.8k$) in this function. As suggested, rotations are hardly useful and actually yield a slightly worse performance than runs with general mutation. As suspected, the function appears too complex to evolve efficient rotations so that angular mutation has hardly any effect on performance.

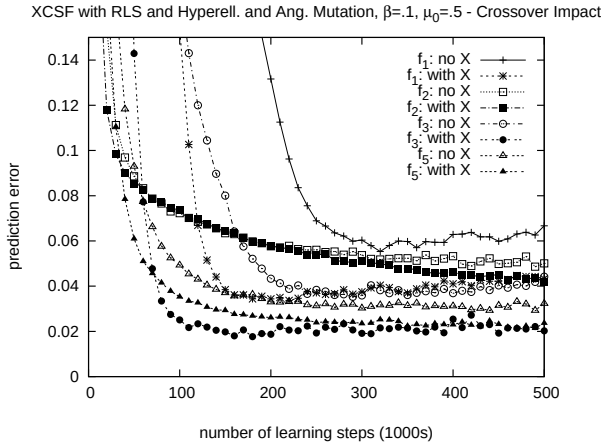


**Figure 9: Uniform crossover yields performance improvement throughout the investigated functions.**

## 4.5 Impact of Crossover

So far, it was not shown if the uniform crossover application has an impact on performance. In the investigated functions, it seems expectable that crossover should be useful to spread suitable, potentially partial ellipsoidal sizes, stretches, and orientations over the search space. Figure 9 shows various runs of XCS without crossover application in the investigated functions. It can be seen that crossover speeds up learning and improves the accuracy reached throughout the investigated functions. Population size (not shown) in the runs without crossover stays slightly lower due to the decrease in mixing. Interestingly, crossover has the least beneficial effect in functions $f_2$—the axis-independent sine function. Required ellipsoidal size continuously change in each dimension so that structural spreading is not beneficial. The possibly beneficial recombination of different ellipsoidal stretches in each dimension shows hardly any beneficial performance impact.

# 5. RESULT STRUCTURES

Since linear approximations yield larger errors in areas in which the second derivative of the target function is large, space partitions should be smaller in these areas. When evolving hyperellipsoids in the oblique 2-D sinusoidal function $f_6 = sin(2\pi(x + y))$, the relative stretch of the ellipse in 2-D space should consequently be stronger, the more curvature in the sine function. This can be observed in Figure 10 in which we plot the ratio between larger and smaller dimensions of the learned two dimensional ellipses (population after 500k learning iterations). Figure 10 is rotated by 45° so that the perspective is perpendicular to the sine function. As can be observed, the stretch (obtained by dividing the smaller deviation by the larger one) actually mimics the absolute value of the second derivative of the sine function (that is, the negative sine), except for in the corners in which undersampling results in noisy boundary effects.

Although the stretches suggest that XCSF evolves suitable ellipsoids, we did not show as yet that the orientation of the evolving ellipsoids also mimics the obliqueness of the function. To investigate this, we tested the final population for the orientation of the evolving ellipsoids. To do so, we

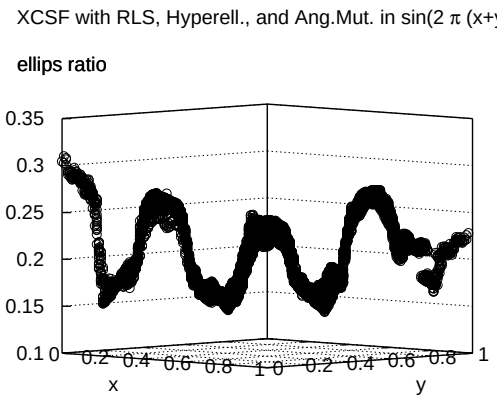


**Figure 10: The stretch factor (ellipse ratio) of the ellipses increases in regions with small second derivatives in the target function.**

