Continuous Endpoint Data Mining with ExSTraCS: A Supervised Learning Classifier System

Ryan J. Urbanowicz
University of Pennsylvania
423 Guardian Dr.
Philadelphia, PA
19104-6021,USA
ryanurb@upenn.edu

Niranjan Ramanand
Dartmouth College
1 Medical Center Dr.
Lebanon, NH 03755,USA
niranjan.ramanand.17
@dartmouth.edu

Jason H. Moore
University of Pennsylvania
423 Guardian Dr.
Philadelphia, PA
19104-6021,USA
jhmoore@upenn.edu

ABSTRACT

ExSTraCS is a powerful Michigan-style learning classifier system (LCS) that was developed for classification, prediction, modeling, and knowledge discovery in complex and/or heterogeneous supervised learning problems with clean or noisy signals. To date, ExSTraCS has been limited to problems with discrete endpoints (i.e. classes). Many real world problems, however, involve endpoints with continuous values (e.g. function approximation, or quantitative trait analyses). In some problems the goal is to predict a specific continuous value with low error based on input values. In other problems it may be more informative to predict continuous intervals (i.e. predict that an endpoint falls within some range to define meaningful thresholds within the endpoint continuum). Thus far, there has not been a supervised learning LCS designed to handle continuous endpoints, nor one that incorporates interval predictions within rules. In this paper, we propose and evaluate (1) a supervised learning approach for solving continuous endpoint problems that connects input states to endpoint intervals within rules, (2) a novel prediction scheme that converts interval predictions into a specific continuous value prediction, and (3) an alternate approach to rule subsumption. Following simulation study analyses, we discuss the benefits and drawbacks of these implementations within ExSTraCS.

Categories and Subject Descriptors

J.3 [**Computer Applications**]:
Life and Medical Sciences[biology and genetics]

Keywords

classifier systems, bioinformatics, rule-based machine learning, quantitative trait analysis

GECCO '15, July 11 - 15, 2015, Madrid, Spain

DOI: http://dx.doi.org/10.1145/2739482.2768453

1. INTRODUCTION

Learning classifier systems (LCS) are rule-based machine learning strategies. Their approach to data mining, endpoint/action prediction, and classification tasks are flexible and powerful partially because they do not make assumptions about the underlying pattern(s) of association in a given environment or training dataset [17]. This versatility makes them particularly well suited to complex, heterogeneous, and distributed problem domains. While LCS algorithms are traditionally and still widely designed for reinforcement learning problems (where the correct prediction for each training instance is unknown, and rewards are typically delayed), they have increasingly been developed for exclusive application to supervised learning problems (where the correct endpoint value is available for each training instance). In this paper we use the term 'endpoint' to refer generally to the dependent variable, sometimes referred to as the 'class', 'action', 'output', 'trait', or 'phenotype' given the relevant context. While reinforcement learning LCS algorithms such as the well-known XCS [20] can also be applied to supervised learning tasks, they are not optimized for this type of learning. LCS algorithms such as UCS [2], XCSCA [7], BIOHel [1], and ExSTraCS [14, 18] perform best-action mapping and apply exclusively to supervised learning problems that typically possess a finite training set.

ExSTraCS is an Extended Supervised Tracking and Classifying System, specifically designed to address single-step supervised learning problems [14]. ExSTraCS is a platform for ongoing Michigan-style LCS development, designed with bioinformatics applications in mind. These problems are noisy, often with a large number of potentially predictive attributes (discrete or continuous-valued), and may involve complex patterns of association including epistasis (i.e. nonlinear attribute interaction effects) and heterogeneity (i.e. independent associations with the same or similar endpoint). ExSTraCS 2.0 introduced strategies to dramatically improve the scalability of Michigan-style LCS's [18]. Furthermore, ExSTraCS 2.0 was the first machine learning algorithm reported to solve the extremely complex 135-bit multiplexer benchmark problem directly.

To date, ExSTraCS implementations have only been designed to accommodate discrete endpoints (i.e. binary class or multi-class endpoints). In an effort to further expand the applicability of ExSTraCS, the next goal is to extend the algorithm to accommodate datasets with continuous-valued (i.e. real-valued) endpoints, also referred to as continuous action problems in reinforcement learning, or quantitative

trait analysis in bioinformatics and epidemiology. This is a critical target for development since many data mining tasks in or outside of bioinformatics involve continuous endpoints. For example, epidemiological research in the field of neurospychiatric disease involves the analysis of continuous dependent variables such as MRI-estimated brain section volumes, or scores on different tests of learning or memory. Also consider that when learning a problem with a continuous endpoint, two scenarios can exist that may impact learning: (1) it is more informative for a rule to predict a specific continuous value with low error (as is the case with function approximation [22]), or (2) it is more informative for a rule to predict a finite interval of continuous endpoint values as a way to identify meaningful thresholds (i.e. underlying unknown discretizations of continuous values within the observed range of endpoint values.

