XCSF with local deletion: preventing detrimental forgetting

Martin Butz, Olivier Sigaud

XCSF with Local Deletion: Preventing Detrimental Forgetting

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

Olivier Sigaud
Institut des Systèmes Intelligents et de Robotique, Université Pierre et Marie Curie - Paris 6. CNRS UMR 7222, 4 place Jussieu,
F-75005 Paris, France
olivier.sigaud@upmc.fr

ABSTRACT

The XCSF classifier system solves regression problems iteratively online with a population of overlapping, local approximators. We show that problem solution stability and accuracy may be lost in particular settings – mainly due to XCSF's global deletion. We introduce local deletion, which prevents these detrimental effects to large extents. We show experimentally that local deletion can prevent forgetting in various problems – particularly where the problem space is non-uniformly or non-independently sampled. While we use XCSF with hyperellipsoidal receptive fields and linear approximations herein, local deletion can be applied to any XCS version where locality can be similarly defined. For future work, we propose to apply XCSF with local deletion to unbalanced, non-uniformly distributed, locally sampled problems with complex manifold structures, within which varying target error values may be reached selectively.

Categories and Subject Descriptors

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

General Terms

Algorithms, Theory, Performance

Keywords

XCSF, function approximation, local deletion, unbalanced problem sampling, manifold

1. INTRODUCTION

Most challenging function approximation problems arise when the function class is unknown, which requires an online approximation of the function surface from incoming samples. Many regression algorithms are furthermore challenged when the sampled values are non-uniformly or non-independently distributed. While facing these challenges, the applied regression algorithm is expected to yield results of a desired accuracy with a representation that approximates the function highly compactly. XCSF is a system that falls in this class of regression algorithms and that indeed also faces these challenges.

The XCS system was introduced in [13] and was extended to the real-valued function approximation system XCSF in [14]. In [1], XCSF was modified to develop hyperellipsoidal receptive fields, that is, Gaussian kernels. In [4], XCSF with hyperellipsoids was extended with effective mutation operators and compared to similar state-of-the-art machine learning techniques, such as the constructive incremental learning approach [8]. Most recently, it was shown that XCSF yields performance comparable or superior [11] to the locally weighted projection algorithm (LWPR) [12] – a well-known tool in the learning robotics literature. Here, we investigate the performance of XCSF further and compare standard XCSF with a modified version, in which GA deletion is applied locally. The challenges we are facing are due to observations that XCSF sometimes does not yield stable approximation surfaces – particularly while applying condensation [13, 4]. Moreover, we are interested in approximating functions whose problem spaces are sampled non-uniformly or non-independently. We show that XCSF with local deletion (referred to as XCSFld) yields more stable and accurate function approximations in various problem settings. Additionally, we study approximation performance in lower dimensional manifolds, which are embedded in higher dimensional input spaces.

We now first motivate and detail the new local deletion mechanism. Next, we provide results for various problem sampling cases in the crossed-ridge function [8, 11] and in the diagonal sine function [1]. It is shown that local deletion yields performance that is either comparable or better than that of XCSF with global deletion. We conclude that local deletion is a powerful tool to avoid detrimental forgetting and to focus local search in XCSF.

2. XCSF WITH LOCAL DELETION

The evolutionary algorithm in XCSF can be characterized as a steady-state GA. Upon GA invocation, usually two classifiers are selected for reproduction in the current match set $[M]_t$ at iteration t. Sometimes, it can also be advantageous to select a different amount of classifiers for reproduction

GECCO'11, July 12–16, 2011, Dublin, Ireland.

(cf. [9]). More importantly for this study, however, is the fact that global deletion is applied [13] – that is, classifiers are selected from the whole population for deletion.

This combination of local reproduction with global deletion yields a now well-documented evolutionary generalization pressure [3]: since the classifier conditions in the match set are on average more general (simply speaking, because more general classifiers will match more often on average) than the classifier conditions in the population as a whole, more general classifiers are favored during reproduction. The resulting generalization pressure has been quantified and verified exactly in various studies, cf. [3]. Recently, the same mechanism was also characterized in XCSF with respect to classifier volumes [4, 10] and most of the theory of XCS was carried over to XCSF.

However, global deletion may result in the deletion of important classifiers when the problem space is non-uniformly sampled. For XCS, it has been shown that when the problem space is covered with highly overlapping conditions, local, random walk competitions can take place amongst the overlapping classifiers [2], which may lead to detrimental forgetting of small subspaces.

In data mining problems, it was shown that unbalanced data-sets may also lead to detrimental forgetting. In this case, the θ_{GA} threshold was reduced automatically, which results in a reduced reproduction rate in over-sampled problem subspaces [7, 6]. However, the unbalance detection mechanism, which is used to tune the θ_{GA} threshold automatically, does not seem to be applicable in XCSF.

We propose to focus the deletion mechanism on the currently sampled, local sub-space, fostering local deletion. To avoid disrupting the generalization pressure of XCSF, mentioned above, a local deletion mechanism is necessary that does not favor more general classifiers. Deletion simply from the usual match set is not appropriate, because classifiers in the match set are on average more general than those in the whole population. Our mechanism deletes within the local sub-space, but it does not favor more general classifiers for deletion. The local subspace is determined by the condition of a randomly chosen classifier cl from the current match set. Classifiers whose condition center overlaps with the condition of cl are considered for deletion. Consequently, we scan the population for all those classifiers whose center of their condition are matched by the condition part of cl. The resulting subset of classifiers $[D]$ is then treated as the candidate list for deletion – as previously the whole population was treated as the candidate list for deletion. Thus, the usual roulette wheel selection with the normal match set size estimate and accuracy estimate-based voting mechanism is applied in $[D]$ – as it would be done in the population $[P]$ in the normal XCSF. That is, first the sum of all deletion votes in $[D]$ is calculated and a classifier is chosen for deletion by roulette wheel selection. Next, the numerosity of that classifier is reduced by one. If its numerosity equals zero, it is deleted from the population. Algorithm 1 specifies this mechanism in algorithmic form. In the remainder of this work, we refer to this modification by XCSF with local deletion (XCSFld).

