Toward Curious Learning Classifier Systems: Combining XCS with Active Learning Concepts

Anthony Stein
Organic Computing Group,
University of Augsburg, Germany
anthony.stein@informatik.
uni-augsburg.de

Roland Maier
University of Augsburg, Germany
roland.maximilian.maier@student.
uni-augsburg.de

Jörg Hähner
Organic Computing Group,
University of Augsburg, Germany
joerg.haehner@informatik.
uni-augsburg.de

ABSTRACT

This paper proposes a novel approach to enhance the rather reactive knowledge generation process in Learning Classifier Systems (LCS) toward a more proactive means. We describe how concepts from the domain of Active Learning can be adapted to XCS's algorithmic structure to introduce 'curiosity'. The overall goal is to allow LCSs to build up new knowledge before it is actually requested during the online learning process. We deem such a methodology meaningful in scenarios where data samples are distributed non-uniformly and partially sparse over the input space. Such data imbalances result in gaps within the knowledge base, i.e. an LCS population. We underpin the general potential of our approaches by presenting preliminary results on a realistic data set from the domain of medical diagnosis as well as on a novel toy problem.

CCS CONCEPTS

•Computing methodologies → Active learning settings; Machine learning approaches; Rule learning;

KEYWORDS

Learning Classifier System, Knowledge Gaps, Active Learning, Oracle, Uncertainty Sampling, Query Synthesis

ACM Reference format:
Anthony Stein, Roland Maier, and Jörg Hähner. 2017. Toward Curious Learning Classifier Systems:
Combining XCS with Active Learning Concepts. In *Proceedings of GECCO '17 Companion, Berlin, Germany, July 15-19, 2017,* 8 pages.
DOI: http://dx.doi.org/10.1145/3067695.3082488

1 INTRODUCTION

Learning Classifier Systems (LCS) have gained plenty of research attention since their invention by John Holland in 1976 [10]. Initially designed to solve binary encoded reinforcement learning tasks, today many applications to real-world problems can be found in the literature. For instance, Goldberg applied LCS to simulated gas pipeline control [8]. An application to robot arm control was reported by Stalph and Butz in [28]. Bull et al. proposed the application of an LCS to traffic management in [3]. In Prothmann et al. [22], a strongly modified LCS that changes and constrain the generalizing nature of the conventional system is presented and applied to adapt traffic lights at urban intersections. What such real-world scenarios typically have in common is the complexity of the underlying problem space. Not all possible states are known a priori what necessitates learning during the system's lifetime. In consequence, the system has to cope with unforeseen or not anticipated situations at runtime.

LCSs are flexible, evolutionary rule-based online machine learning systems. These systems evolve a population $[P]$ of IF-THEN rules (*classifiers*) that partition the input space X and thus approximate the problem space locally. A steady-state niche *genetic algorithm* (GA) optimizes the classifier coverage of X globally in terms of maximum generality and maximum *strength* or else *accuracy*. By nature, LCSs, and more specifically the probably mostly investigated derivative, Wilson's *Extended Classifier System* [1] (XCS) [37], are capable of dealing with the aforementioned challenges. A mechanism called *covering* assures that in any situation at least one matching IF-THEN production rule (i.e. a *classifier*) is created. However, these classifiers are usually initialized with partially predefined initial but also randomly selected values. Furthermore, the covering mechanism seems to be a rather reactive behavior. Besides the covering mechanism, a second component is responsible for creating new knowledge in form of classifiers – a steady-state niche GA. Well-proven classifiers are selected to be reproduced, recombined and slightly mutated to search the local neighborhood of the environmental niche for a more suitable coverage. 'More suitable' in this context means that a globally optimal subregion of the input space is found which at the same time yields the maximum accuracy in predicting the correct (re)action to the observed environmental stimulus. Even the latter mechanism can be interpreted to act rather reactive since it is executed periodically on the basis of the last time it was invoked in the same environmental niche and selects already well-performing classifiers to be reproduced among rules that match the current situation.

Knowledge Gaps. Our overall research goal is to shift the means of building up novel knowledge within LCSs from a reactive to a *proactive* process. Therefore, we enable LCS to be 'curious', [2] i.e. that it can identify gaps in the population, or else its knowledge base. Hence, we first have to define the term *Knowledge Gap* (KG):

Definition 1. *A Knowledge Gap is a certain region within a learning algorithm's knowledge base $\hat{K}$ that is not covered by any experience the algorithm made so far.*

GECCO '17 Companion, Berlin, Germany

DOI: http://dx.doi.org/10.1145/3067695.3082488

[1] We assume a certain degree of familiarity with XCS in this paper and allow us to refer the reader to [6, 37] for a more detailed description of XCS and all algorithmic steps. We further note that we make use of a specific extension of XCS to allow for real-valued inputs, called XCSR [38].
[2] Computational curiosity was recently reviewed more thoroughly elsewhere [41].

In our work, we formally interpret a knowledge base $\hat{K}$ as an approximation of the mapping $X \rightarrow O$, i.e. from the input space X to any output from an output space O, regardless of whether the output is a scalar reward, an error term or a specific prediction. In a *markov decision processes*, X is extended by an action space A. Thus, $\hat{K}$ is an incrementally learnt approximation of the problem space $PS: X \times A \rightarrow O$. If we solely consider Definition 1, KGs would only occur at the very beginning of a learning task, when the knowledge bases, e.g. the population of an LCS, is empty. The covering mechanism in combination with a default generality parameter initially set to a high value might prevent the appearance and thus the identification of KGs (according to Def. 1) in later phases of the learning progress. Accordingly, we have to extend our definition:

Definition 2. *We classify a region/niche within $\hat{K}$ also as KG, when it is only represented by knowledge of low quality.*

Now, not only the disappearance of knowledge $k \notin \hat{K}$ determines a KG, but also does each knowledge element $k \in \hat{K}$ that does not exceed a threshold θ_q regarding a quality metric $q(k)$ to be defined.

As stated above, in an LCS, knowledge is represented by a population of IF-THEN rules, also termed classifiers, that maintain certain statistics about their quality. There exist different approaches to determine quality. The most investigated variants are: (1) basing the quality on the *predicted reward* or *strength*, e.g. [36]. (2) accuracy-based LCSs, such as XCS [37] or UCS [2] that judge on the basis of a classifier's accuracy in predicting reward. In the LCS context, using the mean accuracy of the population as a threshold and a particular classifier's accuracy as the quality metric is imaginable.