Within the literature, a number of strategies have been developed to adapt LCS to problems with a continuous endpoint. LCSs using fuzzy logic were the first to produce real-valued outputs [19, 3]. Fuzzy systems can be applied either as controllers requiring continuous output assignment, or used to determine the degree of membership among a set of discrete classes [8]. The XCS algorithm was later adapted by Wilson to yield XCSF, which utilized reward predictions (for reinforcement learning) that were 'calculated' rather than 'fixed scalars' in order to perform function approximation [22, 23]. This strategy combined state values with an adapting weight vector to compute piece-wise linear approximations of a function. A single 'dummy' action was used for function approximation such that the meaningful output came from the reward prediction. Wilson later discussed three methods for continuous action learning designed for reinforcement learning: Interpolating Action Listener (IAL), Continuous Actor-Critic (CAC), and the General Classifier System (GCS) [25]. Both IAL and CAC operate by running two LCS algorithms in tandem, with one observing and learning from the other in order to achieve a continuous-valued action output. Differently, GCS involves a single LCS that includes the action as part of the rule condition. Similar to state conditions, this action-condition can be represented by either a simple interval predicate (which would yield a hyper-cube in problems with higher dimensions) or more efficiently by parameters defining an ellipse (yielding a hyperellipsoid at higher dimensions) as proposed by Butz [4]. Wilson's later related work dealt with continuous payoff functions for discrete actions in reinforcement learning [24]. Tran *et. al.* expanded on the XCSF concept with the introduction of computed continuous actions in XCSFCA[10]. This approach preserves the XCSF algorithm but adds an action weight vector from which to compute real-valued actions, and an evolutionary strategy that relies on a similarly evolving standard deviation vector to evolve the actions weights. More recently Iqbal *et. al.* developed XCSRCFA and XCSCFA which took XCSR [21] and XCS [20] respectively, and replaced discrete actions in rules with code fragments [5, 6]. Code fragments are tree-expressions similar to trees generated in genetic programming. In follow-up work Iqbal *et. al.* compared their original XCSCFA to a similar version that computed actions based on the state values from the environment rather than from the states included in the rule condition, noting key advantages of each.

In the present study, our main goal is to develop a continuous endpoint strategy for LCS that will function within the ExSTraCS supervised learning specific framework. As a data mining tool, it is just as important to preserve rule interpretability and flexibility to handle different unknown patterns of association as it is to achieve high prediction accuracy. Unlike existing continuous endpoint LCS implementations that exist, ExSTraCS is a purely supervised learning method. It does not include rule estimation of payoff or reward assignment, and it replaces the action set [A] with a correct set [C], like its predecessor UCS [2]. Given that a correct set is impossible to define in the context of other continuous endpoint implementations, we wanted to see if we could design a strategy for continuous endpoint prediction that preserved [C]. Additionally, we want to develop an approach flexible enough to predict endpoint values with low error (for applications like function approximation), as well as uniquely capture meaningful endpoint intervals for knowledge discovery in the case where thresholds within an endpoint continuum were more informative than a precise value prediction. It should be noted that while this approach uses endpoint intervals within rules as predictions, the prediction output of the algorithm is a specific endpoint value for each instance in a dataset.

Our proposed approach makes the following major changes to ExSTraCS whenever a continuous endpoint is detected in the training dataset: (1) replace the discrete class/action value of a rule with an endpoint interval predicate; this is inspired by the idea of action interval predicates proposed in GCS [25], (2) the correct set [C] is defined as the set of classifiers in the match set [M] whose endpoint interval range includes the correct endpoint value (3) rule fitness is not only based on accuracy defined by inclusion in correct sets, but also on the average error in endpoint prediction (calculated from the centroid of the prediction range), (4) mutation and crossover within the genetic algorithm (GA) can probabilistically operate on rule endpoint intervals, (5) the traditional subsumption mechanism proposed by Wilson [20] has been completely removed and replaced by a largely new approach, better suited to ExSTraCS and continuous endpoints, and (6) a new prediction array voting scheme is applied for continuous endpoints converting rule prediction intervals into a single endpoint prediction value. We have implemented these proposed changes into ExSTraCS and evaluate their efficacy on a simple set of toy simulated datasets, as well as over a spectrum of complex simulated epidemiological datasets all with continuous-valued endpoints. As a preliminary study we do not seek to conclude that our approach is better suited than any other continuous endpoint LCS approach at solving any particular types of problems. Rather, we seek to both demonstrate that the ExSTraCS algorithm can be adapted to problems with a continuous endpoint and see if a rule-based machine learning approach can solve continuous endpoint problems by representing rule endpoint predictions as interval predicates.

2. METHODS

In this section we introduce (1) the ExSTraCS 2.0 algorithm, (2) the proposed algorithm adaptations for continuous endpoint problems and (3) the simulated datasets and evaluation strategy applied to test this proposed approach.

2.1 ExSTraCS 2.0

The ExSTraCS 2.0 algorithm [18] is a Michigan-style LCS algorithm, more generally referred to as a rule-based evolu-

tionary algorithm. ExSTraCS 2.0 has been expanded and adapted to better suit the needs of real-world supervised learning problems wherein effective and efficient classification, prediction, data mining, and/or knowledge discovery is the goal. The features of ExSTraCS 2.0 that most differentiate it from XCS [20] or UCS [2] include: a rule specificity limit to address scalability issues [18], built-in rapid expert knowledge (EK) generation algorithms [14], EK guided covering and mutation for efficient learning and scalability [15, 18], attribute tracking and feedback for reusing useful attribute combinations and characterizing patterns of heterogeneity [12], built-in rule compaction strategies [9], and the consolidation of explore/exploit to perform both simultaneously [13]. For a detailed description of the ExSTraCS 2.0 algorithm see [18]. For a complete software users guide see [11]. In this study the altered ExSTraCS 2.0 algorithm that includes our proposed continuous endpoint learning strategy will be referred to as ExSTraCS_CE 2.0. Both versions of this algorithm are available on *sourceforge.com* as ExSTraCS versions 2.02 and 2.03 respectively.