Algorithm 1 Local Deletion

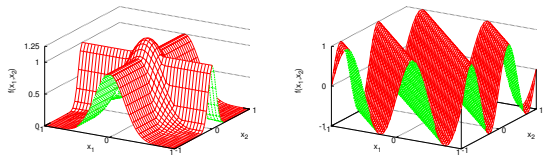
```
Select random classifier cl from [M].
[D] = ∅
for all c ∈ [P] do
    if cl does match center of c then
        add c to candidate list [D]
    end if
end for
DELETE FROM CANDIDATE LIST [D]
```

3. PERFORMANCE COMPARISON

We consider n-dimensional real-valued functions

$$f : \mathbb{R}^n \to \mathbb{R},$$

which are sampled iteratively, that is, each time step a sample $(\vec{x}, y)$ is given to the algorithm, where $y = f(\vec{x})$. The goal is to approximate the function surface online from the given samples. To increase the difficulty of the problem further, we add random Gaussian noise with a standard deviation of 0.001 to each sampled output value.



(a) Crossed Ridge (b) Sine

Figure 1: Surface plots of (a) the Crossed Ridge function f_1 and (b) the Sine function f_2.

The functions are defined in the interval $[-1, 1]^2$. Without loss of generality, the input space is normalized to $[0, 1]^2$ for XCSF.

1. The *Crossed Ridge* function contains a mix of linear and non-linear subspaces.

$$f_1(x_1, x_2) = \max\left[\exp(-10x_1^2),\ \exp(-50x_2^2),\ 1.25\exp(-5(x_1^2 + x_2^2))\right]$$

2. The *Sine* function is constant in the $(1, -1)$ direction but highly non-linear in the perpendicular $(1, 1)$ direction.

$$f_2(x_1, x_2) = \sin(2\pi(x_1 + x_2))$$

Figure 1 shows surface plots of the two functions.

We mostly use standard parameter settings for XCSF[1], We conduct ten independent runs for each reported experimental setup. The currently sampled mean absolute prediction error (MAE), the root mean square error (RMS) of points sampled in a grid of 21 by 21 locations, as well as the

[1]XCSF's parameters are set to $N = 4000$, $\varepsilon_0 = 0.002$, $\beta = 0.1$, $\delta = 0.1$, $\alpha = 1$, $\theta_{\text{GA}} = 50$, $\theta_{\text{del}} = \theta_{\text{sub}} = 20$, $\chi = 1$. The mutation probability for each attribute of the rotating hyperellipsoidal structures (center, stretch, and angle) are set to $\mu = 1/n = 0.2$. The center is mutated within the receptive field bounds, the stretches are decreased or increased in size maximally doubling or halving their current size; the angles are uniformly changed by maximally 45. The initial radius of receptive fields is taken uniformly random from $[0.00, 1]$. GA subsumption is applied. Condensation is applied after 80% of the learning iterations.

number of distinct classifiers (number of macro-classifiers) and the average generality (volume) of the classifiers in the population are reported. In the case of Gaussian sampling, only those grid points are considered that lie in the 95% confidence interval.

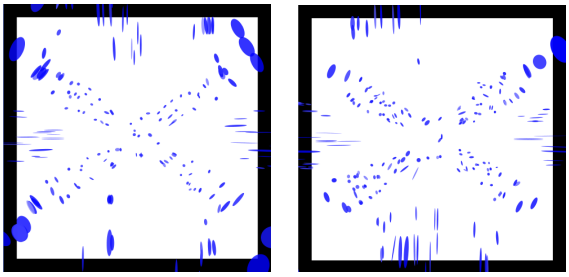
We report results with various sampling approaches: First, standard uniform sampling is applied. Next, random walk sampling is investigated, where each successive sample depends on the previous sample. We add Gaussian noise to the previous sample to derive the next one. If the resulting x_1 or x_2 value lies outside of the input space $[0, 1]$, a different Gaussian noise value is sampled. Note that such sampling mimics a continuous interaction with an environment – previously, for example, investigated in multi-step reinforcement learning problems (mazes) with XCS, where teletransportation was proposed to improve learning [5]. However, teletransportation cannot be used when the system is supposed to learn fully autonomously.

Moreover, we report results where sampling is restricted to a ring manifold in the two dimensional space, centered on .5, .5 and with a minimum radius of .3 and a maximum radius of .4, within which once again random walk sampling is applied. Furthermore, we report results on Gaussian problem sampling where samples are either generated by a Gaussian process with center .5, .5 and variable widths or where we apply a Gaussian ring sampling, in which case the average ring radius is sampled from a Gaussian process with mean .3 and particular standard deviations and the angle is chosen uniformly randomly between 0 and 2π.