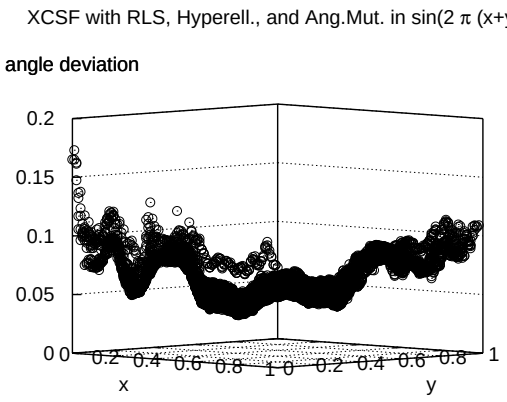


**Figure 11: XCSF approximates the required $\pi/4$ orientation of the ellipses quite accurately. In regions with high second derivatives of the target function, the approximation becomes more precise since larger deviations cause even higher inaccuracies in the function value predictions.**

tested for the angles and investigated to which degree the orientation mimics the required 45 (which corresponds to $\pi/4$) orientation. To do so, the evolving ellipsoids and degrees had to be converted appropriately. Figure 11 shows the average absolute difference from the target angle $\pi/4$. It can be seen that XCSF approximates the orientation of the function quite accurately. Moreover, the deviations reveal that in more complex regions of the function (that is, in those regions with larger absolute second derivatives), the angular deviations are slightly smaller than in regions with smaller second derivatives. Since in the latter case angular inaccuracies have a lesser effect on prediction errors, less fitness pressure towards maximally accurate angles applies, which explains the higher deviation from $\pi/4$ in these regions.

## 6. SUMMARY AND CONCLUSIONS

This study on XCS's capability in function approximation problems has shown that XCSF is able to approximate complex three dimensional functions. XCSF approximates the target function by overlapping, piecewise-linear approximations. Hereby, XCSF evolves condition parts that optimize the angular orientation as well as the stretch of the evolving hyperellipsoids to achieve a maximally accurate piecewise-linear function approximation.

The experimental results showed furthermore that XCSF's performance can be optimized on two sides of the spectrum: (1) condition structures and structure optimization can be optimized to partition the problem space more effectively; (2) linear approximation can be optimized to yield stable, and maximally accurate approximations as fast as possible. The improvements can be combined yielding additive benefits.

Despite these encouraging results, XCSF still needs quite a large population size to approximate the underlying function. In the case of ellipsoidal condition structures, however, it is hard to determine which classifier condition includes other classifier conditions. Thus, subsumption hardly ever applies. Further investigations in fitness sharing and other niching techniques may lead to the evolution of more compact representations without affecting performance. Another method could be the further refinement of post-processing, compaction algorithms proposed elsewhere [19]. A progressive freezing of the variation operators may yield the desired effect. On the other hand, XCSF evolves a rather redundant encoding of the underlying problem structure, which may be useful in some applications as well. Thus, another way of investigating the strength of XCSF's representation may be to test its robustness by e.g. deleting random classifiers.

The study showed that a redundant phenotype encoding is usually not as helpful as often observed in the genetic algorithms literature [9]. On the other hand, we showed that if required orientations change over the problem space, redundancy may be slightly advantageous. This suggest, as observed for GAs, that also XCSF uses available paths to optimal solutions. Redundant encodings may open new paths. In most of the investigated functions, though, direct mutation of ellipsoidal orientation was the most suitable path.

Despite the successful improvement of the genetic search in XCS(F), for real-valued problems it remains an open question how insights from Evolution Strategies (ES) may be implemented in XCS, and other learning classifier systems for that matter. The covariance matrix adaptation (CMA) algorithm may enable a faster adaptation of condition structures in XCS [10]. However, we have seen that XCSF does not evolve one classifier condition structure over the whole search space but actually evolves many, suitably adapted to the complexity of the problem subspace in question. Thus, the covariance matrix of CMA may need to be either evolved for separate classifiers, or classifier clusters. Research in this direction promises to yield interesting new insights on XCS's distributed problem representation.

