An Extended Michigan-Style Learning Classifier System for Flexible Supervised Learning, Classification, and Data Mining

Ryan J. Urbanowicz, Gediminas Bertasius, and Jason H. Moore

Institute for Quantitative Biomedical Sciences, Department of Genetics
Dartmouth Medical School, Lebanon, NH, USA
{ryan.j.urbanowicz,jason.h.moore}@dartmouth.edu
http://www.epistasis.org/

Abstract. Advancements in learning classifier system (LCS) algorithms have highlighted their unique potential for tackling complex, noisy problems, as found in bioinformatics. Ongoing research in this domain must address the challenges of modeling complex patterns of association, systems biology (i.e. the integration of different data types to achieve a more holistic perspective), and 'big data' (i.e. scalability in large-scale analysis). With this in mind, we introduce ExSTraCS (Extended Supervised Tracking and Classifying System), as a promising platform to address these challenges using supervised learning and a Michigan-Style LCS architecture. ExSTraCS integrates several successful LCS advancements including attribute tracking/feedback, expert knowledge covering (with four built-in attribute weighting algorithms), a flexible and efficient rule representation (handling datasets with both discrete and continuous attributes), and rapid non-destructive rule compaction. A few novel mechanisms, such as adaptive data management, have been included to enhance ease of use, flexibility, performance, and provide groundwork for ongoing development.

Keywords: Learning Classifier System, Genetics, Epidemiology, Epistasis, Heterogeneity, Evolutionary Algorithm, Systems Biology

1 Introduction

Machine learning algorithms driven by evolutionary mechanisms offer a promising avenue for data mining within complex, noisy problem domains. Michigan-style learning classifier systems (LCS) constitute a unique class of algorithms that distribute learned patterns over a collaborative population of individually interpretable (IF:THEN) rules, allowing them to flexibly and effectively describe complex and diverse problem spaces found in behavior modeling, function approximation, classification, and data mining. Michigan LCS algorithms apply iterative rather than batch-wise learning, meaning that rules are evaluated and evolved one data instance at a time. This makes them naturally well-suited to learning different problem niches found in multi-class, latent-class, or heterogeneous problem domains. LCS algorithms are fundamentally multi-objective,

evolving rules toward maximal accuracy and generality(i.e.rule simplicity) to improve predictive performance [1]. We focus on classification and data mining problems in genetics and epidemiology where risk factors that explain variation in disease phenotypes are sought. Certain complicating phenomena are known to interfere with the traditional mapping of genotype to phenotype [2]. We explicitly consider two such phenomenon; epistasis (i.e. gene-gene interaction) and genetic heterogeneity. Also, new systems biology approaches will require the integration of different data types (e.g. genetic, epigenetic and environmental) embracing a more holistic perspective when searching for predictive or causal disease risk factors. Difficulty is further compounded in the context of large-scale bioinformatic investigations where the computational and methodological limitations hinder scalability with increasing numbers of attributes and/or training instances.

With these challenges in mind, we introduce the Extended Supervised Tracking and Classifying System (ExSTraCS). ExSTraCS has been primarily designed to address complex, noisy, supervised learning, single-step problem domains with 2 or more balanced/imbalanced classes, and with continuous or discrete attributes. ExSTraCS is descended from a lineage of Michigan-style LCS algorithms, founded on the architecture of Wilson's Extended Classifier System (XCS) [3], the most successful and best-studied LCS algorithm to date. The Supervised Classifier System (UCS) [4] replaced XCS's reinforcement learning scheme with a supervised learning strategy to deal explicitly with single-step problems such as classification and data mining. Comparing select Michigan and Pittsburgh-style LCS algorithms, UCS showed particular promise when applied to complex biomedical data mining problems with patterns of epistasis and heterogeneity [5, 6]. UCS inspired two algorithmic expansions named Attribute Tracking and Feedback UCS (AF-UCS) and Expert Knowledge UCS (UCS-EK). AF-UCS introduced mechanisms that improved learning and uniquely allowed for the explicit characterization of heterogeneous patterns and the identification of candidate disease subgroups [7, 8]. UCS-EK incorporated of expert knowledge into UCS learning for smart population initialization, directed rule discovery, and reduced run time [9]. Recently, novel rapid rule compaction strategies were developed and evaluated for post-processing rule populations to enhance interpretability and improve predictive performance [10]. ExSTraCS merges successful components of this algorithmic lineage with other valuable LCS research, and a redesigned UCS-like framework with a few novel features. In addition to integrating attribute tracking/feedback, expert knowledge covering, and rapid rule compaction, ExSTraCS (1) adopts a flexible and efficient rule representation similar to the one described in [11], to accommodate data with both discrete and continuous attributes, (2) outputs attribute tracking scores and global statistics (in addition to a rule population) for significance testing, and visualization-guided knowledge discovery as described in [12], (3) includes an adaptive data detection scheme to adjust the algorithm to the characteristics of the dataset, and (4) includes a built-in selection of four attribute weighting algorithms to discover potentially useful expert knowledge as a pre-processing step.