3.1 Crossed Ridge Results

The results in figures 4(a), 4(b) suggest that XCSF with local deletion yields a more stable condensation process during uniform sampling in the crossed ridge function. During learning, learning progress and achieved values seem generally comparable. Similar differences during condensation as well as slight learning advantages during learning can be identified also for different random walk sampling cases (figures 4(c)-4(f)). Also the behavior of the generalization values and the population sizes are generally comparable. The number of macro-classifiers stays slightly higher when local deletion is applied – probably an additional indicator for the prevention of detrimental forgetting. Generally these results suggest that XCSF can deal rather well with sampling based on random walk. Only during condensation, during which neither mutation nor crossover operators are applied, the danger for detrimental forgetting is larger in XCSF. Figure 2 shows the developed classifier condition structures, which cover the problem input space, for the random walk sampling with $\sigma = 0.1$ after condensation. Local linearities appear nicely exploited. However, in this case a difference between XCSF and XCSFld cannot be deduced.

When sampling the Crossed Ridge function on a ring, figures 5(a), 5(b) show that performances are generally comparable. When Gaussian sampling is applied (figures 5(c), 5(d)), the same robustness is observable when XCSFld is applied. However, in this case a similar stable solution takes slightly longer to develop. Finally, when Gaussian ring sampling is applied (figures 5(e), 5(f)), we can observe the evolution of a final solution with lower prediction error and higher average rule generality when XCSFld is applied – indicating that detrimental forgetting due to global deletion in XCSF is prevented – once again particularly strongly during condensation. Note also that these results show for the first time that XCSF – regardless if deleting globally or locally – can solve an approximation problem in a local manifold better than when approximating a function in a larger problem space (compare prediction errors in Figure 4 with those in Figure 5).



(a) XCSF (b) XCSFld

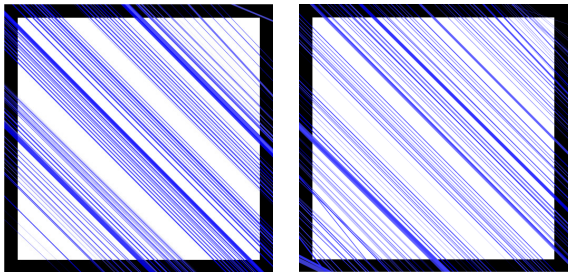
Figure 2: Structure of ellipsoids of XCSF and XCSFLD after condensation in exemplary runs with random walk sampling, standard deviation $\sigma = 0.1$, in the crossed ridge problem. Shown are contour plots of 20% of the actual size of the classifier receptive fields with respect to matching. Darker blue indicates higher fitness values.

In sum, the results show that XCSFld does not yield any worse performance than normal XCSF in any of the investigated cases in the Crossed Ridge function. However, in most cases, the prediction errors stay significantly more stable during condensation.

3.2 Sine Results

The sine function is a rather tough case to learn a fully accurate model that achieves an error below $\epsilon_0 = .002$ with $N = 4000$ classifiers. In the uniform sampling case (figures 6(a) and 6(b)) XCSFld yields better RMS values during learning and much more stable final solution sustenance, preventing detrimental forgetting. This result indicates that the local ellipsoidal structures, which typically orient themselves diagonally in space in this sine function, are strongly locally overlapping. When random walk sampling is applied, the problem space is more accurately approximated with XCSFld – as indicated by the RMS measure — and huge prediction error differences after condensation can be observed (figures 6(c)-6(f)). In the most extreme case, the final RMS for normal XCSF yields an error above 0.4 while the error of XCSFld stays below 0.004 – two orders of magnitude better performance due to the local deletion mechanism (see figures 6(c) and 6(d)). Figure 3 again shows the developed classifier condition structures after condensation for the random walk sampling with $\sigma = 0.1$. The local linearities are clearly exploited. In the case of XCSF, particularly the coverage at the two corners (0,0) and (1,1) has been lost. XCSFld maintains a full coverage, preventing detrimental forgetting.

Similar results can also be observed in the other sampling cases (Figure 7). While the random walk in ring sampling does not yield extreme differences (figures 7(a), 7(b)), in the case of Gaussian Centered sampling again detrimental forgetting is prevented during condensation due to the local deletion mechanism (figures 7(c)-7(f)).



(a) XCSF (b) XCSFld

Figure 3: Structure of ellipsoids of XCSF and XCSFLD after condensation in the sine function.

4. SUMMARY AND CONCLUSIONS

This paper proposes for the first time that the "global deletion dogma" in XCS may be modified to prevent detrimental forgetting. XCSF appears to be particularly prone for detrimental forgetting when condensation is applied and when classifiers are locally highly overlapping. While condensation may be disregarded, it is certainly a very useful tool to extract a final, highly compact problem solution representation. Moreover, several results indicate that local deletion may also be advantageous during learning. However, for standard function approximation problems, the effect appears not to be as significant. Thus, partially the results also point out that XCSF indeed is a very robust, iterative, locally weighted regression system. However, local deletion appears to expand this robustness beyond the original capabilities without any apparent drawback. Indeed, we ran many more experiments with other step sizes in the random walk experiments, other Gaussian noise sampling widths, and XCSF parameter modifications, which all suggested the same tendencies reported here.

In conclusion, local deletion can improve problem solution sustenance in XCSF and thus the overall performance of the system, by preventing detrimental forgetting. A most exciting perspective that we believe opens up, besides this performance improvement, however, is that this local deletion mechanism may be also used to selectively learn higher accurate and lower accurate approximations in different problem subspaces. If this is desired, local deletion is expected to enable the selective improvement of the developing problem approximations. The selectivity may, for example, be determined by additional importance indicators, such as reward values. Thus, future research should further elaborate XCSFld and particularly investigate its potential in tasks where highly non-uniformly varying target errors are strived for in different problem subspaces.

