Self-Adaptive Mutation in XCSF

Martin V. Butz
Department of Psychology
Röntgenring 11
97070 Würzburg, Germany
butz@psychologie.uni-wuerzburg.de

Patrick Stalph
Dep. of Computer Science
Am Hubland
97074 Würzburg, Germany
patrick.stalph@stud-mail.uni-wuerzburg.de

Pier Luca Lanzi
Artificial Intelligence and Robotics Laboratory
Politecnico di Milano, I-20133,
Milano, Italy
pierluca.lanzi@polimi.it

ABSTRACT

Recent advances in XCS technology have shown that self-adaptive mutation can be highly useful to speed-up the evolutionary progress in XCS. Moreover, recent publications have shown that XCS can also be successfully applied to challenging real-valued domains including datamining, function approximation, and clustering. In this paper, we combine these two advances and investigate self-adaptive mutation in the XCS system for function approximation with hyperellipsoidal condition structures, referred to as XCSF in this paper. It has been shown that XCSF solves function approximation problems with an accuracy, noise robustness, and generalization capability comparable to other statistical machine learning techniques and that XCSF outperforms simple clustering techniques to which linear approximations are added. This paper shows that the right type of self-adaptive mutation can further improve XCSF's performance solving problems more parameter independent and more reliably. We analyze various types of self-adaptive mutation and show that XCSF with self-adaptive mutation ranges, differentiated for the separate classifier condition values, yields most robust performance results. Future work may further investigate the properties of the self-adaptive values and may integrate advanced self-adaptation techniques.

Categories and Subject Descriptors

F.1.1 [**Models of Computation**]: Genetics Based Machine Learning, Learning Classifier Systems

General Terms

Algorithms, Performance, Generalization.

Keywords

LCS, XCS, Mutation, Self-Adaptation

GECCO'08, July 12–16, 2008, Atlanta, Georgia, USA.

1. INTRODUCTION

Michigan-style learning classifier systems (LCSs) [2, 11] are distributed learning systems that evolve a set of rules, the so-called population of classifiers. Classifiers usually consist of conditions, which cluster the problem input space into problem niches, actions, which propose an appropriate action or classification in the specified niche, and predictions, which reflect the appropriateness of the action or classification in the niche. Generally, LCSs can be characterized as distributed problem solvers that use genetic algorithms for rule structure evolution and local approximation techniques for rule evaluation.

The classifier system XCS [15] was shown to approximate target functions maximally accurate—be it Q-value functions, discrete constant functions, or real-valued functions. In either case, XCS evolves rules that partition the problem space in such a way that maximally accurate, maximally general function approximation is possible, that is, maximally accurate approximations with the least number of maximally general classifiers [7]. XCS has been shown to also be modifiable to approximate real-valued functions [16, 17], then called XCSF. In this case, the classifier structure consists of a condition and a prediction only (although also the encoding of an action part is possible). As before, the condition specifies the input subspace in which the classifier is applicable and the prediction approximates the function in this subspace with a linear approximation given the input values. Throughout this paper, we use hyperellipsoidal condition structures [8].

Besides successfully approximating various real-valued functions, recent research has shown that the evolutionary progress in XCS can be improved by adding self-adaptive mutation to the system [4, 12]. In these cases, each classifier in the addressed LCS (XCS or ZCS) maintained its own mutation rate that self-adapted upon offspring generation by a random value uniformly chosen within a pre-specified range. Results showed that learning became more independent of initial parameter settings and was advantageous in dynamic environments and environments where long reward-chains had to be learned. Most recently, it was also shown that self-adaptive mutation can be useful for a neural learning classifier system [13] as well as for anticipatory LCSs, for which the mutation rate was adapted for condition, action, and anticipation parts separately [3].

In this paper, we combine these two advances and show that XCSF's learning reliability can be further enhanced by adding self-adaptive mutation mechanisms. We show that the simple addition of a self-adaptive mutation rate alone in-

creases the system's independence on initial mutation rate settings, but it does not increase accuracy compared to the best fixed mutation rate runs. More fruitful performance improvements are gained if the range in which mutation changes classifier values is self-adapted. Since these ranges again are dependent on the part of the classifier condition they modify, we show that the self-adaptation of mutation ranges of center, stretch, and angle of a classifier condition yields the most accurate learning results in the problems investigated. Moreover, we show that self-adaptive mutation ranges enhance learning accuracy and reliability not only when the initial mutation values were set unsuitably but also when compared with the best mutation value settings.

For further information on XCS and XCSF, the reader is referred to the cited literature [15, 17, 7]. The XCSF implementation and parameter settings used herein are taken from [8] [1]. We now first evaluate simple self-adaptive mutation in XCSF. Next, we introduce parameter-relative self-adaptive mutation ranges and evaluate their performance verifying superior performance. A final discussion concludes the paper.

2. SIMPLE SELF-ADAPTIVE MUTATION

A first approach to self-adaptive mutation in XCSF is simply to adapt the mutation rate μ in XCSF. The mutation rate specifies the probability of mutating a real-valued entry in a classifier condition in XCSF. However, it does not specify how strong the mutation will be, that is, within which range the mutated value may lie.

The range of the mutation of a value was previously specified as follows. A classifier's hyperellipsoidal condition specifies center, stretch, and rotation values [8]. For the center values, each center was mutated in such a way that the resulting center point still lies within the specified hyperellipsoid for each dimension. For the stretch values, each stretch value was maximally increased or decreased by 50% of its previous value. Finally, the rotation values were mutated by a uniform random number between $-.5\pi$ and $.5\pi$.