2 ExSTraCS

The Extended Supervised Tracking and Classifying System (ExSTraCS) combines a number of existing and completely novel aspects into a single, flexible LCS framework aimed at overall functionality, ease of use, as a platform for ongoing algorithmic development. The algorithm itself is coded in Python, well annotated, and freely available on *sourceforge.net*. We begin with an overview of ExsTraCS referencing a schematic of major components given in Figure 1. We follow with details of components that differentiate ExSTraCS from the UCS or XCS algorithms.

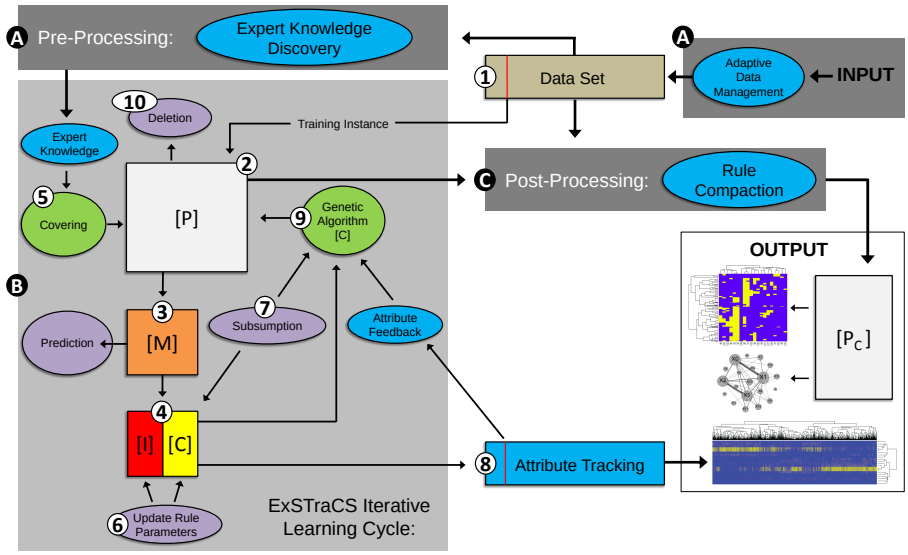

Fig. 1. ExSTraCS Schematic: Ovals are mechanisms, bordered squares are sets of either data or classifiers, green = classifier discovery mechanism, purple = traditional LCS mechanism, and blue = mechanisms unique to ExSTraCS.

ExSTraCS begins with (A) data pre-processing, followed by (B) algorithm learning/training, and ending with (C) rule population post-processing. (A) ExSTraCS will accept a finite dataset with some number of independent attributes and a single class variable as the training dataset. A testing dataset may be optionally loaded for complete rule population evaluations. Adaptive data management initially determines and stores key characteristics of this dataset for use during learning iterations. Lastly, expert knowledge (EK) may be loaded or discovered from the dataset using one of four implemented attribute weighting algorithms. These weights are converted describing relative probabilities that attributes in the data will be valuable for discriminating class/endpoint. (B) The core ExSTraCS algorithm largely follows a typical iterative Michigan-style learning cycle that includes the following 10 steps repeatedly up to a maximum number of learning iterations: (1) One training instance is taken from the dataset without replacement. (2) The training instance is passed to a population