Finally, XCSF may be modified to process different in-

put values differently. Currently, all input values are used to determine (1) classifier activity as well as (2) the prediction value, given the classifier matches (that is, has sufficient activity). Another option would be to restrict the determination of activity to a subset of input values and the generation of the prediction value to a different (potentially overlapping) subset of input values. In this case, XCS(F) may then utilize "context" information to control classifier activity and "predictive" information to generate currently applicable output predictions. Wilson [20] made a first step in this direction by investigating classifiers with continuous actions and resulting continuous, action-dependent payoff.

## Acknowledgments

This work was supported by the European commission contract no. FP6-511931. Additional support from the Research Board at the University of Missouri is acknowledged.

## 7. REFERENCES

[1] C. G. Atkeson, A. W. Moore, and S. Schaal. Locally weighted learning. *Artificial Intelligence Review*, 11:11–73, 1997.

[2] E. Bernadó, X. Llorà, and J. M. Garrell. XCS and GALE: A comparative study of two learning classifier systems and six other learning algorithms on classification tasks. In P. L. Lanzi, W. Stolzmann, and S. W. Wilson, editors, *Advances in Learning Classifier Systems (LNAI 2321)*, pages 115–132. Springer-Verlag, Berlin Heidelberg, 2002.

[3] 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.

[4] L. B. Booker, D. E. Goldberg, and J. H. Holland. Classifier systems and genetic algorithms. *Artificial Intelligence*, 40:235–282, 1989.

[5] M. V. Butz. Kernel-based, ellipsoidal conditions in the real-valued XCS classifier system. *GECCO 2005: Genetic and Evolutionary Computation Conference: Volume 2*, pages 1835–1842, 2005.

[6] M. V. Butz, D. E. Goldberg, and P. L. Lanzi. *Foundations of Learning Classifier Systems*, chapter Computational Complexity of the XCS Classifier System, pages 91–126. Studies in Fuzziness and Soft Computing. Springer-Verlag, Berlin Heidelberg, 2005.

[7] M. V. Butz, D. E. Goldberg, and K. Tharakunnel. Analysis and improvement of fitness exploitation in XCS: Bounding models, tournament selection, and bilateral accuracy. *Evolutionary Computation*, 11:239–277, 2003.

[8] M. V. Butz and S. W. Wilson. An algorithmic description of XCS. In P. L. Lanzi, W. Stolzmann, and S. W. Wilson, editors, *Advances in learning classifier systems: Third international workshop, IWLCS 2000 (LNAI 1996)*, pages 253–272. Springer-Verlag, Berlin Heidelberg, 2001.

[9] M. Ebner, M. Shackleton, and R. Shipman. How neutral networks influence evolvability. *Complexity*, 7(2):19–33, 2001.

[10] N. Hansen and A. Ostermeier. Completely derandomized self-adaptation in evolution strategies. *Evolutionary Computation*, 9:159–195, 2001.

[11] S. Haykin. *Adaptive filter theory*. Prentice Hall, Upper Saddle River, NJ, 4th edition edition, 2002.

[12] 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.

[13] P. L. Lanzi. Generalization in the XCSF classifier system: Analysis, improvement, and extensions. IlliGAL report 2005012, Illinois Genetic Algorithms Laboratory, University of Illinois at Urbana-Champaign, 2005.

[14] P. L. Lanzi, D. Loiacono, S. W. Wilson, and D. E. Goldberg. Extending XCSF beyond linear approximation. *GECCO 2005: Genetic and Evolutionary Computation Conference: Volume 2*, pages 1827–1834, 2005.

[15] E. W. Euler angles. From MathWorld–A Wolfram Web Resource. http://mathworld.wolfram.com/EulerAngles.html, 1999.

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

[17] 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.

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

[19] S. W. Wilson. Compact rulesets from XCSI. In P. L. Lanzi, W. Stolzmann, and S. W. Wilson, editors, *Advances in learning classifier systems: Fourth international workshop, IWLCS 2001 (LNAI 2321)*, pages 196–208. Springer-Verlag, Berlin Heidelberg, 2002.

[20] S. W. Wilson. Classifier systems for continuous payoff environments. *Proceedings of the Sixth Genetic and Evolutionary Computation Conference (GECCO-2004): Part II*, pages 824–835, 2004.