Simple self-adaptation now modifies the individual mutation rate μ_{cl} upon offspring generation of individual classifiers. That is, each classifier cl now maintains its own mutation rate μ_{cl}. A classifier generated during covering sets this values to the initial mutation rate μ. Upon offspring generation, the mutation rate $\mu_{cl'}$ of the offspring is randomly changed by either increasing or decreasing the value by a certain factor (step-based adaptation), that is

$$\mu_{cl'} = \begin{cases} \mu_{cl}(1+x) & \texttt{with 50\% probability} \\ \mu_{cl}/(1-x) & \texttt{otherwise} \end{cases}, \quad (1)$$

where x is a specified fixed value. Alternatively, a Gaussian distribution is used for the adaptation process (Gaussian-based adaptation), in which case the value is mutated as follows,

$$\mu_{cl'} = \mu_{cl}(1+x), \quad (2)$$

where x is chosen from a Gaussian distribution with a specified standard deviation σ_x. Finally, the condition values of the offspring are mutated dependent on $\mu_{cl'}$. The mutation rate values of individual classifiers were additionally constraint to lie within 0 and 2, that is, if self-adaptation increases to a value above 2 or decreases to a value below zero, it is set to 2 or to the absolute value, respectively[2]. The application of crossover does not affect the individual mutation rates.

Performance of self-adaptive mutation (SAM) was tested in in the oblique sine function problem (cf. [8]):

$$f_1(x_1, ..., x_n) = a \sin(2b\pi \sum_i x_i), \quad (3)$$

where the function modification parameters were set to $a = 1$ and $b = 2$ throughout the experiments. Figure 1(a,b) shows that when the initial mutation rate is set to a small value ($\mu_{ini} = .025$) performance of XCSF with SAM does not beat performance without it. Rather, no SAM is advantageous, since, as it appears, SAM is not necessary in this problem so that the additional search space due to the self-adaptation decreases learning speed as well as final accuracy (although the targeted accuracy of $\epsilon_0 = .01$ is met in all cases). If the initial mutation rate is set higher ($\mu_{ini} = .1$), SAM shows to perform equally well dependent on the type and strength of the self-adaptation. The comparison in Figure 1 (c,d) shows that Gaussian-based self-adaptation with a standard deviation of $\sigma = .5$ yields best performance. Finally, a high mutation rate of $\mu_{ini} = .5$ prevents XCSF without SAM to reach an error level below $\epsilon_0 = .01$. SAM fixes this problem by suitably adjusting the mutation rate.

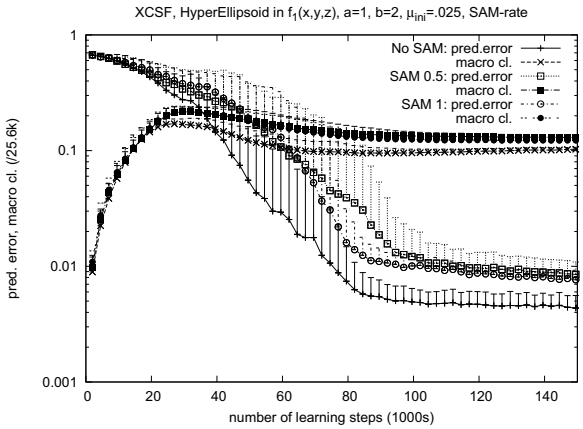
Figure 2 shows how the mutation rate adapts (averaged over all classifiers in the population) with different mutation initialization settings and different self-adaptation methods. It can be seen that there is a general initial trend to increase the mutation rate. This trend is stronger when step-based adaptation is applied, which is also due to the fact that Equation 1 yields higher mutation values on average. Ultimately, though, independent of the initial mutation rate setting and the adaptation method applied, the self-adaptation rates converge to small values. Due to the setting of $\epsilon_0 = .01$ the mutation rate adaptation pressure seizes once the error threshold is reached (cf. performance curves in Figure 1).

In sum, the results so-far show that SAM can alleviate the problem of inappropriate initial parameter settings. This has also been shown for SAM experiments with XCS [12]. Moreover, the analysis of the evolving mutation rates shows that first the rates increase—consequently enhancing the breadth of the search—and then decrease and converge to a low level—consequently ensuring sufficient convergence. These observations strongly relate to research in evolution strategies and SAM approaches, such as the 1/5-rule [1].

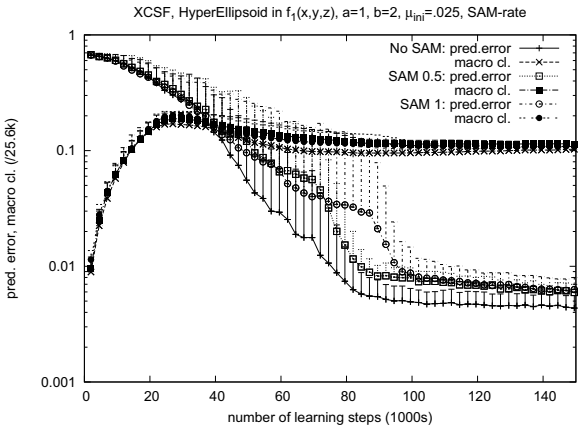
Despite these promising results that show that SAM makes XCSF independent from unsuitable initial mutation rates, there is a concern in problems in which the different condition parameters, and, in particular, center, stretch, and rotation values, need to be adapted with different speeds and strengths. In this case, a self-adaptive mutation rate would not have a direct impact since all three values are affected similarly. Moreover, the self-adaptive mutation rate only