2.2 Continuous Endpoint Strategy

2.2.1 Rule Representation

Our modified rule representation preserves the condition representation described in [18] which only stores specified attribute states (for discrete attributes) or specified interval predicates (for continuous attributes). To promote efficient computation, attributes that are generalized in a rule (i.e. represented as a 'don't care' or '#' in traditional LCS algorithms) are simply left out of the rule representation. However, when ExSTraCS detects a continuous/real-valued endpoint as described in [14] and [18], the predicted class of a rule is represented as an interval predicate rather than a single discrete class. This endpoint interval predicate is maintained and evolved in a manner similar to the strategy implemented for continuous attribute intervals. Identical to discrete endpoint ExSTraCS learning, the interval predicate endpoint does not influence rule matching.

2.2.2 Forming a Correct Set

Following the formation of a match set [M], any rule in [M] with an interval predicate (i.e. continuous value range) that correctly includes the true continuous endpoint value of the current training instance is included in the correct set [C]. The formation of a [C] is an important part of the attribute tracking and feedback mechanisms previously developed for and implemented within ExSTraCS [12] to improve learning and characterize patterns of heterogeneity. This is one of the primary motivations for developing a continuous endpoint strategy that preserves the assembly of a correct set. Another would be to preserve correct set parent selection as a precursor for applying the GA to generate new offspring. In this case, only parent rules with an endpoint interval containing the current correct endpoint value will be utilized in the generation of offspring rules.

2.2.3 Rule Parameters

The rule parameters maintained by ExSTraCS are updated as previously described [18] with the following exceptions: (1) The accuracy of a rule is no longer simply the number of times a rule has been included in a [C] divided by the number of times that rule has been included in a [M] as originally introduced in UCS [2]. Preliminary trials indicated that in the context of continuous endpoints, this type of rule accuracy yielded rules with large interval predicates (sometimes spanning the entire range of endpoint values observed in the training dataset). This is logical, since a matching rule would be considered 'correct' as long as the true value fell within the specified endpoint interval. If this interval included all possible values, then such a rule would misleadingly be considered 'correct' every time it was in [M]. Instead, accuracy is calculated as $(1 - sumErr)/matchCount$. The $matchCount$ parameter is the number of times the given rule has been included in an [M]. The parameter $sumErr$ is updated whenever a rule is included in a [M]. If the endpoint interval of the rule does not include the true value (i.e. it does not get into [C]), then a value of 1 (i.e. the maximum error) is added to $sumErr$. Alternatively, if the rule is included in [C] then the error added to $sumErr$ is the scaled difference between the rule's prediction and the true continuous endpoint value $RuleErr$. Since a rule doesn't strictly have a specific prediction value, we apply the centroid of the the predicted endpoint interval as a surrogate prediction. The prediction error (i.e. the difference between this surrogate prediction and the true endpoint value) is then scaled by the maximum possible error for the given rule (i.e. the distance between the centroid of the endpoint interval and either its upper or lower bound). This serves two purposes: (1) the error added to $sumErr$ is always a value between 0 and 1 (which is critical to the calculation of rule accuracy), and (2) the range of a rule's endpoint interval alone does not impact error. This second purpose seeks to avoid biasing the system unnecessarily towards narrower endpoint intervals. An alternative approach might include scaling error by the maximum error possible given the range of endpoint values observed in the training dataset as a whole. However, preliminary testing suggested that this alternative scaling approach needlessly promotes much narrower endpoint intervals in problems where larger endpoint-value ranges define meaningful predictive patterns. This ultimately detracts from solution interpretability in such problems, where multiple rules would be evolved to represent a pattern that could have been captured by a single rule. However, this alternative approach may perform better in problems like function approximation, where a specific endpoint value is desired for each input.

As one additional caveat to the calculation of rule error, rules that have an endpoint prediction interval that goes beyond the bounds of endpoint values observed in the training dataset have an adjusted calculation for the surrogate prediction value. Specifically, if the upper or lower bound of a rule's prediction interval goes outside the bounds of observed endpoint values in the training dataset, the training-set observed boundary is used in the calculation of the interval centroid (i.e. the surrogate prediction) instead of the prediction interval bound. While rules are given the flexibility to evolve interval predicates somewhat outside the range of values observed in the training dataset, we restrict our calculation of predictions and error to the range of observed training values. This helps to avoid additional prediction bias introduced by a rule prediction interval that spans outside of observed values. At the same time, this gives rules the ability to potentially cover testing instances (i.e. instances that have not yet been seen) that may fall outside of the endpoint value extremes in the training data.