Acknowledgments

The first author acknowledges funding from the Emmy Noether program (German Research Foundation, DFG, BU1335/3-1) and thanks the COBOSLAB team.

5. REFERENCES

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

[2] M. V. Butz, D. E. Goldberg, P. L. Lanzi, and K. Sastry. Problem solution sustenance in XCS: Markov chain analysis of niche support distributions and the impact on computational complexity. *Genetic Programming and Evolvable Machines*, 8:5–37, 2007.

[3] 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.

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

[5] P. L. Lanzi. An analysis of generalization in the XCS classifier system. *Evolutionary Computation*, 7(2):125–149, 1999.

[6] A. Orriols-Puig and E. Bernadó-Mansilla. Evolutionary rule-based systems for imbalanced data sets. *Soft Computing - A Fusion of Foundations, Methodologies and Applications*, 13:213–225, 2009.

[7] A. Orriols-Puig, E. Bernadó-Mansilla, D. E. Goldberg, K. Sastry, and P. L. Lanzi. Facetwise analysis of xcs for problems with class imbalances. *IEEE Transactions on Evolutionary Computation*, 13(5):1093–1119, 2009.

[8] S. Schaal and C. G. Atkeson. Constructive incremental learning from only local information. *Neural Computation*, 10:2047–2084, 1998.

[9] P. O. Stalph and M. V. Butz. How fitness estimates interact with reproduction rates: Towards variable offspring set sizes in xcsf. In J. Bacardit, W. Browne, J. Drugowitsch, E. Bernadó-Mansilla, and M. V. Butz, editors, *Learning Classifier Systems*, volume 6471 of *Lecture Notes in Computer Science*, pages 47–56. Springer Berlin / Heidelberg, 2010.

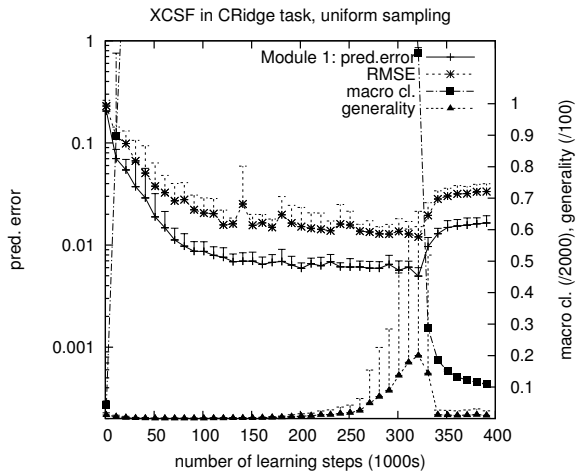
[10] P. O. Stalph, X. Llorà, D. E. Goldberg, and M. V. Butz. Resource management and scalability of the XCSF learning classifier system. *Theoretical Computer Science*, 2010. DOI: 10.1016/j.tcs.2010.07.007.

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

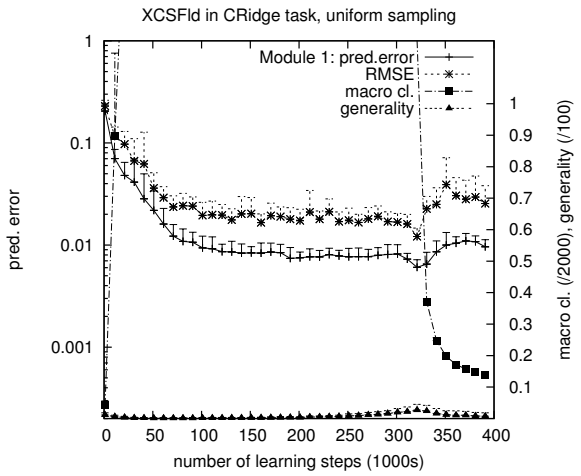
[12] S. Vijayakumar, A. D'Souza, and S. Schaal. Incremental online learning in high dimensions. *Neural Computation*, 17:2602–2634, 2005.

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

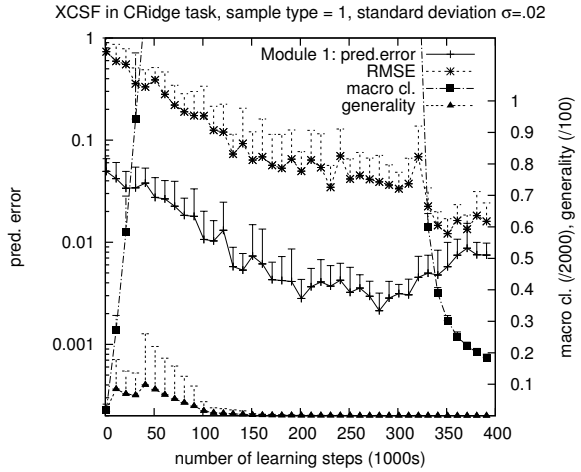
[14] S. W. Wilson. Classifiers that approximate functions. *Natural Computing*, 1:211–234, 2002.