[1] All results reported show averages and estimated standard deviations from 20 independent runs, for which problem instances were sampled uniformly randomly. The parameter settings (if not stated differently) were $N = 6400$,$\beta = .1$, $\alpha = 1$, $\varepsilon_0 = .01$, $\nu = 5$, $\theta_{GA} = 50$, $\chi = 1.0$, $\mu = .025$, $r_0 = 1$, $\theta_{del} = 20$, $\delta = .1$, $\theta_{sub} = 20$.

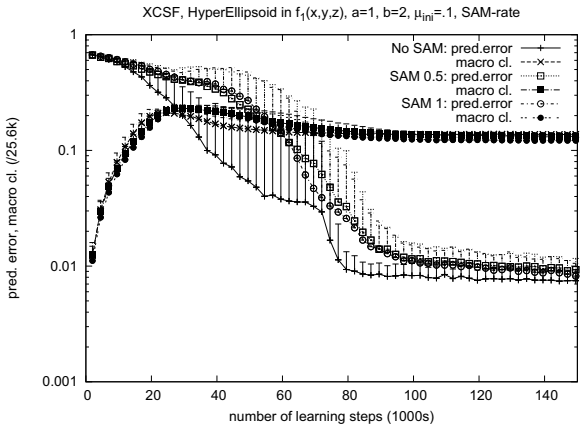
[2] This method results in a slight bias towards larger mutation rates. However, setting a value to zero would result in no mutation and the self-adaptive approach used prevents the recovery from a zero entry. Any value above one means that mutation is applied to all condition parameters.



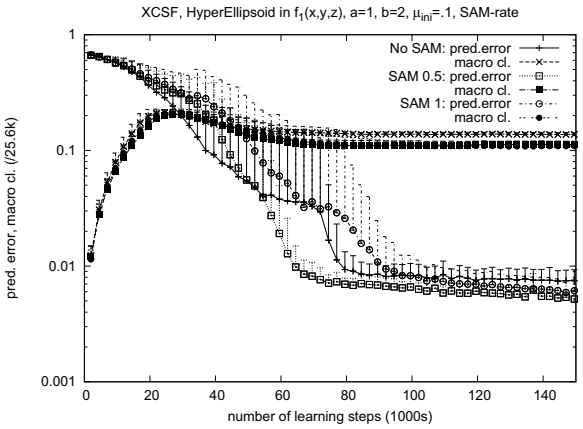
(a) $\mu_{ini} = .025$, Step-based SAM-rate



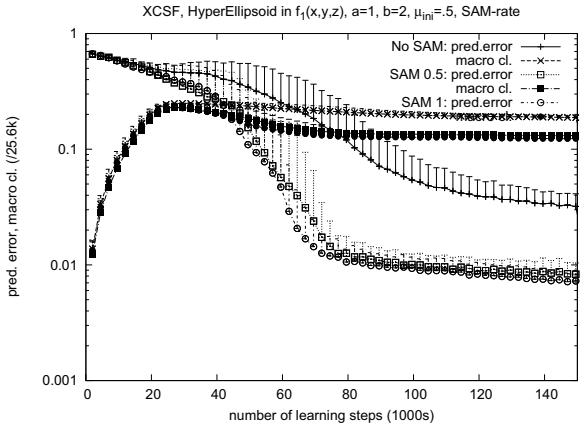
(b) $\mu_{ini} = .025$, Gaussian-based SAM-rate



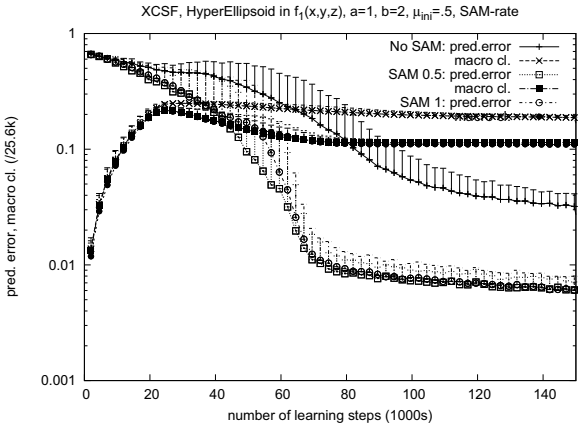
(c) $\mu_{ini} = .1$, Step-based SAM-rate



(d) $\mu_{ini} = .1$, Gaussian-based SAM-rate



(e) $\mu_{ini} = .5$, Step-based SAM-rate



(f) $\mu_{ini} = .5$, Gaussian-based SAM-rate

Figure 1: In the oblique sine function, XCSF with simple self-adaptive mutation (SAM) benefits only when the initial mutation rates are set very high (e,f). In an intermediate setting of $\mu_{ini} = .1$, Gaussian-based adaptation appears to be able to pick up the signal slightly faster (c,d) and it yields more accurate performance than without self-adaptation (d,f) and in comparison to step-based self-adaptation (a,b; c,d; e,f).