2.2.4 Genetic Algorithm

Following parent rule selection, the GA activates crossover with probability $\chi = 0.8$ as applied in [18]. When crossover is activated, if the endpoint is continuous, there is a 50% chance that the condition will crossover between two parent rules, otherwise the endpoint intervals will be crossed over. Endpoint crossover randomly picks between swapping the lower bounds or the upper bounds of the parent rule endpoint intervals. During mutation, the GA has been updated to include opportunity for the endpoint interval to be mutated in the same way as was described for continuous-valued attribute intervals [18]. Rule endpoint intervals will mutate with probability $\upsilon = 0.04$. The υ parameter is used to determine the number of attributes to be mutated as described in [18]. Additionally it should be noted that when covering generates a new rule, the initial endpoint interval is randomly generated as an attribute interval would be during covering [18].

2.2.5 Subsumption

The traditional GA and correct set subsumption mechanisms proposed by Wilson [20] were meant to apply a rule-generalization pressure to the rule population and were designed to address problems with discrete endpoints and a clean, or nearly clean signal. Given that our target problems are generally noisy, and extend to continuous-valued endpoints, preliminary testing unsurprisingly indicated that the original subsumption mechanisms were not applicable in their original conceptualization. While not the major focus of this work, we explore the implementation of an alternative subsumption mechanism designed explicitly for supervised learning in ExSTraCS designed to function in the context of continuous endpoint problems. This approach also relies on the assumption that only a finite amount of data is available as a training dataset (as is the case in many real-world problems). ExSTraCS_CE 2.0 excludes both traditional forms of subsumption and instead implements this newly proposed strategy.

Our proposed strategy activates once a given rule has had the opportunity to see all instances in the training set. At this point the rule is designated by ExSTraCS to be 'EpochComplete'. Every rule that is not yet 'EpochComplete' is checked to see if it has seen all training instances during the matching phase of the algorithm. When a rule becomes 'EpochComplete' the algorithm checks to see if that rule subsumes or can be subsumed by any other any other 'EpochComplete' rule in the population. A rule is defined as a subsumer if it is (1) 'EpochComplete', (2) has a greater or equal accuracy than the subsumed rule, (3) specifies the same attributes, or a subset of the attributes specified by the subsumed rule, (4) any specified continuous attributes have an interval that fits within the bounds of the subsumed rule, and (5) the endpoint interval fits within the bounds of the subsumed rule. The definition of a subsumer, along with examples of which rules might subsume another are given in Figure 1. We have opted for subsumers to have continuous valued intervals that must fit within the bounds of prospective subsumed rule. This is somewhat of a reversal on the typical strategy of subsumers being more general. While future work may reveal another approach to be better, the logic behind this decision is based on our desire to evolve rules that are maximally accurate, but specify the least number of attributes, while at the same time encouraging a more specific range of continuous attribute values along with a narrower endpoint intervals. We expect that by encouraging narrower continuous value ranges, we will promote more accurate rule predictions (since a given rule's prediction is based on it's centroid surrogate). Future work will explore other variations of this subsumption strategy.

While we have implemented this strategy exclusively for continuous endpoint problems, this concept could be successfully applied to discrete endpoint problems as an alternative to traditional subsumption mechanisms in the context of noisy discrete problem domains. This will be another target for ongoing work.

2.2.6 Prediction Array

Possibly the most critical component in adapting ExSTraCS to continuous endpoint analysis is the introduction of a new prediction array strategy for generating endpoint predictions as output for the algorithm. We have modeled our approach after the way that rules within the match set vote for discrete classes in UCS and XCS. In a preliminary attempt, we tried a strategy where the centroids of all rule endpoint intervals in [M] have a fitness/numerosity weighted influence on the determination of an endpoint prediction. However, unlike a prediction array for discrete endpoint problems, the 'bad' rules in [M] with a prediction that is poor, would still impact the value of the endpoint prediction. Therefore, the approach adopted here segments the entire range of possible endpoint values based on the high and low limits of endpoint interval predicates of rules in [M]. Figure 2 illustrates this approach. Essentially these range-limits define unique segments (i.e. candidate intervals) that will be treated as 'classes' from which we will choose the one with the largest vote to be the best interval. Once the candidate intervals are determined, any rule in [M] with an endpoint interval that includes a given candidate interval contributes a vote of $numerosity * fitness$ to that candidate interval segment. The candidate interval with the highest vote is selected to be the 'best' prediction interval. As was the case with the calculation of rule accuracy, the resulting prediction value is the centroid of this interval. While this centroid prediction surrogacy offers the system flexibility, this approach is expected to achieve a somewhat limited accuracy for problems like function approximation, where a continuous mapping between attribute inputs and endpoint outputs is sought.

2.3 Evaluation

In order to evaluate our proposed strategy, we have generated a number of simulated datasets including a group of simple clean-signal toy datasets exploring some basic properties of a continuous endpoint problem. We have also generated a full spectrum of complex noisy continuous endpoint datasets based on discrete datasets used to evaluate the original versions of ExSTraCS [14, 18]. A total of 6 toy datasets with a clean signal were simulated (each with 20 attributes, and 1600 instances) where only a single continuous-valued attribute was predictive of the continuous-valued endpoint. In three of these datasets, the non-predictive attributes were randomly simulated discrete values, while in the other three they were randomly simulated continuous values. For both of these sets of three datasets, one dataset was simulated with a linear relationship to the endpoint (as would be expected in a simple function approximation problem), and the