We assume KGs to arise mainly due to the following reasons: (1) *Imbalanced data*, resulting in class imbalances [9] and small disjuncts [35], and (2) *Non-uniform input distributions* determining the sampling of the input space X [5]. In dynamic real world environments, the input space distributions are supposed to change during the system's lifetime. This is a specific form of concept drift, also called *virtual/covariate drift* [34]. More formally, $P_t(Y = y) \neq P_{t+n}(Y = y)$ holds, where $t \neq t + n$ are certain points in time, $P(\cdot)$ denotes the probability distribution and Y is a random variable whose outcomes y are instances from the underlying input space. To summarize, we interpret each niche/region within a learning algorithm's knowledge base $\hat{K}$, that follows at least one of the above definitions, as KG. So our focus is set on enabling LCSs in general to better cope with KGs by identifying them during the system's lifetime and, on the other hand, allow for a proactive closure of KGs beforehand the system performance is negatively affected.

Contribution. In this paper, we restrict our consideration to accuracy-based LCS, more precisely to the XCS. Our goal is to propose a proactive knowledge creation strategy by endowing XCS itself with the capability to decide which regions in its knowledge base are only insufficiently represented yet and, thus, to autonomously identify KGs. To achieve this, we borrow concepts from the domain of *Active Learning* (AL) [25]. More precisely, we adapt techniques called *Least Confident Uncertainty Sampling* [14] as well as *Query-by-Committee* [26] to integrate AL into XCS' algorithmic structure and thus implement some form of curiosity. A further goal is to make XCS building up knowledge (i.e. classifiers) proactively, that is, before it is actually requested. Therefore, we propose to rely on so-called *Query Synthesis* approaches first presented in [1]. So far, we limited our investigations to a fully randomized approach.

The remainder of this paper is structured as follows: In Section 2, we review related work. Section 3 presents our first attempt to make XCS curious in the sense that it strives to learn more about regions which define a KG. We demonstrate the promising potential of our approach based on the results of preliminary experiments in Section 4. In Section 5, we conclude with a discussion about gained insights and with an overview of future work.

2 RELATED WORK

Injecting knowledge from external sources is not a novel means to enhance the performance of LCS. Urbanowicz et al. propose to use expert knowledge to guide the discovery processes of LCSs in [32, 33] and applied it to the problem domain of genetic association. The authors show that using expert knowledge can significantly improve learning efficiency of UCS. This approach differs from our work in that probabilities to influence the specificity of newly generated classifiers are derived in a preprocessing step. In our work, we strive for guiding the learning process of XCS online.

In [17, 18], Najar et al. presented the *Socially Guided XCS* and the *Social-Value XCS*. The authors introduced a model-based learning approach, where the task model is supported by a social model, as well as an contingency model that serves as bridge between the former two. Their system is applied to a multi-step reinforcement learning task where a robot is asked to decide which button to press according to a visual stimulus. A teacher can guide the robot's decision by means of teaching signals, e.g. pointing to the correct button. The authors found that the learning success can be improved in terms of the total amount of received reward, the number of steps until the robot finds the correct button, as well as the compactness of the evolved rule-base. In contrast to the method proposed in this paper, the Socially Guided XCS is a model-based reinforcement learner that may, or may not be taught by a human teacher. In our work, we allow the XCS to be curious about states where it could not gain a sufficient level of experience so far. Furthermore, the external source of knowledge is not necessarily assumed to be a human teacher (cf. Sect. 3).

By allowing XCS to build up knowledge before it is requested, its input space coverage strategy is affected. Nakata et al. raised the question of "How should Learning Classifier Systems cover a state-action space?" [19]. They introduced various ways to combine the advantages of accuracy-based LCS and strength-based LCS which were also investigated by Kovacs in [12]. A learning strategy to create a so-called *weighted complete action map* is introduced that enables XCS to evolve a population that is complete in the sense of learning the entire $X \times A \rightarrow O$ mapping, but assigns more classifiers to the highest-return actions for each state.

The question on the learning capabilities of XCS and UCS on imbalanced data where thoroughly investigated and theoretically modeled by Orriols-Puig et al. [21]. In their work, mainly the issues of *rare classes* and *rare cases* are focused and alleviated by online adaptation of XCS parameters such as the learning rate β and the threshold determining the GA activation θ_{GA}.

Butz and Sigaud propose to change the means of classifier deletion in [5]. The authors introduce *local deletion* to prevent XCSF

from detrimental forgetting of certain, already covered subspaces within the knowledge base ($[P]$) in non-uniformly and independently sampled regression problem domains. From our viewpoint, such forgotten subspaces constitute KGs.

Another related branch of research can be found in the domain of evolutionary algorithms (EAs). Lehman and Stanley introduced the *Novelty Search* algorithm in [13]. Novelty search replaces the objective or else fitness function of EAs with a novelty metric that forces the search for behavioral novelty. A more thorough investigation of their definition of novelty might be also fertile for our work. However, in contrast to the work of Lehman and Stanley we search for KGs in the evolved knowledge bases of learning algorithms instead of explicitly seeking novel behaviors.

3 ACTIVE LEARNING CLASSIFIER SYSTEMS

Active Learning (AL) [7, 25] is a semi-supervised machine learning paradigm that has been proven to enhance the efficiency of supervised learning algorithms in various classification tasks (cf. e.g. [23]). It is assumed that only a small labeled training set $\mathfrak{L}$ is available due to high labeling costs or rather efforts to obtain training examples of the form $(\vec{x}, y)$. Besides $\mathfrak{L}$, there is also a large unlabeled data set $\mathfrak{U}$ assumed since obtaining unlabeled samples is supposed to be inexpensive. With AL, the learning algorithm is endowed with the decision capability to decide which unlabeled instance $\vec{x} \in \mathfrak{U}$ should be selected to be sent to an omniscient and omnipresent oracle that obtains the correct label. This setting is a specific form of AL called *pool-based* AL. Besides pool-based AL, in essence there exist two further approaches – *stream-based* AL and *query synthesis*. In this work, we restrict our focus to pool-based AL, where the match set $[M]$ of XCS serves as pool of 'unlabeled samples' $\mathfrak{U}$, as well as on query synthesis to create knowledge in completely under-explored niches of the input space X.

For the work reported in this paper, a query that might be forwarded to the oracle is determined by a particular classifier cl_Q, more precisely by its condition $cl_Q.C$. Thus, not only a single vector $\vec{x} \in X$ gets queried, but rather a certain subspace of the entire input space. The task of the oracle is to assign the correct label, or action, for the condition of cl_Q. If the oracle is not able to answer such a query with a predefined level of minimum confidence, the query is rejected and no answer is fed back to XCS.

In case of query rejection, it is assumed that the queried (macro-)classifier cl_Q negatively influences the overall learning process and is thus completely deleted from $[P]$. On the other hand, when an oracle answer is received, a new classifier cl^* is constructed with the action $cl^*.a$ set to the oracle answer, the condition $cl^*.C := cl_Q.C$, a payoff prediction value of $cl^*.p = \text{maxPayoff}$ (here 1000 for correct classifications), a fitness of $cl^*.F := 0.9$ (the minimum confidence level) and an experience set to the XCS parameter θ_{sub} to enable cl^* to immediately act as subsumer. All remaining attributes are initialized as usual. cl^* is then added to $[P]$ and $[M]$ before XCS continues with its action selection mechanism.