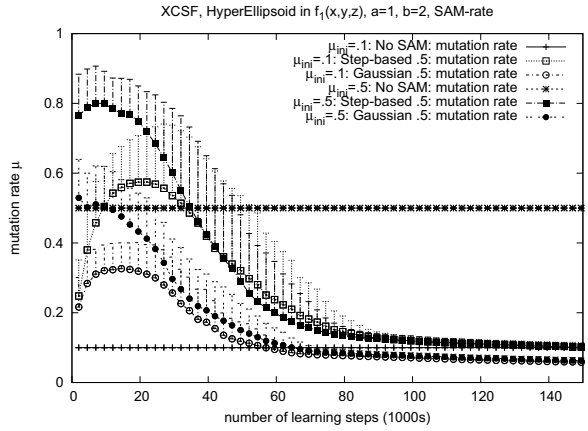


Figure 2: During self-adaptation, the mutation rate first increases and then converges to a suitable, low level.

adapts the probability if a value is mutated but not to what extend it is mutated.

These concerns led us to test XCSF on a radial-sine function that requires the evolution of different stretches and angles dependent on the problem subspace identified by the center of the condition:

$$f_2(x_1, ..., x_n) = ae^{-8\sum_i (x_i - .5)^2} cos(2b\pi \sum_i (x_i - .5)^2) \quad (4)$$

where a and b are function constants that allow further modifications. Here, we set $a = 1$ and $b = 6$ and the population size $N = 12800$ throughout the experiments. As in the sine function, the defined problem input space is in $[0, 1]^n$. Figure 3 shows that both, XCSF without SAM and XCSF with SAM-rate solve the problem reaching an error level of .05 or worse. Again, we see a strong influence of initial mutation rate, yielding best performance with a mutation rate of .1. With SAM-rate, the error does not even drop below .07. Learning does not decrease the function approximation error but stalls on a constant, high (macro classifier) population size level.

To improve performance on this problem, we now enhance SAM individuating self-adaptation for the different types of condition values. Moreover, we self-adapt the mutation *ranges* rather than the mutation *rates*. We show that this methodology does achieve both, it alleviates inappropriate parameter settings and yields better solutions to the harder radial-sine function, in which conditional variations are dependent on the problem subspace the condition applies in.

3. SELF-ADAPTIVE MUTATION RANGES

While the last section has confirmed the applicability of SAM in the real-valued domain, so far, performance only outperformed the runs with fixed mutation rates when the rates were set inappropriately. Moreover, one mutation rate was evolved for rather different condition parameters of the specified hyperellipsoids, that is, condition center, stretch, and orientation. Thus, we now enhance the self-adaptation capabilities and adapt mutation *ranges*, that is, the range within which a value may change when mutation is applied.

Moreover, since center, stretch and orientation are rather different values with different value ranges and effects on the hyperellipsoidal structure, we maintain separate mutation range parameters for center, stretch, and angles. These values are set to one initially—effectively yielding the exact same mutation effects that result in the setting without SAM.

In principle, SAM-ranges are adapted as the SAM-rate adaptation described above. We distinguish between step-based and Gaussian-based SAM-ranges. When a new classifier is generated via covering, each mutation range parameter r^i_{cl} of a classifier cl is set to a specified value r_0. When a classifier has offspring, the offspring ranges are modified according to equations 1 or 2. Then, the condition values are mutated.

Each condition value is mutated with a probability μ (or μ_{cl}, given adaptive mutation rate is applied as well). If a value is mutated, then it is mutated in the following way. If a center value is adapted, then its new value is uniformly randomly chosen amongst the values that lie within r^1_{cl} times the width of the ellipsoidal bound of the parent. If a stretch value is mutated, it is set uniformly randomly in the interval of size 50% times r^2_{cl} around the parental value. If an angle is mutated, the new angle is chosen uniformly randomly within an interval of $-.5\pi r^3_{cl}$ to $.5\pi r^3_{cl}$ plus the parental value. For example, given $r_{cl} = (.1, 2, .5)$ and the parental values for center, stretch, and angle are $(.5, .4)$, $(.1, .6)$, and (π), respectively, then the first center value will be chosen uniformly randomly in the interval $[.49, .51]$, the second one will lie within $[.34, .46]$, the first stretch value will lie within $[0, .2]$ and the second within $[0, 1.2]$, and the angle will lie within $[.75\pi, 1.25\pi]$. As done elsewhere [6], the center is constrained to lie within problem boundaries, stretch values may not be smaller than zero, and angles are constraint to rotate within 0 and 2π. Moreover, as for the SAM-rate values, the mutation range values were constraint to lie within 0 and 2.

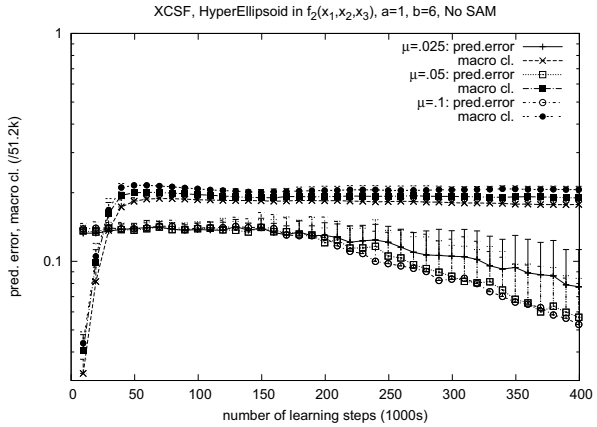
3.1 Parameter Independence

We now first evaluate XCSF's performance with self-adaptive mutation ranges similar to the one conducted for the SAM-rates. We again evaluate step-based and Gaussian-based mutation in the oblique sine function with various parameter settings.