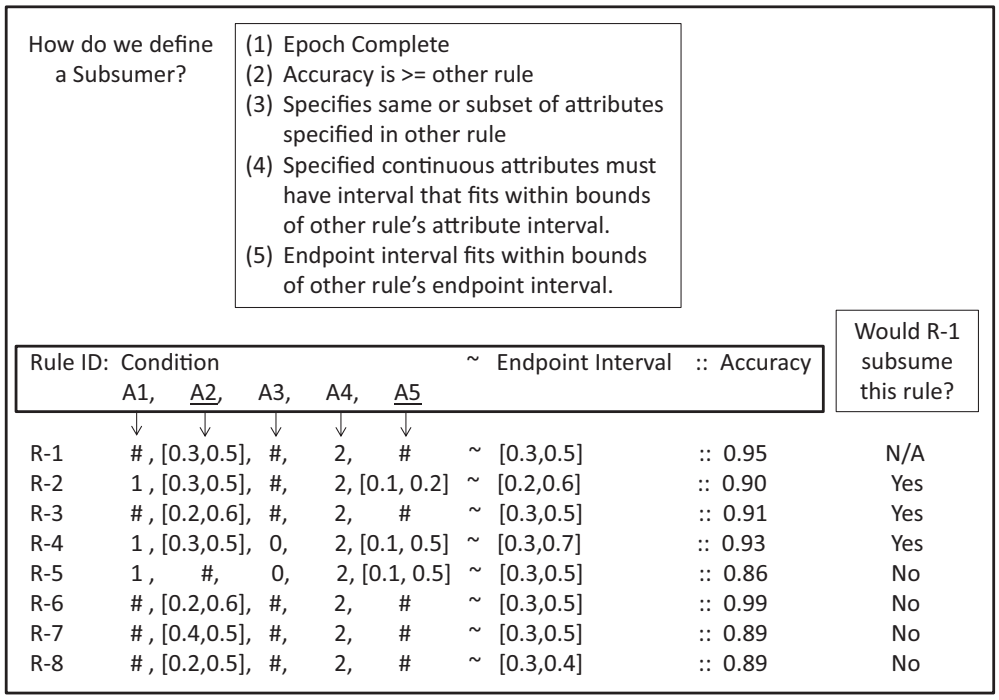

Figure 1: A summary of the definition of a subsumer. Rule examples are provided (R-1 to R-8). Each rule condition is defined by 5 attributes (A1 - A5). Attributes A2 and A5 are continuous-valued. Rules that can be subsumed by 'R-1' are indicated on the right-hand column.

other two datasets simulated threshold relationships with either two or four relevant endpoint intervals respectively (corresponding to one or three relevant thresholds). In the first threshold model, if the predictive attribute was in the lower half of values simulated for that attribute, the instance was assigned a random continuous endpoint value between 0 and 50. Otherwise it was assigned a value between 50 and 100. A similar pattern was constructed for the dataset with four relevant endpoint intervals. Lastly, a version of each dataset was generated with a permuted endpoint (i.e. the endpoint column in the dataset was randomized in to eliminate any modeled signal. All 12 of these datasets are available on *sourceforge.com* included with the ExSTraCS_CE 2.0 implementation. For each of these 12 toy datasets, 10-fold cross validation was performed in analyzing them with ExSTraCS_CE 2.0 for 200,000 learning iterations as well as application of the quick rule filter (QRF) to remove clearly poor rules. The goal of this simple toy dataset analysis was to determine how ExSTraCS_CE 2.0 would perform on function approximation-like problems vs. continuous endpoint problems that model associations based on some relevant number of thresholds (i.e. high vs. low endpoint value). In this evaluation of toy datasets we compare key performance metrics between the non-permuted and permuted versions of these datasets to see how the algorithm functions when there is or is not a respective underlying model. In these toy datasets, 'power' refers to the proportion of cross validation datasets in which the modeled predictive attribute was correctly prioritized by the algorithm (i.e. it was the most specified attribute in rules in the final population).

The second part of our evaluation was performed over a set of 960 noisy complex simulated genetic datasets with 20 discrete-valued attributes generated using GAMETES [16]. Similar to [14] and [18], each dataset concurrently modeled patterns of epistasis and heterogeneity, where four of the attributes were predictive and 16 were non-predictive. However, for this investigation, each dataset was converted to a continuous-valued endpoint threshold problem by replacing discrete-valued classes of '0' with a random value between 0 and 50 and discrete-valued classes of '1' with a random value between 50 and 100. 20 replicates of each dataset were analyzed and 10-fold cross validation (CV) was employed to measure average testing accuracy and account for over-fitting. As with the toy datsets, an archive of 960 datasets with permuted endpoints was also generated for comparison. ExSTraCS_CE 2.0, with and without our proposed subsumption mechanism were each run on this full set of 9600 datasets (960*10) up to 200,000 learning iterations with QRF, as well as upon the 9600 permuted versions of these datasets for comparison. In each analysis, default ExSTraCS parameters were applied as described in [18]. Similar to our previous analyses on ExSTraCS, pair-wise statistical comparisons were made using the Wilcoxon signed-rank tests [14, 18]. All statistical evaluations were completed using R. Comparisons were considered to be significant at $p \leq 0.05$. Analyses were performed using 'Discovery', a 2400 core Linux cluster available to the Dartmouth research community.