The Omniscient Oracle. The aforementioned assumptions about the oracle seem to be unrealistic and most often have to be relaxed when human experts are involved. Humans are not necessarily omnipresent and certainly not omniscient. Accordingly, the uncertainty of humans as well as the interaction load it is confronted with have to be considered. In our work, however, we do not restrict the oracle to be realized as a human annotator. Rather, we assume that in many real-world scenarios simulation models or heuristics exist that can be used alongside or instead of humans. For example, in the adaptive traffic light scenario reported in [22], a microscopic traffic simulation software was incorporated to optimize adequate reactions to so far unseen situations on demand. In the same scenario, a heuristic to approximate the average delay at urban intersections was available which serves as the fitness function for the utilized evolution strategy. Although such 'artificial oracles' might exist, the situation space is often far too complex to simulate all possible situations in advance. Hence, machine learning algorithms are used to generalize over the situation space and learn how to act correctly in similar (i.e. adjacent in X) situations. We deem the occurrence of KGs highly probable in exactly such large and dynamic real-world scenarios. Another advantage of an artificial oracle is that omnipresence is given, naturally limited by the computational resources. Nevertheless, either humans nor machines are omniscient, such that uncertainties must be presumed. On the other hand, human oracles might be more competent in answering difficult queries such as very specific and seldom cases. An artificial oracle that implements a certain heuristic could deliver inaccurate answers due to weaknesses of the underlying model. It clearly depends on criticality of the learning task and the according time constraints whether to take a fast reacting and non-fatiguing artificial oracle or rather a human oracle with highly accurate domain competence but reduced availability into the loop.

Figure 1 illustrates a generic two-layer architectural approach to make online machine learning algorithms proactive, here depicted for LCS. The bottom layer constitutes the conventional LCS architecture, where the population $[P]$ serves as the knowledge base $\hat{K}$. The LCS works as usual, i.e. it acts reactively to environmental stimuli. On the top layer, two components are illustrated: (1) A *Knowledge Gap Identifier*, which is responsible for seeking KGs in the algorithm's knowledge base $\hat{K}$ by means of uncertainty sampling or density estimation for instance, and (2) the *Knowledge Gap Closer*, which contains the oracle that can be realized by different means, e.g. by a human expert, a simulation or an *interpolation component* as introduced in [30].

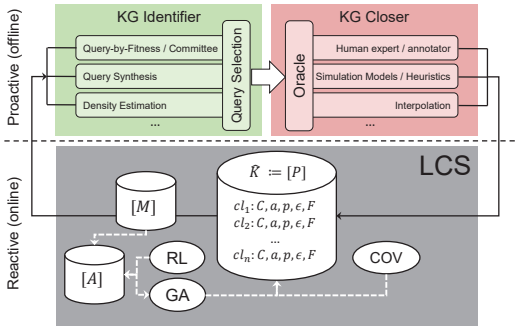


Figure 1: A two-layer pattern to implement a proactive LCS.

The next paragraphs each briefly render the original concept from the AL domain and subsequently describe our adaption to be used with the AL-extended XCS (ALXCS).

3.1 Curiosity in XCS

Uncertainty Sampling as introduced by Lewis et al. [14] is an AL technique that aims to query instances from $\mathfrak{U}$ for which the currently trained model is most uncertain regarding the correct output.

Least Confident (LC) uncertainty sampling is one straight-forward approach. The formula (cf. [25]) for LC is given by:

$$x^*_{LC} \coloneqq \underset{x}{\operatorname{argmax}} \left[1 - P_\theta(\hat{y}|x)\right], \text{ where } \hat{y} \coloneqq \underset{y}{\operatorname{argmax}} \left[P_\theta(y|x)\right]. \quad (1)$$

P_θ denotes the posterior probability of the current model (or hypothesis) θ. $\hat{y}$ is the label or class that maximizes the posterior probability for the currently considered unlabeled instance $x \in \mathfrak{U}$. Therefore, the formula yields an instance x^*_{LC} for which the model θ is most uncertain about.

It becomes apparent that with the above formula a posterior probability is needed. However, that is only the case when the given model is a probabilistic one, such as the *naïve bayes* classifier. Because XCS is not a probabilistic model but a space partitioning local approximator, we have to find another measure that resembles the posterior probability to judge on the uncertainty.

Intuitively, in this work we used the fitness estimate $cl.F$ to express confidence of a certain classifier about the subspace it is responsible for. If the fitness is low, the classifier's prediction quality is weak. However, this does not mean that it is incorrect at all. After generating a classifier its initial fitness is set to a low value, so that it has to prove itself during the next situations it matches and is selected for action execution. Accordingly, the prediction is weak for the moment but also uncertain since it is possible that it becomes accurate in the future. To be aware of this circumstance, we also incorporated a classifier's experience $cl.exp$. In this work, we realized the LC uncertainty sampling in the XCS environment as follows and refer to this method as *Query-by-Fitness* (QBF):

$$cl_Q \coloneqq \underset{cl \in [M]}{\operatorname{argmax}} \left[(1 - cl.F) \cdot cl.exp\right] \quad (2)$$

cl_Q is the selected classifier $cl \in [M]$ to be queried. The above formula makes sure to query a classifier that has a rather low fitness but has already gained a certain degree of experience. This ensures that no classifiers are chosen that were just created.

With this approach, we make XCS curious in the sense that it decides to query particular subspaces of X it is uncertain about, or it has difficulties to learn the correct action. The latter case might be due to *overgenerals* [12] which match in many situations where different actions would be correct. So, with the KG identification technique introduced above, XCS is enabled to identify such classifiers and accordingly take certain countermeasures. We could also identify just created classifiers by inverting $cl.exp$ in Equation 2. In this case, Equation 2 would yield a classifier from $[M]$ with the lowest fitness and at the same time the smallest experience. This method could be used to bootstrap the initial learning phase by asking the oracle for the correct action at an early stage of a classifier's life. The investigation of the latter method is left for future work.

3.2 The Committee of Matching Classifiers

Query-by-Committee (QBC) is a query selection strategy that uses a variety of independently trained models. The approach was first introduced by Seung et al. [26]. The collection of models serves as committee $\mathfrak{C} = \left\{\theta^{(1)}, \theta^{(2)}, \ldots, \theta^{(|\mathfrak{C}|)}\right\}$. The idea behind QBC is to analyze the models' outputs to identify conflicts. The conflict's extent is determined by a measure of disagreement between the different models. One approach is to use a so-called *vote entropy* as reported in [25] which essentially counts the number of times a committee member votes for a certain label, derives a probability for each possible label and subsequently calculates the *Entropy* [27].