[P] of classifiers/rules that is initially empty. A *classifier* is a simple IF/THEN rule comprised of a condition (i.e. specified attribute states), and what is traditionally referred to as an action (i.e. the state of the class or endpoint). (3) A match set [M] is formed, that includes any classifier in [P] that has a condition matching the training instance. (4) [M] is divided into a correct set [C] and an incorrect set [I] based on whether each classifier specified the correct or incorrect class/phenotype. (5) If, after steps 3 and 4, [C] is empty, covering applies expert knowledge to intelligently generate a matching and 'correct' classifier added to [M] and [C]. (6) For every classifier in [P], a number of parameters are maintained and updated throughout the learning process such as: numerosity (the number of copies of a given classifier in [P]), rule accuracy which is the proportion of times in which a classifier has been in a [C] over all times it has been in a [M]; and classifier fitness, which is simply equal to classifier accuracy in this implementation (i.e. $\nu = 1$). We had previously observed that placing too much emphasis on optimal accuracy in calculating fitness led to dramatic overfitting in noisy problem domains [5]. Classifier parameters (e.g. fitness) are updated for classifiers within [C] and [I]. (7) Subsumption, a generalization mechanism is applied to [C] [3]. A similar subsumption mechanism is also applied to new classifiers generated by the genetic algorithm (GA). (8) Classifiers in [C] are used to update attribute tracking scores for the current training instance. (9) The GA uses tournament selection to pick two parent classifiers from [C] based on fitness and generates two offspring classifiers which are added to [P]. The GA includes two discover operators: crossover and mutation ($\chi = 0.8$ and $\upsilon = 0.04$, respectively). All classifiers in [C] and [I] are returned to [P]. (10) Lastly, whenever the size of [P] is greater than the specified maximum, a deletion mechanism decrements the numerosity of a classifier (assuming it is > 1) or removes it from [P]. Deletion probability is a function of classifier numerosity, average [M] size and is inversely proportional to fitness. Notably, ExSTraCS cycles do not alternate between an explore/exploit phases as described in XCS [3] due to supervised learning. However, for performance tracking and prediction evaluation, a prediction array is generated every iteration from [M] to obtain a class prediction. A class prediction is made by a fitness weighted vote of all classifiers within [M]. The class with the largest 'vote' is the predicted class. (C) After all learning iterations have completed, rule compaction is applied as a post-processing step to remove poor and/or redundant rules from [P] to yield [P_c]. Upon request, ExSTraCS will yield up to four distinct output files after the final iteration, or any iteration at which a full evaluation is requested. These include (1) the population of classifiers collectively constituting the prediction 'model' (Note: ExSTraCS is uniquely set up to load a given population file and continue learning from where it left off), (2) population statistics, summarizing major performance statistics including global training and testing accuracy of the classifier population [12], (3) co-occurrence scores for the top specified pairs of attributes in the dataset [12], and (4) attribute tracking scores for each instance in the dataset [7]. These outputs may be evaluated and visualized to facilitate knowledge discovery as described in [12].

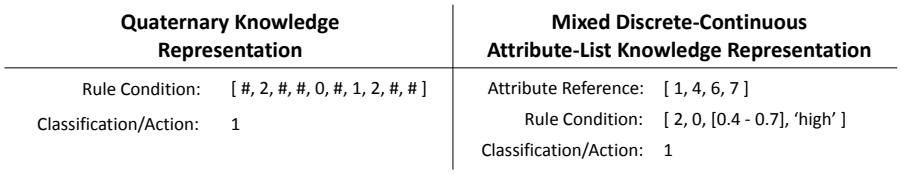

Fig. 2. Knowledge Representations: Quaternary vs. Mixed Discrete-Continuous Attribute List. The '#' symbol indicates 'attribute not specified', which matches any attribute state. The mixed representation only stores specified attributes, represents continuous value states as flexible 'ranges', and allows for non-numerical states.

Adaptive Data Management We introduce a simple adaptive data management (ADM) scheme to facilitate ease of use, improve efficiency, and algorithmic adaptation to different datasets. ADM will load and format training (and optionally testing) data, automatically identifying key characteristics including: number of attributes, number of instances, the location of the endpoint variable column, and the location of a column for instance identifiers (optional, but useful in tying attribute tracking scores back to specific individuals in the dataset). Also, ADM examines state values for all attributes and applies a user defined run parameter (*discreteAttributeLimit*) to determine and store whether each attribute is to be treated as discrete or continuous. If discrete, each possible state value stored, while if continuous, the maximum and minimum values are stored, for use in limiting covering and GA mechanisms. We plan to expand ADM to store state frequency information which we expect can be applied to further improve performance. ADM reduces redundancy, simplifies data formatting requirements, and paves the way for further algorithmic enhancements.