Statistical comparisons were performed over a set of key performance metrics. Accuracy metrics were calculated as a respective 'balanced accuracy' to account for any imbalanced datasets. 'Both Power' is the ability to correctly identify both two-locus heterogeneous models. 'Single Power' is the ability to have found at least one. 'Co-occur. Power'

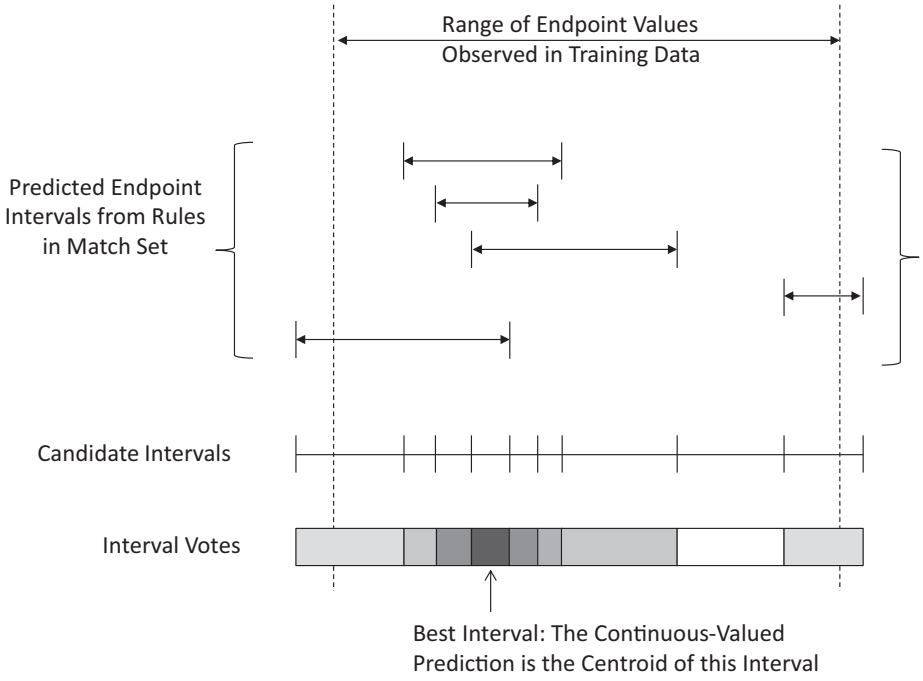

Figure 2: An illustration of the proposed prediction array mechanism for determining a specific continuous endpoint prediction value.

indicates the ability to detect the correct heterogeneous pattern. Generality refers to classifier generality, or the average proportion of unspecified attributes across the classifier population. Macro Population refers to the number of unique classifiers in the classifier population. These metrics are described in greater detail in [18].

3. RESULTS AND DISCUSSION

Tables 1 and 2 present the results of the 6 toy datasets that include discrete or continuous non-predictive values, respectively. Every metric entry in either table is the average over 10 cross validation analyses. Beginning with Table 1, we observe that ExSTraCS_CE 2.0 is able to obtain a much higher testing accuracy than for the associated permuted datasets for each of the three models. Notably, with permutation, testing accuracy typically ends up with a value close to 0.75 (this is approximately the best accuracy we might expect by random chance). This finding suggests that training and testing accuracies computed using our continuous endpoint strategy are not directly comparable to discrete endpoint accuracy score, where randomly picking between two classes would yield an expected accuracy of 0.5. Further supporting the ability of ExSTraCS_CE 2.0 to learn either linear or threshold models, we find that the predictive attribute is correctly identified out of the 20 attributes in the dataset for the linear, 3-threshold, and 1-threshold models (indicated by a power of 1.0). This is in comparison to the respective permuted datasets which yield a power of 0 to 0.1 in these analyses. Furthermore, rule generality was much higher for each model than when compared with the respective permuted analysis. Macro population size, on the other hand, was consistently (sometimes dramatically) lower for each model than for the permuted analyses. These testing accuracy, power, rule generality, and macro population size findings were also consistent in the toy datasets with continuous non-predictive attributes summarized in Table 2. Also consistent between both tables is the observation that lower testing accuracies were observed for the 1-Threshold datasets than for the Linear or 3-Threshold models. This finding makes sense, given that in the 1-Threshold model, we can only hope to predict that an endpoint value is either high or low (i.e. in the upper or lower set of endpoint values). Therefore, the error in specific endpoint value predictions is expected to be larger. Comparing Table 1 with Table 2, we also notice that typically both the macro population size and the run time are dramatically larger for the datasets that include continuous non-predictive attributes. This enforces the idea that to effectively solve problems with continuous attributes or endpoints additional time and/or a greater number of rules in the population may be required. As a proof of concept these toy datasets indicate that our proposed endpoint interval approach is functional, and flexible enough to handle very different types of continuous endpoint problems from function approximation to threshold models (where a single threshold is the most difficult to learn). While we cannot say definitely without statistical analysis on these toy datasets that our results are meaningful, based on previous evaluations on similarly sized datasets using the original ExSTraCS algorithm, we believe these results to be strongly indicative of successful modeling/prediction and a overall promising strategy.