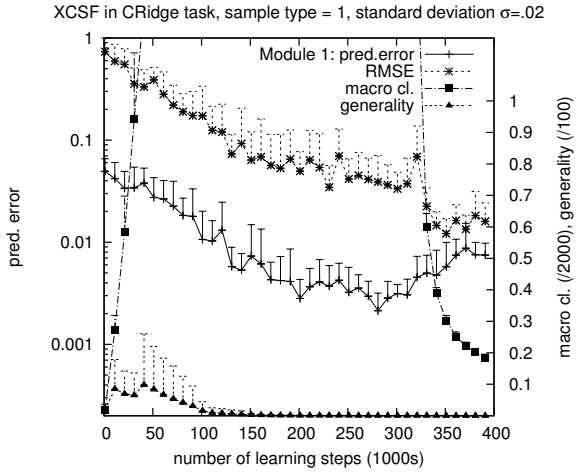
(a) XCSF: Uniform Sampling



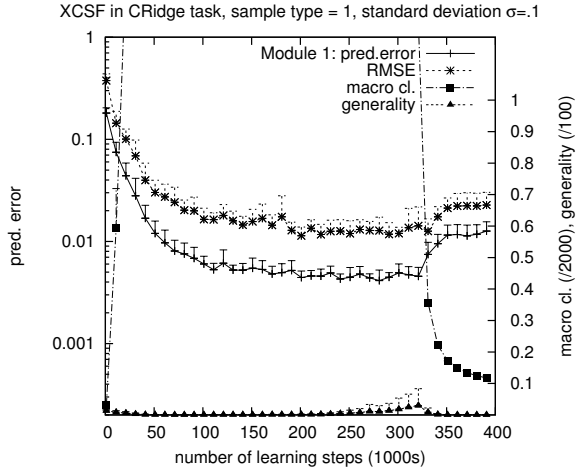
(b) XCSFld: Uniform Sampling



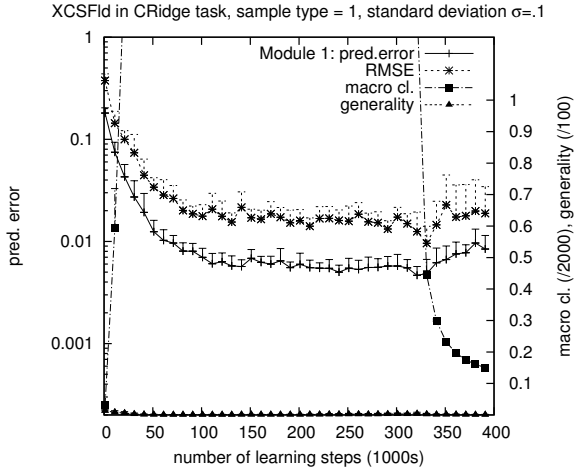
(c) XCSF: Rand.Walk Sampling, $\sigma = .02$



(d) XCSFld: Rand.Walk Sampling, $\sigma = .02$

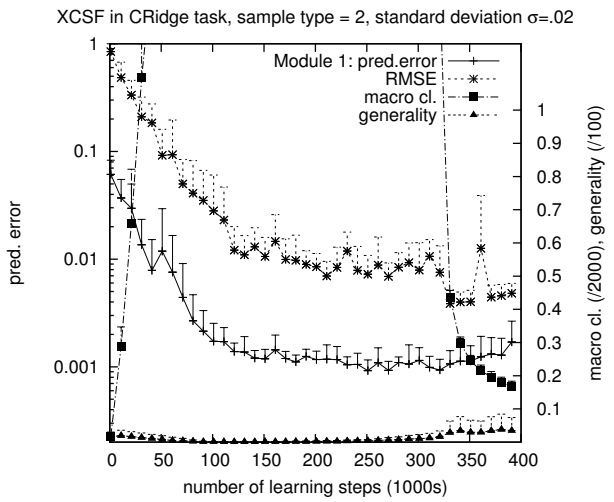


(e) XCSF: Rand.Walk Sampling, $\sigma = .1$

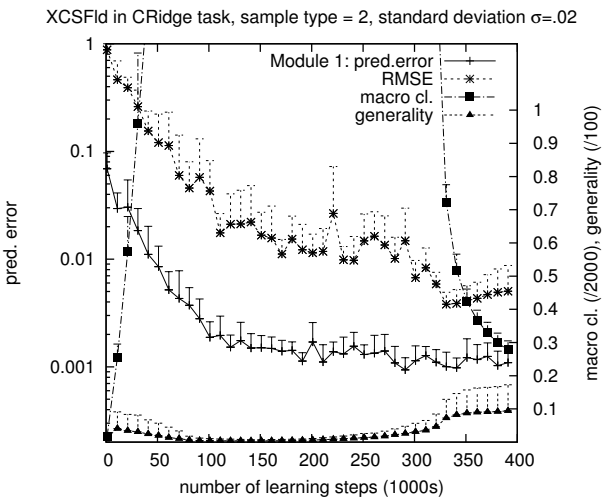


(f) XCSFld: Rand.Walk Sampling, $\sigma = .1$

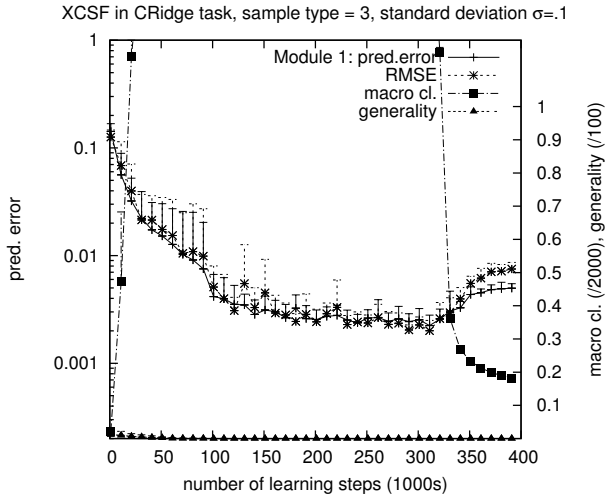
Figure 4: Crossed Ridge Function: Uniform Sampling and Random Walk Sampling. Condensation starts after 320k learning iterations. Reported are online prediction error and average root mean square prediction error (both log-scaled) as well as the number of distinct (that is, macro) classifiers and the average generality of classifier conditions.