To use QBC within the context of XCS one could decide to train several XCSs at the same time. However, there is a less costly approach in terms of computational power which interprets a classifier as an entire model for the current environmental niche determined by the match set $[M]$. Let $A_{[M]}$ define the set of distinct actions in $[M]$. We calculate the vote entropy $H_{[M]}$ as given by:

$$H_{[M]} = \sum_{a \in A_{[M]}} V(a) \log_2 V(a) \quad (3)$$

with the actual vote for each action a calculated by

$$V(a) = \frac{\sum_{cl \in [M]^{(a)}} cl.num}{\sum_{cl \in [M]} cl.num}, \quad (4)$$

and the subset of $[M]$ containing all classifiers with the currently considered action a defined by $[M]^{(a)} : \{cl \in [M] | cl.a == a\}$.

After determining $H_{[M]}$, we also calculate the maximum entropy $H_{max} = \log_2 |A_{[M]}|$ that can be interpreted as the situation of a fully uniform vote distribution, in the simplest case when each committee member advocates one of the distinct actions. The closer $H_{[M]}$ is to H_{max}, the higher is the decision conflict about which action is most suitable. Based on $H_{[M]}$ and H_{max} we decide to send a query to the oracle only if $H_{[M]} \geq H_{max} \cdot (1 - \theta_{sim})$. Thereby, θ_{sim} defines a tolerance factor to control the extent of the conflict necessary to ask the oracle. If the decision is felt to send a query to the oracle, the query itself is then selected by means of QBF. Thus, QBC restricts the execution of QBF to match sets where the classifiers are not certain about the most appropriate action, at all. This, in the end, is one approach to reduce the interaction load between the learning algorithm and the oracle.

3.3 Knowledge Generation by Query Synthesis

Membership Query Synthesis [1] is a special case of query selection where the unlabeled instance x that is forwarded to the oracle is generated *de novo*, i.e. synthesized. There are several approaches for synthesizing queries. Probably the simplest form is to generate an instance x completely at random by choosing an arbitrary vector from the input space X. More sophisticated approaches are imaginable, such as iteratively estimating the distribution of the underlying data generating process by means of density estimation or histogram construction. Our current efforts are targeted at such forms of query synthesis, however, in this paper, we restrict our focus to randomized query generation.

Accordingly, we propose the concept of *Query-Synthesis-at-Random* (QSR). Here, we create a completely new classifier cl^* that also serves as the classifier to be queried cl_Q. A condition is created by selecting a random vector $\vec{x} \in X \subseteq \mathbb{R}^n$ from the input space and subsequently create an interval predicate within predefined bounds. For determining these bounds, it is imaginable to use the default spread parameter r_0 [39], so that each interval predicate

is restricted to $[0, r_0]$. However, we decided to use different spreading bounds s_{min} and s_{max} for the AL process that guarantee an interval predicate to lie within $x_i \pm s \in [s_{min}, s_{max}], \forall i = 1 \ldots n$.

Generating queries de novo is known to not be suitable in any kind of learning problem. For example, in handwritten character classification, completely undefined symbols can be produced [25]. For well defined problem spaces such as the toy problem we present in Section 4.2, QSR is assumed to be beneficial, whereas in very domain specific problem types such as medical diagnosis, randomly created instances are likely to confuse a human oracle since the attribute combination might be completely arbitrary.

4 PRELIMINARY EXPERIMENTS

The following paragraphs report on the results of so far conducted preliminary experiments. We evaluated our approach on two different classification scenarios – the well-known *Wisconsin Breast Cancer* (WBC) dataset [16] and a novel toy problem. Since XCS is conventionally a reinforcement learning system, but here applied to a single-step supervised classification task, we set the immediate reward to 1000 for correct classifications and to 0 for incorrect ones.

4.1 Wisconsin Breast Cancer Data Set

The WBC data set [16] is available from the UCI Machine Learning Repository [15] and can be found in the category of life sciences. It consists of 699 instances each having 9 attributes which show the results of a fine needle aspiration for the diagnosis of breast cancer. The class labels are '2' and '4' signaling a benign or malignant tumor, respectively. The attributes are encoded by integers in the range $[1, 10]$. Without loss of generality, we normalized the attributes to the range $[0, 1]$. Missing attribute values are marked as '?' and assumed to be matched in any case. This dataset bears a class imbalance ratio of $0.655 : 0.345$ for the benign and the malignant class, respectively. We decided to use this data set since it already has been investigated in the LCS literature before [11, 39].

The oracle was simulated by a rather simple heuristic. The heuristic is realized by a simple nearest neighbor search through the available instances. Therefore, the Euclidean distance between the center point of a queried classifier's condition $cl_Q.C$ and the data instance from the dataset was calculated. The label of the instance with the minimal distance was returned as oracle answer. This approach implicitly bears a certain degree of oracle uncertainty, since the nearest neighbor is not guaranteed to yield the correct classification. We transformed the distance to a similarity measure defined between $[0, 1]$ by normalizing the Euclidean distance and subtracting it from 1. Whenever this similarity measure is < 0.9 no answer is provided by the oracle, i.e. the query is rejected. For the WBC experiments, we configured XCS according to [39]: $N = 6400$, $\alpha = 0.1$, $\beta = 0.2$, $\delta = 0.1$, $\nu = 5$, $\theta_{GA} = 48$, $\epsilon_0 = 1$, $\theta_{mna} = 2$, $\theta_{del} = 50$, $\theta_{sub} = 50$, $\chi = 0.8$, $\mu = 0.04$, $p_{ini} = 10.0$, $\epsilon_{ini} = 0.0$, $F_{ini} = 0.01$, $r_0 = 0.4$, $m_0 = 0.2$. For a detailed description of the standard parameters the reader is referred to [6, 38, 39]. We used the *unordered bound representation* [31] for encoding the conditions.

4.2 Toy Classification Problem: Mario Pixel Art

Our novel Mario environment consists of a 16x16 pixel art representing the famous video game character Super Mario [20]. The pixel art is formed using seven different colors. The goal for the XCS is to learn the correct color for each possible position $\vec{x} \in \mathbb{R}^2$ in the pixel art. This toy problem resembles the well-known checkerboard problem which was used as an XCS testbed e.g. in [30, 31]. In contrast to the checkerboard, it allows for more general classifiers, i.e. the blue trousers of mario, but at the same time entails a more complex action space A of size $|A| = 7$ (i.e. the colors of Mario, see colored version of Fig. 2). The real-valued input space $X \subseteq [0, 1]^2$ is large and infinite.