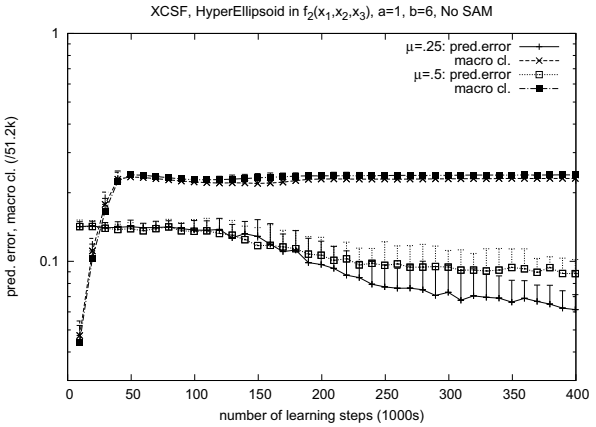
Figure 4 shows the resulting performance. As in the case of SAM-rates, XCSF with SAM-ranges alleviates high initial mutation rates (e,f). XCSF with step-based self-adaptation outperforms XCSF without self-adaptation in all runs (a,c,e). XCSF with Gaussian-based adaptation, on the other hand, requires a strong-enough signal, that is, a high-enough mutation rate, in order to self-adapt appropriately and thus solve the problem successfully (b,d,f). In sum, the results show that XCSF with step-based SAM-ranges solves the oblique sine problem most robustly and rather parameter independent. The best parameter settings are to allow maximum mutation rates ($\mu = 1$) and high self-adaptation (.5 or above).

3.2 Distributed Self-Adaptation

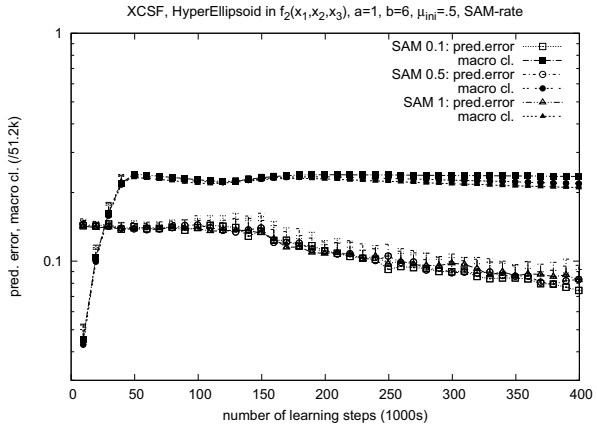
While the oblique sine function is an interesting problem, which demands a proper stretch and angular orientation of all classifiers, the whole problem space generally demands a similar strong stretch and orientation. Thus, both stretch and orientation need to adapt to a certain level independent of the problem subspace covered (cf. [6]). The radial-sine



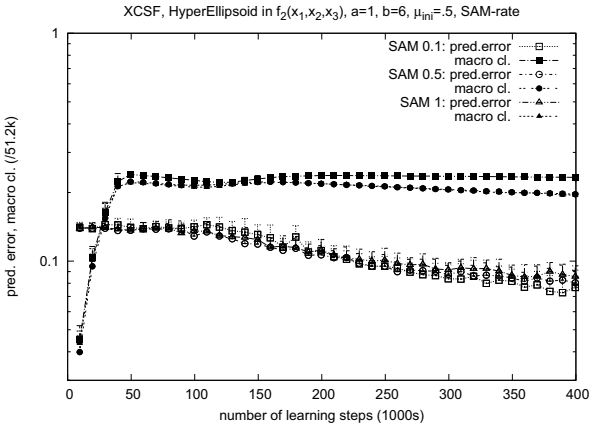
(a) XCSF without self-adaptive mutation, low μ



(b) XCSF without self-adaptive mutation, high μ



(c) XCSF with step-based self-adaptive mutation rates



(d) XCSF with Gaussian-based self-adaptive mutation rates

Figure 3: When no self-adaptive mutation rate is applied, XCSF cannot approximate the radial function problem (a,b). Even with self-adaptive mutation, performance hardly improves over time (c,d). Shown are the best runs, which had an initial mutation rate of $\mu_{ini} = .5$.

function introduced above (f_2) is harder because it demands a subspace-dependent ellipsoidal orientation and stretch.

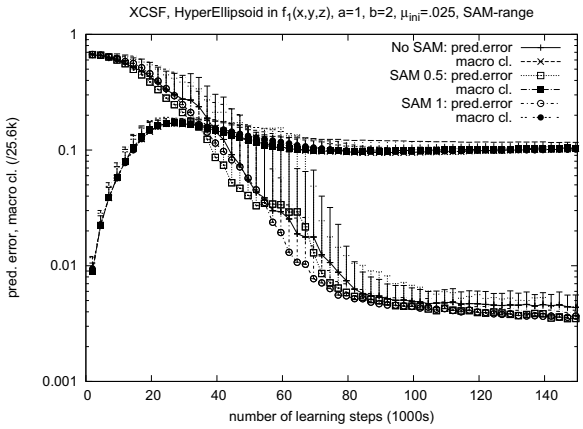
Figure 5 shows that XCSF with SAM is able to approximate the radial-sine function problem with higher accuracy, reaching a level of .035. The results indicate performance influences of two parameters, the mutation rate μ and the scaling parameter x for the SAM-ranges (cf. equations 1 and 2). Given a low mutation rate μ, the mutation range adaptation needs to be small as well to be successful. With larger mutation range adaptation values, the ranges may adapt to unsuitable values because the mutation ranges adapt also when no values are mutated. If a large mutation rate μ is chosen, however, larger mutation range adaptation values yield faster learning. In this case the slow adaptation of the range values may simply be too slow decreasing the learning speed.