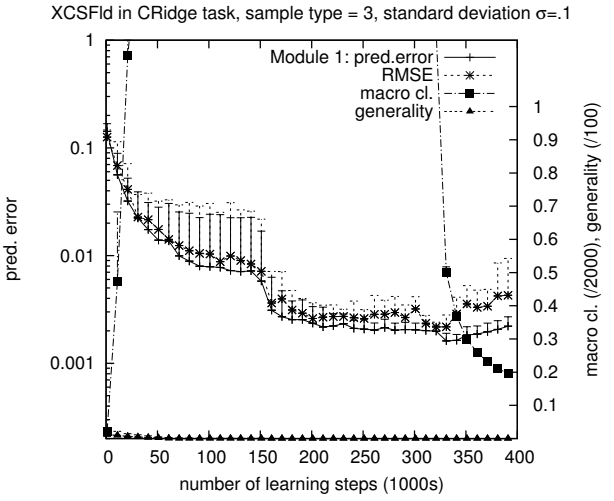
(a) XCSF: Random Walk in Ring



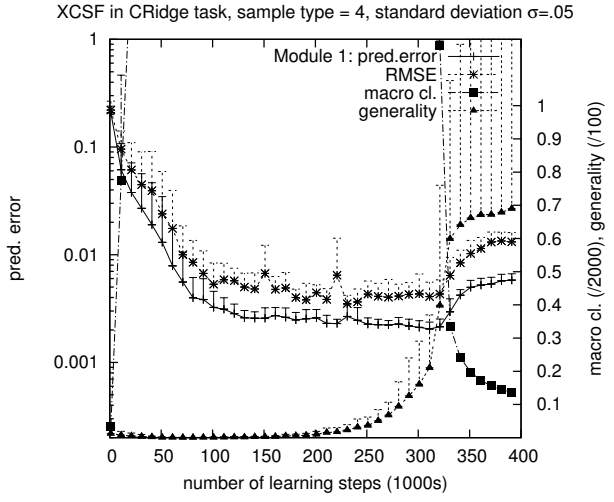
(b) XCSFld: Random Walk in Ring



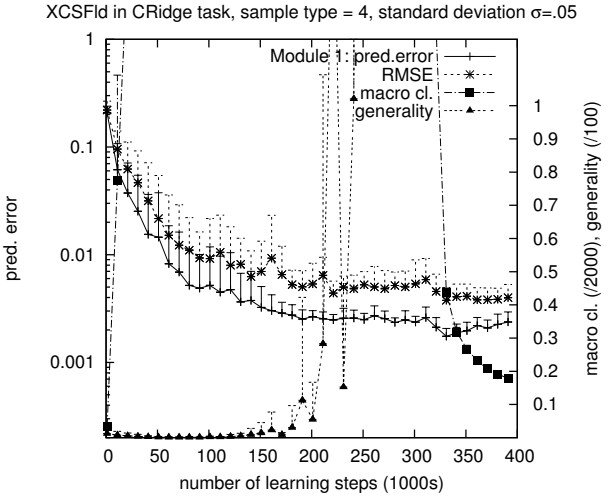
(c) XCSF: Gaussian Centered



(d) XCSFld: Gaussian Centered

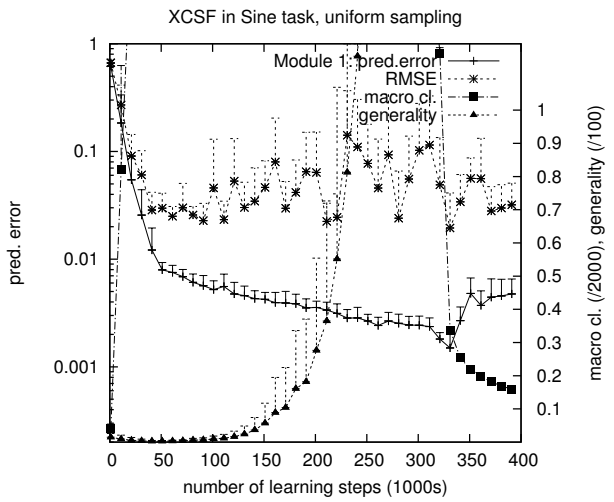


(e) XCSF: Gaussian Ring Sampling

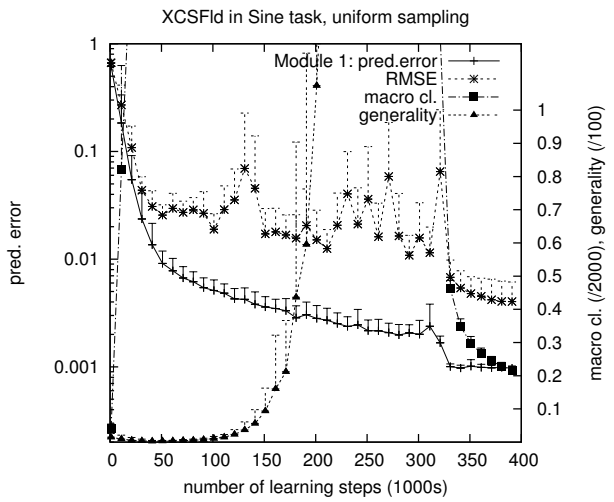


(f) XCSFld: Gaussian Ring Sampling