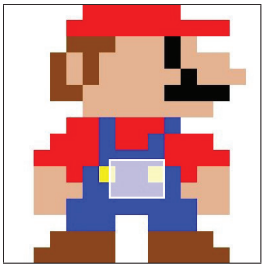


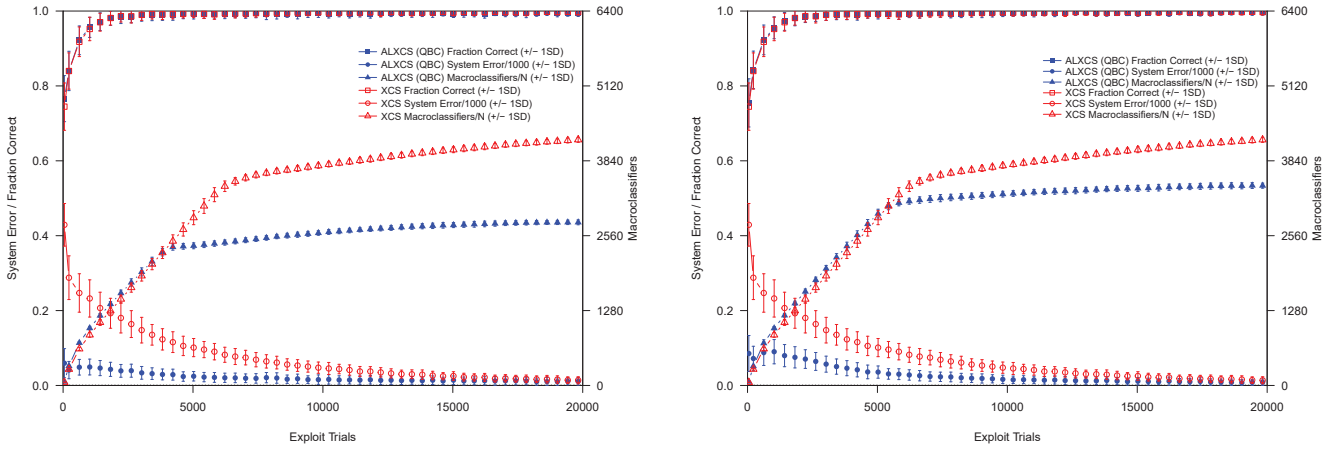
Figure 2: The Mario Toy Classification Problem: A 16x16 Pixel Art grid showing Super Mario [20]. The inner rectangle shows a queried classifier's condition that is expected to be answered by the oracle.

The oracle for this environment was realized as follows: We determined the fraction of all possible colors (i.e. actions) over all pixels that are encompassed by the queried classifier's condition (see the rectangle on Mario's trousers on Fig. 2 for an example). Whenever no color exceeds the certainty threshold of 0.9, the query is rejected by the oracle. Otherwise, the color with a fraction ≥ 0.9 is returned as answer. XCS was configured as follows: $N = 7000$, $\alpha = 0.1$, $\beta = 0.3$, $\delta = 0.1$, $\nu = 5$, $\theta_{GA} = 30$, $\epsilon_0 = 10$, $\theta_{mna} = 6$, $\theta_{del} = 50$, $\theta_{sub} = 50$, $\chi = 0.8$, $\mu = 0.04$, $p_{ini} = 10.0$, $\epsilon_{ini} = 0.0$, $F_{ini} = 0.01$, $r_0 = 0.1$, $m_0 = 0.1$. Again, the *unordered bound representation* [31] was used.

4.3 Experimental Setup

All experiments were run over 100.000 alternating explore/exploit trials each and repeated for 30 i.i.d. runs. Statistical significance on the differences between ALXCS and XCS was determined using pairwise t-tests. We used a *test-then-train* learning strategy for both scenarios to keep the online learning nature of XCS. Instances from the WBC dataset are selected uniformly at random and with replacement. For the synthetic and continuous toy problem, the input space was sampled using a uniform distribution. Thus, for both scenarios, KGs and accordingly the potential for improvements are expected at the beginning of the learning task. For the QBF and the QSR technique, the oracle is asked every step. Using the QBC approach, the number of oracle interactions is self-adapting.

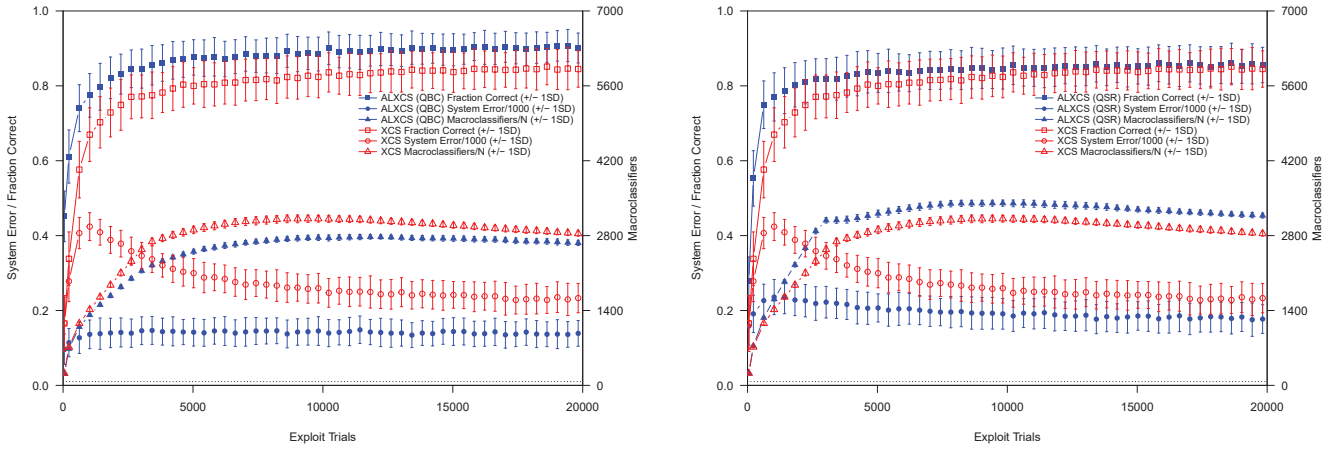
For these preliminary experiments, we simulated the oracles by means of simple heuristics as described in the respective sections. Additionally, we conducted an experiment with a real human oracle for the Mario problem over 1000 explore/exploit trials each which was repeated for ten times.



(a) QBC with $\theta_{sim} = 0.20$ on the first 20k trials

(b) QBC with $\theta_{sim} = 0.01$ on the first 20k trials

Figure 3: ALXCS (blue lines) with QBC and two different values for the similarity tolerance θ_{sim} vs. XCS (red lines[3]) on WBC



(a) QBC with $\theta_{sim} = 0.20$ on the first 20k trials

(b) QSR on the first 20k trials

Figure 4: ALXCS (blue lines) with QBC (left) and QSR (right) vs. XCS (red lines) on the Mario toy problem

Table 1: Summary of results on WBC dataset. Average values and standard deviations over two phases of the entire learning task (until 20k trials on the left and 100k exploit trials on the right) from 30 i.i.d runs are shown. * (**) indicates statistically (highly) significant deviations of the reported metrics compared to XCS, i.e. that for the p-values of paired one-sided t-tests hold $p < \alpha = 0.05$ (0.01). ↑ and ↓ indicate whether the value has increased or decreased, respectively, in comparison to XCS.