As a more challenging and comprehensive analysis Table 3 summarizes the findings over our set of 960 complex genetic simulated datasets with continuous endpoint values

Table 1: ExSTraCS_CE 2.0 Toy Dataset Analyses: Discrete Non-Predictive Attributes

200,000 Iterations +QRF						
Performance Statistics	Discrete Non-Predictive Attributes					
	Linear	⇒ Permuted	3-Threshold	⇒ Permuted	1-Threshold	⇒ Permuted
Train Accuracy	.9052	.7531	.9140	.7509	.8579	.7504
Test Accuracy	.8979	.7450	.9106	.7434	.8547	.7415
Power	1.0	0.1	1.0	0	1.0	0
Rule Generality	.9397	.7670	.8607	.7635	.8446	.7647
Macro Population	281.5	1343.2	842.9	1326.9	1118.0	1376.2
Run Time (min)	101.73	117.77	47.54	122.28	104.05	115.59

Table 2: ExSTraCS_CE 2.0 Toy Dataset Analyses: Continuous Non-Predictive Attributes

200,000 Iterations +QRF						
Performance Statistics	Continuous Non-Predictive Attributes					
	Linear	⇒ Permuted	3-Threshold	⇒ Permuted	1-Threshold	⇒ Permuted
Train Accuracy	.9419	.7489	.9202	.7481	.8724	.7497
Test Accuracy	.9394	.7427	.9170	.7407	.8670	.7435
Power	1.0	0	1.0	0	1.0	0
Rule Generality	.8890	.6312	.7193	.6250	.6886	.6196
Macro Population	944.5	1554.9	1470.8	1548.6	1501.8	1557.5
Run Time (min)	77.42	197.39	214.83	219.11	244.21	221.62

simulated with a 1-threshold. Each metric value in the table is the average over all 9600 cross validation runs of ExSTraCS_CE 2.0 either on the original simulated datasets, or upon the permuted versions (in which the association with endpoint has been obliterated). Despite the complexity of these problems, we observe a significantly higher testing accuracy, and significantly better power estimates than on the permuted datasets. While significant, the increase in testing accuracy is small. However we expect only a small increase in testing accuracy given that these datasets are extremely noisy added to the fact that they are 1-threshold models. Notably this strategy takes about twice as long to run as the discrete endpoint implementation of ExSTraCS 2.0 [18], but this is to be expected considering the added complexity of checking and evolving endpoint intervals and employing the newly proposed subsumption mechanism.

Lastly, Table 4 summarizes the results in comparing the performance of ExSTraCS_CE 2.0 with and without our proposed subsumption mechanism. While we observe a dramatic and highly significant reduction in macro population size, as well as a significant rise in rule generality as would be expected for our proposed subsumption mechanism, we observe no significant impact on testing accuracy, and significant loses in three power metrics. Additionally, as implemented, this mechanism more than doubles our run time. These trade-offs with success will lead us to revisit subsumption in future work, as the current proposed approach is clearly not optimal.

Table 3: ExSTraCS_CE 2.0: Original vs. Permuted Datasets

200,000 Iterations +QRF			
Performance Statistics	ExSTraCS_CE 2.0		
	Original	Permuted	p
Train Accuracy	.8015	.7913	↑ ***
Test Accuracy	.7453	.7284	↑ ***
Both Power	.2417	0	↑ ***
Single Power	.5844	0	↑ ***
Both Co-Power	0	0	-
Single Co-Power	.5260	0	↑ ***
Rule Generality	.8053	.7922	↑ ***
Macro Population	984.55	1061.39	↓ ***
Run Time (min)	136.55	149.00	↓ ***

– No significant change
* $p < 0.05$ (Direction of change given by arrows)
** $p < 0.001$
*** $p << 0.001$

4. CONCLUSIONS

In this study we have introduced a new approach for rule-based supervised learning in problems with a continuous or quantitative endpoint. This approach has been designed particularly to expand the ExSTraCS algorithm to handle these sorts of problems. This new strategy calls for a number of modifications to ExSTraCS, but preserves the formation of correct sets. Results from this work in progress suggest that rule endpoint intervals can effectively and flexibly learn patterns in both function approximation-like problems, as well as on problems where one or more threshold are meaningful to underlying patterns of association. We also have demonstrated that our proposed prediction array strategy can effectively make specific endpoint value predictions by combining multiple rule endpoint intervals from a given match set. However it is unlikely that this approach would achieve better endpoint predictions on function approximation problems than other LCS computed endpoint strategies such as XCSFCA [10] or using code fragments [5, 6]. This is primarily due to the use of a surrogate predicted endpoint value by rules which instead learn endpoint intervals. Lastly, while we report somewhat mixed results for the proposed subsumption mechanism, we believe that this will make an interesting target for future work. Ultimately, this study indicates that ExSTraCS can be expanded to continuous endpoint problems, however we do not claim that it is better than any other LCS continuous endpoint strategy. We do suspect that our rule endpoint interval strategy allows for better problem handling flexibility. Future work will seek to compare and perhaps integrate this approach with one that adopts code fragments as proposed by [5, 6].

Table 4: ExSTraCS_CE 2.0: With vs. Without Subsumption

200,000 Iterations +QRF			
Performance Statistics	ExSTraCS_CE 2.0		
	With Sub.	No Sub.	p
Train Accuracy	.8015	.8003	↑ ***
Test Accuracy	.7453	.7455	-
Both Power	.2417	.2615	↓ ***
Single Power	.5844	.6490	↓ ***
Both Co-Power	0	0	-
Single Co-Power	.5260	.5490	↓ ***
Rule Generality	.8053	.7694	↑ ***
Macro Population	984.55	1314.55	↓ ***
Run Time (min)	136.55	51.67	↓ ***

5. ACKNOWLEDGMENTS

This work was supported by NIH grants AI59694, LM009012, LM010098, EY022300, LM011360, CA134286, and GM103534.