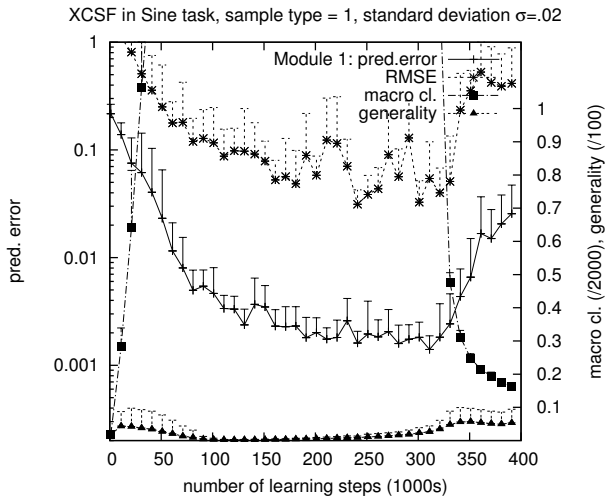
Figure 5: Crossed Ridge Function: Random Walk in Ring, Gaussian Centered, and Gaussian Ring Sampling



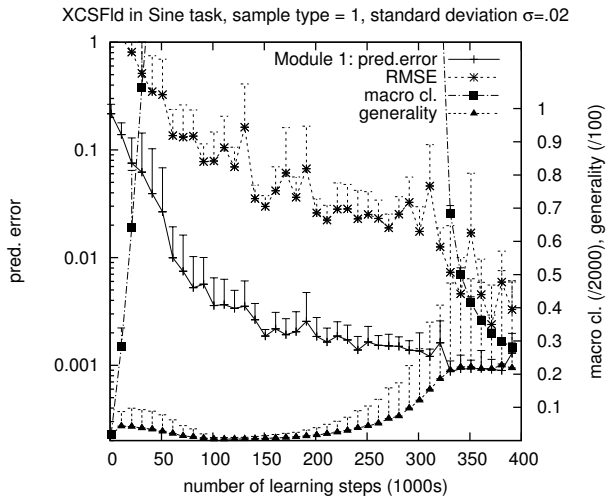
(a) XCSF: Uniform Sampling



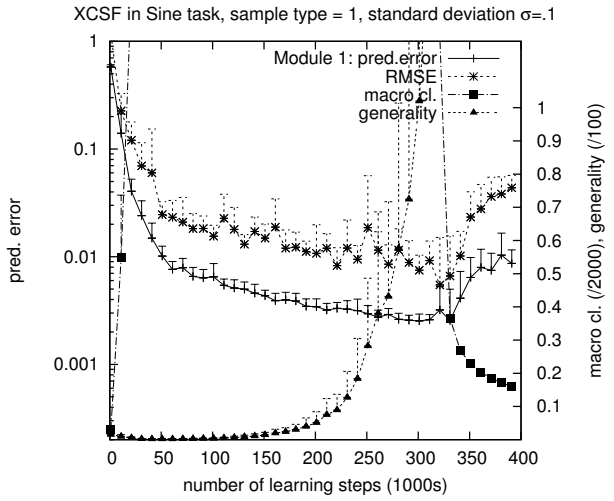
(b) XCSFld: Uniform Sampling



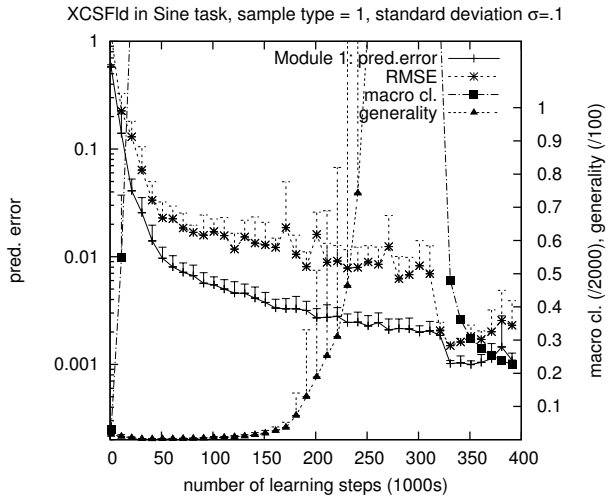
(c) XCSF: Rand.Walk Sampling, $\sigma = .02$