WBC Data Set / **Artificial Oracle**	*Fraction Correct* mean ±1SD	*System Error* mean ±1SD	*Macro-Classifiers* mean ±1SD	*Fraction Correct* mean ±1SD	*System Error* mean ±1SD	*Macro-Classifiers* mean ±1SD
Exploit Trials	First 20.000			Entire 100.000		
ALXCS w/ **QBF**	$987.00^{\downarrow **} \pm 2.03$	$20.01^{\downarrow **} \pm 2.58$	$2285.78^{\downarrow **} \pm 32.48$	$995.32^{\downarrow **} \pm 1.28$	$9.79^{\downarrow **} \pm 1.76$	$2345.03^{\downarrow **} \pm 28.81$
ALXCS w/ **QBC**, $\theta_{sim} = 0.01$	$988.38^{\downarrow} \pm 1.72$	$27.37^{\downarrow **} \pm 2.26$	$2906.76^{\downarrow **} \pm 33.90$	$996.61^{\downarrow **} \pm 0.85$	$8.68^{\downarrow **} \pm 1.41$	$2964.57^{\downarrow **} \pm 32.59$
ALXCS w/ **QBC**, $\theta_{sim} = 0.20$	$987.16^{\downarrow **} \pm 1.98$	$20.58^{\downarrow **} \pm 2.43$	$2396.39^{\downarrow **} \pm 28.30$	$995.72^{\downarrow **} \pm 1.01$	$9.28^{\downarrow **} \pm 1.55$	$2417.17^{\downarrow **} \pm 31.07$
ALXCS w/ **QSR**	$985.22^{\downarrow **} \pm 1.60$	$87.36^{\uparrow **} \pm 3.44$	$5392.15^{\uparrow **} \pm 14.39$	$988.70^{\downarrow **} \pm 0.76$	$71.38^{\uparrow **} \pm 3.15$	$5902.69^{\uparrow **} \pm 9.14$
Standard XCS	988.78 ± 1.48	74.30 ± 2.80	3287.55 ± 38.99	997.20 ± 0.55	17.70 ± 0.94	3749.73 ± 32.48

Table 2: Summary of results on the novel Mario environment. Average values and standard deviations over two phases of the entire learning task (until 20K trials on the left and 100K exploit trials on the right) from 30 i.i.d runs are shown. Asterisks * (**) and arrows (↑ and ↓) are to be interpreted as for Table 1.

Mario Pixel Art Artificial Oracle	*Fraction Correct* mean ±1SD	*System Error* mean ±1SD	*Macro-Classifiers* mean ±1SD	*Fraction Correct* mean ±1SD	*System Error* mean ±1SD	*Macro-Classifiers* mean ±1SD
Exploit Trials	First 20.000			Entire 100.000		
ALXCS w/ **QBF**	$872.78^{\uparrow **} \pm 4.43$	$139.57^{\downarrow **} \pm 3.71$	$2480.61^{\downarrow **} \pm 39.92$	$889.40^{\uparrow **} \pm 2.92$	$140.50^{\downarrow **} \pm 2.98$	$2307.29^{\downarrow **} \pm 33.03$
ALXCS w/ **QBC**, $\theta_{sim} = 0.01$	$819.32^{\uparrow **} \pm 6.38$	$238.41^{\downarrow **} \pm 6.09$	$2788.34^{\downarrow} \pm 36.72$	$870.25^{\uparrow **} \pm 4.71$	$188.29^{\downarrow **} \pm 4.27$	$2296.43^{\downarrow **} \pm 26.02$
ALXCS w/ **QBC**, $\theta_{sim} = 0.20$	$873.07^{\uparrow **} \pm 4.59$	$140.37^{\downarrow **} \pm 4.31$	$2493.43^{\downarrow **} \pm 26.81$	$898.99^{\uparrow **} \pm 3.12$	$140.59^{\downarrow **} \pm 3.13$	$2313.82^{\downarrow **} \pm 27.44$
ALXCS w/ **QSR**	$834.02^{\uparrow **} \pm 4.89$	$195.86^{\downarrow **} \pm 4.79$	$3094.81^{\uparrow **} \pm 37.62$	$869.96^{\uparrow **} \pm 3.82$	$163.09^{\downarrow **} \pm 3.24$	$2940.72^{\uparrow **} \pm 17.65$
Standard XCS	801.00 ± 6.87	276.23 ± 6.46	2784.98 ± 41.92	866.23 ± 5.22	201.14 ± 5.84	2385.95 ± 20.60

Table 3: Summary of results on the novel Mario environment with a human oracle. Average values and standard deviations over the entire learning task of 1000 explore/exploit trials from the 10 i.i.d. runs are shown. Asterisks * (**) and arrows (↑ and ↓) are to be interpreted as for Table 1.

Mario Pixel Art Human Oracle	*Fraction Correct* mean ±1SD	*System Error* mean ±1SD	*Macro-Classifiers* mean ±1SD
ALXCS w/ **QBF**	$573.80^{\uparrow **} \pm 18.35$	$279.44^{\downarrow **} \pm 17.07$	$1022.11^{\downarrow} \pm 24.38$
ALXCS w/ **QBC**, $\theta_{sim} = 0.01$	$622.90^{\uparrow **} \pm 26.48$	$251.06^{\downarrow **} \pm 14.54$	$1036.67^{\uparrow} \pm 14.71$
ALXCS w/ **QSR**	$565.30^{\uparrow **} \pm 29.39$	$297.49^{\downarrow **} \pm 14.06$	$1019.90^{\downarrow} \pm 25.12$
Standard XCS	503.10 ± 20.16	358.73 ± 7.91	1024.05 ± 24.37

4.4 Results

All results we observed after conducting the experiments are summarized in the Tables 1-3. Examples of plots depicting the learning progress for both scenarios are given by Figures 3 and 4.[3] As assumed, the main benefit of the presented AL techniques appears at the beginning of the investigated learning tasks. QBF as well as the related QBC technique can noticeably reduce the number of needed macroclassifiers to obtain a strongly decreased system error. This holds true even when we look at the entire learning tasks over 100k exploit trials for both scenarios. The fraction of correct classifications, however, is marginally reduced throughout all WBC experiments when QBF and QBC is incorporated.

For the WBC setting, the QSR technique yields negative impacts on all three figures of merit. This underpins the known disadvantage of query synthesis in general we discussed in Section 3. The strongly increased average population size let us suppose that the randomized queries (i.e. conditions) are too arbitrary to yield a performance gain. The population seems to be 'swamped' with random classifiers which, due to the highly set initial values for fitness and predicted reward, prevents fitness pressure. This resembles the effect of a *cover-delete-cycle* [4]. A more deeper investigation on the impacts on classifier evolution is subject of current work.