Thus, to adapt suitably, SAM-ranges need to be balanced with the mutation rates. Small mutation rates can only work together with small mutation range adaptations. Larger mutation rates require larger range adaptations to maintain a good learning speed. As in the oblique sine function, the results further indicate that the mutation rate can be safely set to one when mutation ranges are self-adapted and self-adaptation is sufficiently strong to apply swiftly and effectively. In sum, the results show that parameter-dependent SAM-ranges can further improve XCSF's performance in function approximation problems. Performance becomes more independent of the chosen mutation rate and learning success becomes more reliable.

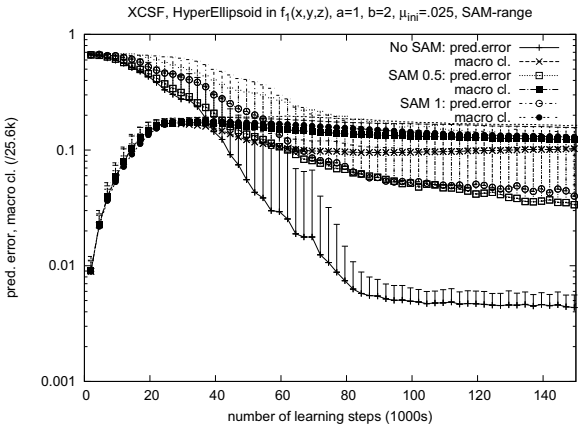
4. SUMMARY AND CONCLUSIONS

While self-adaptive parameters had been added to the ZCS and XCS classifier systems previously [4, 12], as well as to an anticipatory learning classifier system [3], this paper investigated self-adaptive mutation (SAM) in XCSF [16, 17]. It was shown that SAM-rates can alleviate the problem of inappropriate initial mutation rate settings. However, it was also shown that XCSF without or with SAM-rates evolves mediocre approximations for a more complex radial-sine function problem. The approximation can be further improved when SAM-ranges were applied. When mutation ranges were adapted independently for location, stretch, and orientation of ellipsoidal classifier conditions, then XCSF performance reached most accurate approximation results.

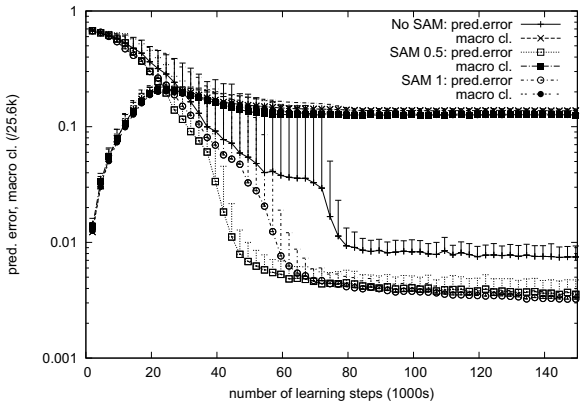
The results, in general, are promising making XCSF further independent of parameter settings by enabling it to



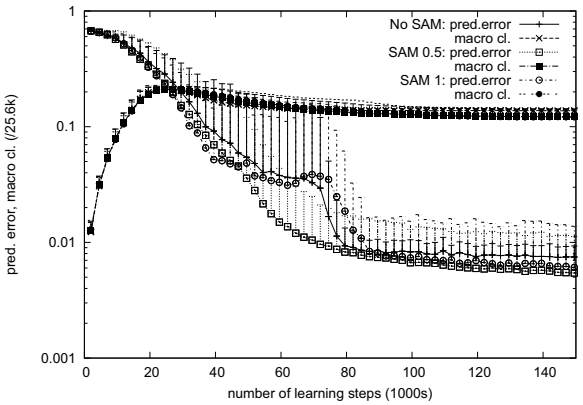
(a) $\mu_{ini} = .025$, Step-based SAM-range



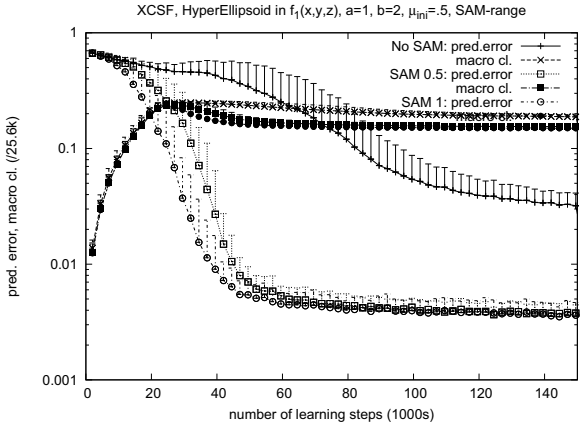
(b) $\mu_{ini} = .025$, Gaussian-based SAM-range