(d) XCSFld: Rand.Walk Sampling, $\sigma = .02$

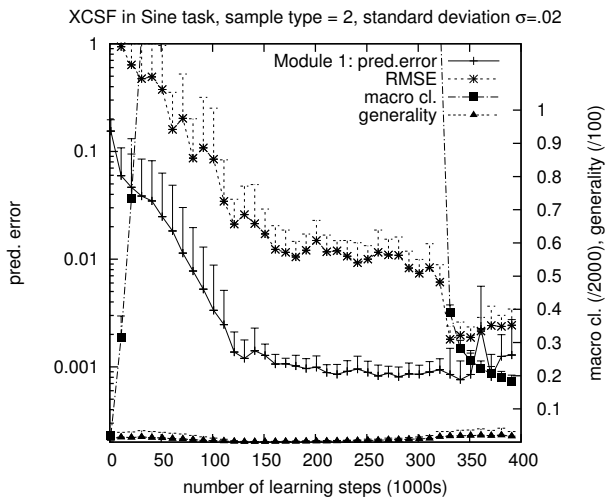


(e) XCSF: Rand.Walk Sampling, $\sigma = .1$

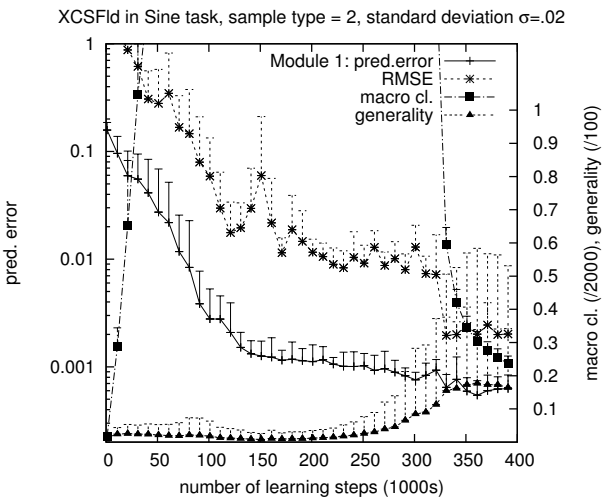


(f) XCSFld: Rand.Walk Sampling, $\sigma = .1$

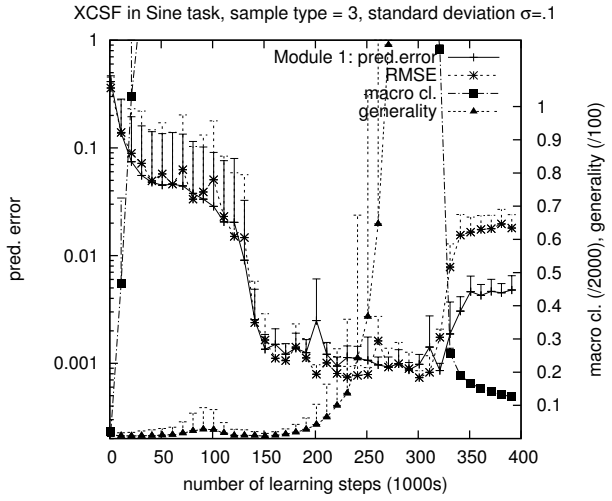
Figure 6: Sine Function: Uniform Sampling and Random Walk Sampling



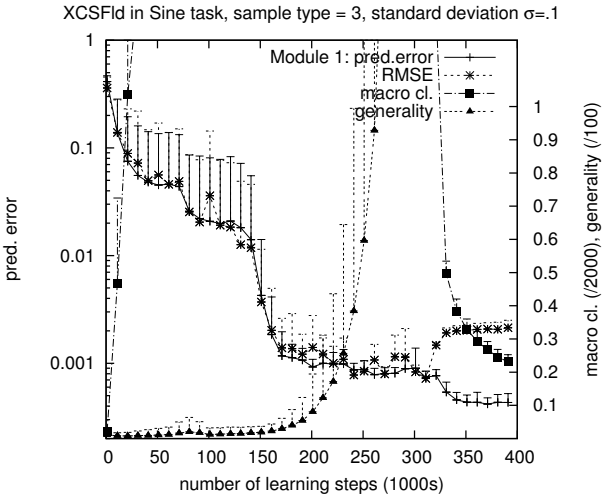
(a) XCSF: Random Walk in Ring



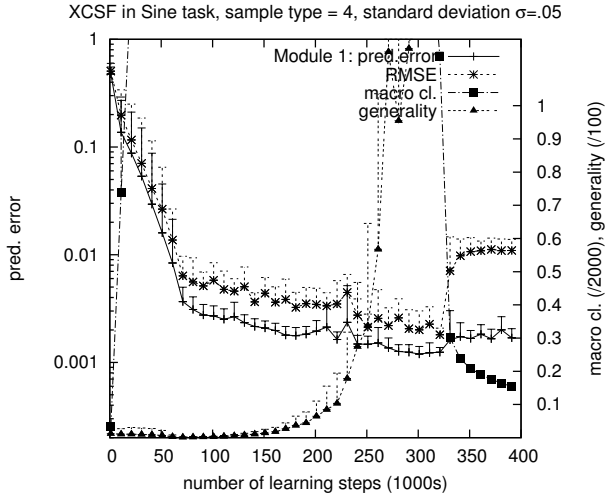
(b) XCSFld: Random Walk in Ring



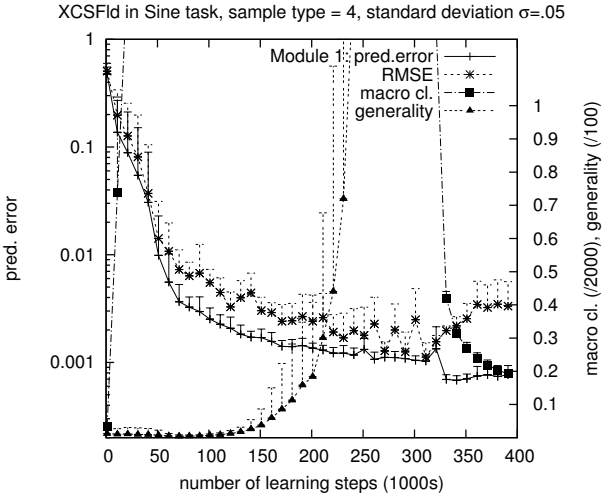
(c) XCSF: Gaussian Centered



(d) XCSFld: Gaussian Centered



(e) XCSF: Gaussian Ring Sampling



(f) XCSFld: Gaussian Ring Sampling

Figure 7: Sine Function: Random Walk in Ring, Gaussian Centered, and Gaussian Ring Sampling