For the Mario environment, however, we obtained positive impacts of the QSR technique. Since the problem domain is sampled uniformly it can be expected that any region of X is visited equally likely. Thus, a proactive knowledge generation is hypothesized to be useful. The results reported in Table 2 and shown in Figure 4b seem to underpin this hypothesis. The average values for the fraction of correct classifications as well as for the system error can be clearly improved but at the expense of an increased average population size. Although the discussed results on QSR seem to be like a double-edged sword regarding the two scenarios, we still deem proactive classifier generation as a meaningful approach to overcome challenges such as drifting input distributions and highly imbalanced data streams. We currently work on more sophisticated methods to decide in which region of the knowledge base synthesized classifiers would be helpful. We reported on two different values for the similarity tolerance θ_{sim} that controls the committee voting conflict necessary to interact with the oracle. Obviously, the higher value 0.2 yields better results since the oracle is queried more frequently. We set this value to a reasonable small value 0.01 to provide a first hint on possible improvements when the number of oracle interactions is drastically reduced. We could retain the positive impacts on all three figures of merit even if the number of oracle queries is reduced by $\approx 76\%$ for the WBC setting, and $\approx 92\%$ for the Mario environment. This is a very promising result since we were able to show that even with a small fraction of the learning steps we can obtain significantly performance improvements.

To consider the aspect of human uncertainty, we carried out another experiment with a human oracle. The results are presented in Table 3. We observed improved performance throughout all configurations for all three metrics. Interestingly, the best configuration was QBC with a similarity tolerance $\theta_{sim} = 0.01$, which resulted in a low interaction load for the human oracle.

5 CONCLUSION

The overall aim of this work was to present the vision of proactive learning classifier systems. We showed how to endow XCS with some form of curiosity that results in the desire to learn more about uncertain regions of the already evolved knowledge base. Furthermore, we demonstrated that proactive knowledge generation, i.e. building classifiers in regions of the input space before they are

[3] Please note that in Figure 3 the developments regarding the fractions of correct classifications of both XCS variants are nearly congruent what leads to a hidden (red) curve for standard XCS.

actually needed, could have beneficial effects particularly at the beginning of a learning task, even if these regions are selected completely at random so far. More sophisticated approaches are subject of current investigations. Although we showed promising results, the experiments are preliminary and, thus, limited in at least two aspects: (1) Non-drifting uniform sampling of the problem's input domain, which let us presume that KGs mainly existed at the beginning of the respective learning tasks. (2) At least for the reported QBF and QSR query strategies, the AL component was activated in each step, what results in a noticeable interaction load with the oracle. The latter aspect mainly constitutes a problem when a human oracle is taken in the learning loop. However, as a first countermeasure, we utilized the presented QBC approach, which resulted in a significant reduction of the oracle load.

Future Work. The promising preliminary results we achieved so far drive our efforts toward a more thorough investigation of the presented techniques with a specific attention on: (1) the number of needed oracle interactions to achieve significant improvements, (2) the degree of oracle confidence needed, or rather uncertainty allowed to gain beneficial effects, and (3) the impacts on classifier evolution in terms of e.g. generality, experience and mean lifetime.

We discussed the role of the oracle in Section 3, where we stated that not only humans can be taken in the loop. The combination of proactive knowledge generation outlined in this paper with the interpolation component reported in previous work [29, 30] used for the purpose of closing identified knowledge gaps, constitutes a further top priority on our research agenda.

So far, we applied the ALXCS to classification tasks only. In a next step, we plan to transfer our techniques to support the function approximation capability of XCSF – an XCS derivative for regression tasks [40]. In general, all techniques presented in this paper are easily transferable to other LCS derivatives such as UCS [2] or ExSTraCS [33]. Another branch of investigation is planned to target the use of ALXCS within delayed reward reinforcement learning tasks as presented in [17] or on more complex scenarios such as coverage optimization in self-configuring smart camera networks [24].

We defined the term of knowledge gaps at the beginning of this paper. It was stated that one reason that supports KGs to arise is an empty population at the beginning of a learning phase. On the other hand, KGs might occur due to non-uniform sampling of the input space X or due to abrupt problem space changes regarding the correct classification, i.e. the complexity of the decision boundary. We will investigate the impacts of both non-uniform (and drifting) sample distributions and the presence of small disjuncts or else complex decision boundaries on the learning capability of XCS in general and with a certain focus on the improvement potential that might be achieved by means of proactive knowledge generation driven by curiosity in LCS. Therefore, we plan to conduct more experiments on further realistic data sets as well as on synthetic environments that provoke the aforementioned issues.

REFERENCES

[1] Dana Angluin. 1988. Queries and Concept Learning. *Machine Learning* 2, 4 (1988), 319–342.

[2] E. Bernad-Mansilla and J. M. Garrell-Guiu. 2003. Accuracy-Based Learning Classifier Systems: Models, Analysis and Applications to Classification Tasks. *Evol. Comput.* 11, 3 (2003), 209–238.

[3] L. Bull, J. Sha'Aban, A. Tomlinson, J.D. Addison, and B.G. Heydecker. 2004. Towards Distributed Adaptive Control for Road Traffic Junction Signals using Learning Classifier Systems. In *Applications of Learning Classifier Systems*. Studies in Fuzziness and Soft Computing, Vol. 150. Springer, 276–299.

[4] M.V. Butz, T. Kovacs, P.L. Lanzi, and S.W. Wilson. 2004. Toward a Theory of Generalization and Learning in XCS. *Evol. Comp., IEEE Trans.* 8, 1 (2004), 28–46.

[5] MartinV. Butz and Olivier Sigaud. 2012. XCSF with local deletion: preventing detrimental forgetting. *Evolutionary Intelligence* 5, 2 (2012), 117–127.

[6] Martin Butz and Stewart W. Wilson. 2002. An Algorithmic Description of XCS. *Soft Comput.* 6, 3-4 (2002), 144–153.

[7] David Cohn, Les Atlas, and Richard Ladner. 1994. Improving generalization with active learning. *Machine Learning* 15, 2 (1994), 201–221.

[8] David E. Goldberg. 1987. Computer-aided pipeline operation using genetic algorithms and rule learning. PART II: Rule learning control of a pipeline under normal and abnormal conditions. *Engineering with Computers* 3, 1 (1987), 47–58.

[9] H. He and E. A. Garcia. 2009. Learning from Imbalanced Data. *IEEE Transactions on Knowledge and Data Engineering* 21, 9 (Sept 2009), 1263–1284.

[10] John H. Holland. 1976. Adaptation. In *Progress in Theoretical Biology*, Robert Rosen and F.M. Snell (Eds.). Vol. 4. Academic Press, New York, 263–293.