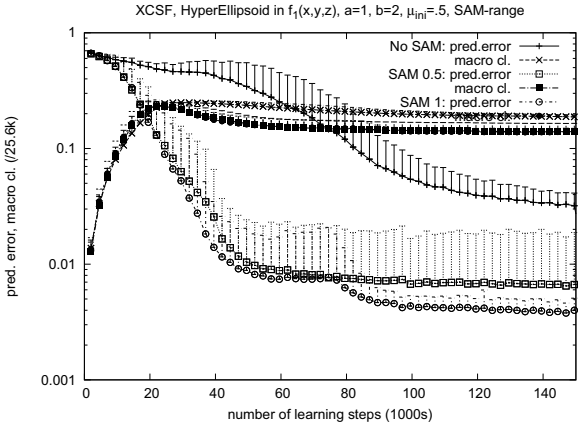
(c) $\mu_{ini} = .1$, Step-based SAM-range



(d) $\mu_{ini} = .1$, Gaussian-based SAM-range

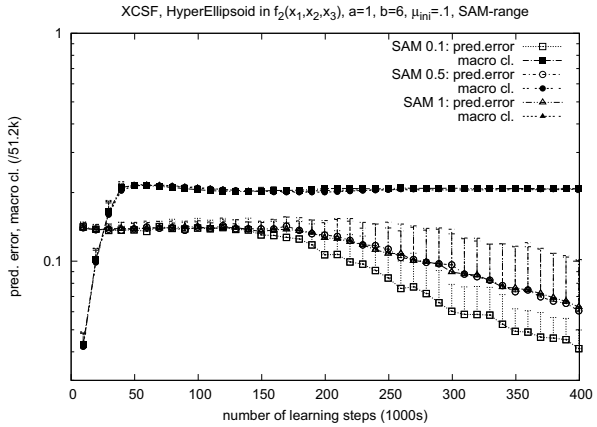


(e) $\mu_{ini} = .5$, Step-based SAM-range

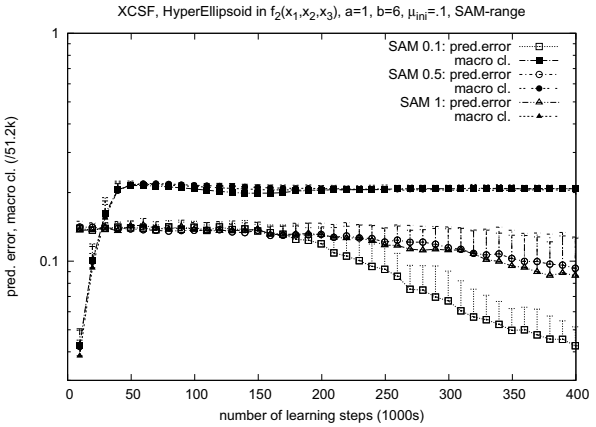


(f) $\mu_{ini} = .5$, Gaussian-based SAM-range

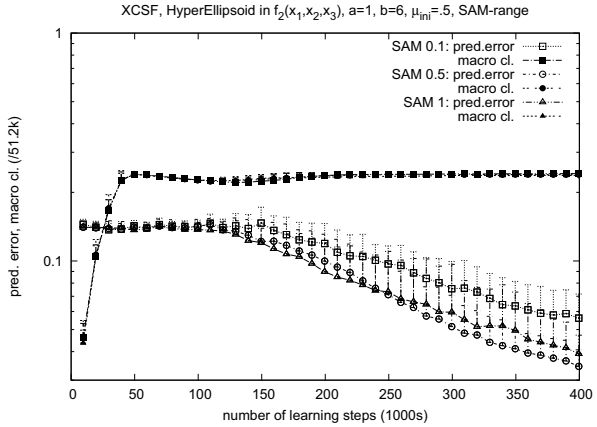
Figure 4: In the oblique sine function, XCSF with SAM-ranges and step-based adaptation performs equally well or better than without self-adaptation (a,c,e). With Gaussian-based adaptation, the mutation rate needs to be sufficiently high (that is, at least $\mu_{ini} = .1$) in order to adapt suitably (b,d,f). Given this is the case, though, also the Gaussian-based SAM-ranges outperform XCSF without self-adaptation.



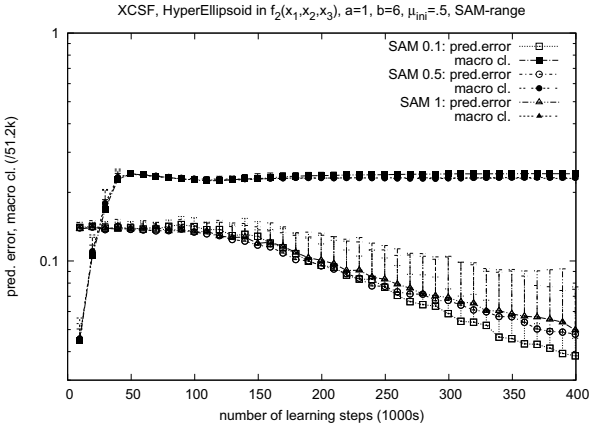
(a) $\mu_{ini} = .1$, Step-based SAM-range