6. REFERENCES

[1] J. Bacardit, E. K. Burke, and N. Krasnogor. Improving the scalability of rule-based evolutionary learning. *Memetic Computing*, 1(1):55–67, 2009.

[2] E. Bernadó-Mansilla and J. M. Garrell-Guiu. Accuracy-based learning classifier systems: models, analysis and applications to classification tasks. *Evolutionary Computation*, 11(3):209–238, 2003.

[3] A. Bonarini. An introduction to learning fuzzy classifier systems. In *Learning Classifier Systems*, pages 83–104. Springer, 2000.

[4] M. V. Butz. Kernel-based, ellipsoidal conditions in the real-valued xcs classifier system. In *Proceedings of the 7th annual conference on Genetic and evolutionary computation*, pages 1835–1842. ACM, 2005.

[5] M. Iqbal, W. N. Browne, and M. Zhang. Xcsr with computed continuous action. In *AI 2012: Advances in Artificial Intelligence*, pages 350–361. Springer, 2012.

[6] M. Iqbal, W. N. Browne, and M. Zhang. Evolving optimum populations with xcs classifier systems. *Soft Computing*, 17(3):503–518, 2013.

[7] P. L. Lanzi and D. Loiacono. Classifier systems that compute action mappings. In *Proceedings of the 9th annual conference on Genetic and evolutionary computation*, pages 1822–1829. ACM, 2007.

[8] A. Orriols-Puig, J. Casillas, and E. Bernadó-Mansilla. Fuzzy-ucs: a michigan-style learning fuzzy-classifier system for supervised learning. *Evolutionary Computation, IEEE Transactions on*, 13(2):260–283, 2009.

[9] J. Tan, J. Moore, and R. Urbanowicz. Rapid rule compaction strategies for global knowledge discovery in a supervised learning classifier system. In *Advances in Artificial Life, ECAL*, volume 12, pages 110–117, 2013.

[10] H. T. Tran, C. Sanza, Y. Duthen, and T. D. Nguyen. Xcsf with computed continuous action. In *Proceedings of the 9th annual conference on Genetic and evolutionary computation*, pages 1861–1869. ACM, 2007.

[11] R. Urbanowicz. *ExSTraCS 2.1*, 2015 (accessed February 4, 2015).

[12] R. Urbanowicz, A. Granizo-Mackenzie, and J. Moore. Instance-linked attribute tracking and feedback for michigan-style supervised learning classifier systems. In *Proceedings of the fourteenth international conference on Genetic and evolutionary computation conference*, pages 927–934. ACM, 2012.

[13] R. Urbanowicz and J. Moore. The application of michigan-style learning classifier systems to address genetic heterogeneity and epistasis in association studies. In *Proceedings of the 12th annual conference on Genetic and evolutionary computation*, pages 195–202. ACM, 2010.

[14] R. J. Urbanowicz, G. Bertasius, and J. H. Moore. An extended michigan-style learning classifier system for flexible supervised learning, classification, and data mining. In *Parallel Problem Solving from Nature–PPSN XIII*, pages 211–221. Springer, 2014.

[15] R. J. Urbanowicz, D. Granizo-Mackenzie, and J. H. Moore. Using expert knowledge to guide covering and mutation in a michigan style learning classifier system to detect epistasis and heterogeneity. In *Parallel Problem Solving from Nature-PPSN XII*, pages 266–275. Springer, 2012.

[16] R. J. Urbanowicz, J. Kiralis, N. A. Sinnott-Armstrong, T. Heberling, J. M. Fisher, and J. H. Moore. Gametes: a fast, direct algorithm for generating pure, strict, epistatic models with random architectures. *BioData mining*, 5(1):16, 2012.

[17] R. J. Urbanowicz and J. H. Moore. Learning classifier systems: a complete introduction, review, and roadmap. *Journal of Artificial Evolution and Applications*, 2009.

[18] R. J. Urbanowicz and J. H. Moore. Exstracs 2.0: Description and evaluation of a scalable learning classifier system. *In Press*, 2015.

[19] M. Valenzuela-Rendón. The fuzzy classifier system: A classifier system for continuously varying variables. In *Proceedings of the Fourth Intemational Conference on Genetic Algorithms (Morgan Kauffman)*, 1991.

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

[21] S. W. Wilson. Get real! xcs with continuous-valued inputs. In *Learning Classifier Systems*, pages 209–219. Springer, 2000.

[22] S. W. Wilson. Function approximation with a classifier system. In *Proc. 3rd Genetic and Evolutionary Computation Conf.(GECCO'01)*, page 974À981. Citeseer, 2001.

[23] S. W. Wilson. Classifiers that approximate functions. *Natural Computing*, 1(2-3):211–234, 2002.

[24] S. W. Wilson. Classifier systems for continuous payoff environments. In *Genetic and Evolutionary Computation–GECCO 2004*, pages 824–835. Springer, 2004.

[25] S. W. Wilson. Three architectures for continuous action. In *Learning Classifier Systems*, pages 239–257. Springer, 2007.