[11] Faten Kharbat, Mohammed Odeh, and Larry Bull. 2008. Knowledge Discovery from Medical Data: An Empirical Study with XCS. In *Learning Classifier Systems in Data Mining*. Studies in Computational Intelligence, Vol. 125. Springer, 93–121.

[12] Tim Kovacs. 2000. *Strength or Accuracy? Fitness Calculation in Learning Classifier Systems*. Springer Berlin Heidelberg, Berlin, Heidelberg, 143–160.

[13] J. Lehman and K. Stanley. 2008. Exploiting open-endedness to solve problems through the search for novelty. In *Artificial Life XI: Proceedings of the Eleventh International Conference on the Simulation and Synthesis of Living Systems*. MIT Press, Cambridge, MA, 329–336.

[14] David D. Lewis and William A. Gale. 1994. A Sequential Algorithm for Training Text Classifiers. In *Proc. of 17th Annual International ACM-SIGIR Conference on Research and Development in Information Retrieval*. Dublin, Ireland, 3–12.

[15] M. Lichman. 2013. UCI Machine Learning Repository. (2013). http://archive.ics.uci.edu/ml

[16] Olvi L. Mangasarian, W. Nick Street, and William H. Wolberg. 1995. Breast Cancer Diagnosis and Prognosis Via Linear Programming. *Operations Research* 43, 4 (1995), 570–577.

[17] Anis Najar, Olivier Sigaud, and Mohamed Chetouani. 2015. *Social-Task Learning for HRI*. Springer International Publishing, Cham, 472–481.

[18] Anis Najar, Olivier Sigaud, and Mohamed Chetouani. 2015. Socially Guided XCS: Using Teaching Signals to Boost Learning. In *Proceedings of the Companion Publication of GECCO'15 (GECCO Companion '15)*. ACM, NY, USA, 1021–1028.

[19] M. Nakata, P. L. Lanzi, T. Kovacs, W. N. Browne, and K. Takadama. 2015. How should learning classifier systems cover a state-action space?. In *2015 IEEE Congress on Evolutionary Computation (CEC)*. 3012–3019.

[20] Nintendo. 1985. Super Mario Bros. Video Game for the Nintendo Entertainment System, Japan. (1985).

[21] Albert Orriols-Puig and Ester Bernadó-Mansilla. 2008. Evolutionary rule-based systems for imbalanced data sets. *Soft Computing* 13, 3 (2008), 213.

[22] H. Prothmann, F. Rochner, S. Tomforde, J. Branke, C. Müller-Schloer, and H. Schmeck. 2008. Organic Control of Traffic Lights. In *Proc. of the 5th Int. Conf. on Autonomic and Trusted Computing*. Springer, Berlin, Heidelberg, 219–233.

[23] Tobias Reitmaier, Adrian Calma, and Bernhard Sick. 2015. Transductive active learning-A new semi-supervised learning approach based on iteratively refined generative models to capture structure in data. *Inf. Sci.* 293 (2015), 275–298.

[24] Stefan Rudolph, Sarah Edenhofer, Sven Tomforde, and Jörg Hähner. 2014. Reinforcement Learning for Coverage Optimization Through PTZ Camera Alignment in Highly Dynamic Environments. In *Proceedings of the International Conference on Distributed Smart Cameras (ICDSC '14)*. ACM, NY, USA, Article 19.

[25] B. Settles. 2009. *Active Learning Literature Survey*. Technical Report 1648. University of Wisconsin-Madison.

[26] H. S. Seung, M. Opper, and H. Sompolinsky. 1992. Query by Committee. In *Proc. of the 5th Annual Workshop on Computational Learning Theory (COLT '92)*. ACM, NY, USA, 287–294.

[27] C. E. Shannon. 1948. A Mathematical Theory of Communication. *Bell System Technical Journal* 27, 3 (1948), 379–423.

[28] Patrick O. Stalph and Martin V. Butz. 2012. Learning local linear Jacobians for flexible and adaptive robot arm control. *Genet Program Evolvable Mach* 13, 2 (2012), 137–157.

[29] A. Stein, C. Eymüller, D. Rauh, S. Tomforde, and J. Hähner. 2016. Interpolation-based Classifier Generation in XCSF. In *2016 IEEE Congress on Evolutionary Computation (CEC)*. 3990–3998.

[30] A. Stein, D. Rauh, S. Tomforde, and J. Hähner. 2017. Interpolation in the eXtended Classifier System: An Architectural Perspective. *Journal of Systems Architecture* (2017). DOI: http://dx.doi.org/10.1016/j.sysarc.2017.01.010

[31] Christopher Stone and Larry Bull. 2003. For Real! XCS with Continuous-Valued Inputs. *Evol. Comput.* 11, 3 (2003), 298–336.

[32] Ryan J. Urbanowicz, Delaney Granizo-Mackenzie, and Jason H. Moore. 2012. *Using Expert Knowledge to Guide Covering and Mutation in a Michigan Style Learning Classifier System to Detect Epistasis and Heterogeneity*. Springer Berlin Heidelberg, 266–275.

[33] Ryan J. Urbanowicz and Jason H. Moore. 2015. ExSTraCS 2.0: description and evaluation of a scalable learning classifier system. *Evolutionary Intelligence* 8, 2 (2015), 89–116.

[34] Geoffrey I. Webb, Roy Hyde, Hong Cao, Hai Long Nguyen, and Francois Petitjean. 2016. Characterizing concept drift. *Data Min. Knowl. Discov.* (2016), 1–31.

[35] Gary M. Weiss. 2004. Mining with Rarity: A Unifying Framework. *SIGKDD Explor. Newsl.* 6, 1 (June 2004), 7–19.

[36] Stewart W. Wilson. 1994. ZCS: A Zeroth Level Classifier System. *Evol. Comput.* 2, 1 (1994), 1–18.

[37] Stewart W. Wilson. 1995. Classifier Fitness Based on Accuracy. *Evol. Comput.* 3, 2 (1995), 149–175.

[38] Stewart W. Wilson. 2000. Get Real! XCS with Continuous-Valued Inputs. In *Learning Classifier Systems*. Lecture Notes in Computer Science, Vol. 1813. Springer Berlin Heidelberg, 209–219.

[39] Stewart W. Wilson. 2001. Mining Oblique Data with XCS. In *Advances in Learning Classifier Systems*. Lecture Notes in Computer Science, Vol. 1996. Springer Berlin Heidelberg, 158–174.

[40] Stewart W. Wilson. 2002. Classifiers that Approximate Functions. *Natural Computing* 1, 2-3 (2002), 211–234.

[41] Qiong Wu and Chunyan Miao. 2013. Curiosity: From Psychology to Computation. *ACM Comput. Surv.* 46, 2, Article 18 (Dec. 2013), 26 pages.