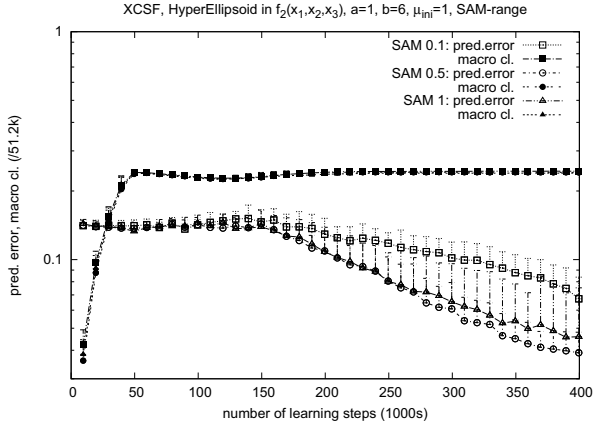
(b) $\mu_{ini} = .1$, Gaussian-based SAM-range



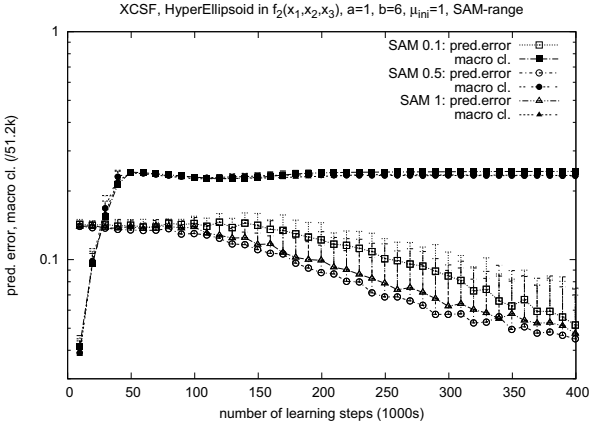
(c) $\mu_{ini} = .5$, Step-based SAM-range



(d) $\mu_{ini} = .5$, Gaussian-based SAM-range



(e) $\mu_{ini} = 1$, Step-based SAM-range



(f) $\mu_{ini} = 1$, Gaussian-based SAM-range

Figure 5: XCSF with SAM-ranges successfully approximates the Radial function reaching an error level of .02 as long as the overall mutation rate is sufficiently high and the scaling factor for the self-adaptation is sufficiently high as well.

self-adapt mutation ranges and thus necessary variational influences in the evolutionary process. The results and conclusions point towards a strong link between the evolutionary process in XCSF and evolution strategies [1]. The real-valued condition attributes in XCSF yield improved learning success when self-adaptation is enabled. Even more advanced self-adaptation techniques, such as covariance matrix adaptation (CMA) [10], may be included in XCSF. However, since XCSF is a distributed learning system that optimizes solutions within problem sub-niches, advanced CMA techniques such as multi-objective CMA [14] are likely to be necessary for a successful integration into the XCSF system. It could also be possible to include further information in the classifier adaptation process, such a informed specializations [5] or advanced recombination techniques [9]. Moreover, it remains to be shown if further differentiated self-adaptive mutation mechanisms such as self-adaptive mutation values for each conditional parameter might yield even better learning. Further experimental investigations as well as mathematical facet-wise models are necessary to solve these open research questions.

5. ACKNOWLEDGMENTS

The first two 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.

6. REFERENCES

[1] T. Bäck and H.-P. Schwefel. Evolution strategies I: Variants and their computational implementation. In G. Winter, J. Périaux, M. Gal'an, and P. Cuesta, editors, *Genetic algorithms in engineering and computer science*, chapter 7, pages 111–126. John Wiley & Sons, Chichester, 1995.

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

[3] L. Bull. On lookahead and latent learning in simple lcs. In J. Bacardit, E. Bernadó-Mansilla, and M. V. Butz, editors, *Learning Classifier Systems. International Workshops, IWLCS 2006-2007*. Springer-Verlag, Berlin Heidelberg, 2008.

[4] L. Bull, J. Hurst, and A. Tomlinson. Self-adaptive mutation in classifier system controllers. *From Animals to Animats 6: Proceedings of the Sixth International Conference on Simulation of Adaptive Behavior*, pages 460–467, 2000.

[5] M. V. Butz. *Anticipatory learning classifier systems*. Kluwer Academic Publishers, Boston, MA, 2002.

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

[7] M. V. Butz, T. Kovacs, P. L. Lanzi, and S. W. Wilson. Toward a theory of generalization and learning in XCS. *IEEE Transactions on Evolutionary Computation*, 8:28–46, 2004.

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

[9] M. V. Butz, M. Pelikan, X. Llorà, and D. E. Goldberg. Automated global structure extraction for effective local building block processing in XCS. *Evolutionary Computation*, 14:345–380, 2006.

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

[11] 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.

[12] J. Hurst and L. Bull. A self-adaptive XCS. In P. L. Lanzi, W. Stolzmann, and S. W. Wilson, editors, *Advances in learning classifier systems: Fourth international workshop, IWLCS 2001 (LNAI 2321)*, pages 57–73. Springer-Verlag, Berlin Heidelberg, 2002.

[13] J. Hurst and L. Bull. A neural learning classifier system with self-adaptive constructivism for mobile robot learning. *Artificial Life*, 12:1–28, 2006.

[14] C. Igel, N. Hansen, and S. Roth. Covariance matrix adaptation for multi-objective optimization. *Evolutionary Computation*, 15:1–28, 2007.

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

[16] S. W. Wilson. Function approximation with a classifier system. *Proceedings of the Third Genetic and Evolutionary Computation Conference (GECCO-2001)*, pages 974–981, 2001.

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