Knowledge Representation Our prior implementations of UCS [5], AF-UCS [7], and UCS-EK [9] were coded with a quaternary knowledge representation to operate on single nucleotide polymorphism (SNP) case/control data. SNPs are discrete genetic attributes with encoded states (0,1,or 2). ExSTraCS adopts a mixed discrete-continuous attribute-list knowledge representation (see Figure 2) allowing learning on datasets with discrete and/or continuous attributes. This strategy is quite similar to the one proposed by Bacardit in [11] which extended the attribute-list knowledge representation (ALKR), designed for continuous attributes, with the GABIL discrete attribute representation [13]. ALKR only stores information about attributes that are specified in a classifier which significantly reduces run time in both matching and attribute tracking. This effect is particularly important in datasets with a large number of non-predictive attributes. ExSTraCS keeps the ALKR representation but avoids the GABIL representation in favor of a simpler but less generalizable strategy for representing discrete attribute states. Specifically, classifier conditions can only specify one state for discrete attributes, while GABIL allows for some subset of attribute states to be simultaneously specified. While this may be advantageous for evolving a maximally compact rule-set, this approach is not in-line with the global approach to knowledge discovery proposed in [12] which relies on important attributes being specified more often across rules in the greater population,

a valuable component to addressing significant noise in LCS data mining. Additionally, this stricter representation yields individual rules that are arguably easier to interpret (less ambiguity within the IF/THEN statement) and that are likely be more accurate individual predictors (since they independently capture a more specific set of attribute states). This representation requires modifications to both covering and the genetic algorithm in order to handle continuous attributes (implemented as described in [11]).

Attribute Tracking and Feedback Attribute tracking (AT) is akin to long-term memory for supervised, iterative learning (see (8) in Figure 1). For a finite training dataset, a vector of accuracy scores is maintained for each instance in the data. In other words, for every instance in the data we increase attribute weights based on which attributes are being specified in rules found in [C] every iteration. Post-training, these scores can be applied to characterize patterns of association in the dataset, in particular heterogeneous patterns which might suggest clinical patient subgroups that may be targeted for research, treatment, or preventative measures [7]. Note that using attribute tracking alone does not impact learning performance. Attribute feedback (AF) is applied to the GA mutation and crossover operators, probabilistically directing rule generalization based on the AT scores from a randomly selected instance in the dataset. The probability that AF will be used in the GA is proportional to the algorithm's progress through the specified number of learning iterations (i.e. AF is applied infrequently early-on, but frequently towards the end). Note that in developing ExSTraCS we realized that AF-UCS was not using the AT scores from the current training instance (as mistakenly described in [7]), but rather the scores from a neighboring instance. This 'error' turned out to be essential to recapitulate attribute feedback performance. AF speeds up effective learning by gradually guiding the algorithm to more intelligently explore reliable attribute patterns. These mechanisms and their application are further detailed in [7] and [8].

Expert Knowledge Covering Previous work exploring the utilization of expert knowledge (EK) in UCS indicated that EK, utilized as probibalistic weights for specifying attributes in rules, significantly sped up learning when applied to covering, but yielded inconsistent success when applied to GA operators [9]. Therefore, ExSTraCS adopts EK covering. EK is essentially an external bias introduced to better guide learning, such that attributes more likely to be important tend to be specified more often when covering. In other words, classifiers tend to be initialized in parts of the problem space deemed by the EK to be mostly likely to predict class status. Notably, the utility of EK is only as good as the quality of the information behind the weights. EK covering is implemented in ExSTraCS as described in [9] including the calculation of EK probability weights from raw EK scores, and the application of these weights within the covering mechanism. In theory the source of EK is up to the user (i.e. classifier population initialization can be biased towards whatever attributes desired). In [9], raw EK scores were obtained externally using a rapid attribute weighting algorithm called SURF [14], designed to estimate attribute quality, in terms of predicting class status. For convenience and flexibility, we have implemented SURF as well

as three other related attribute weighting algorithms into ExSTraCS (ReliefF [15], SURF* [16], and MultiSURF [17]) from which the user may select and discover EK scores for their respective datasets. Each algorithm has been re-implemented to allow for discrete and continuous attributes. ExSTraCS handles EK discovery is a pre-processing step (see (A) in Figure 1). This study applies MultiSURF to discover EK, as it is the newest and most powerful.

Rule Compaction ExSTraCS makes the six rule compaction strategies evaluated in [10] available to post-process the classifier population (see (C) in Figure 1). Rule compaction utilizes the whole training dataset to consolidate the classifier population with the goal of improving interpretation and knowledge discovery. Comparisons in [10] suggested that simple Quick Rule Filtering (QRF) was both the fastest, and particularly was well suited to the theme of global knowledge discovery [12] where it is more important to preserve or improve performance than to minimize rule population size(useful for knowledge discovery by manual rule inspection) [10]. This study applies QRF.

Miscellaneous ExSTraCS naturally handle missing data points without requiring imputation bias. Missing data points require a standard unique designation, and when encountered they match any attribute state specified in the condition of a classifier. If desired, imputation can still be performed prior to running ExSTraCS, but this is not currently built-in. ExSTraCS can perform a complete evaluation of the classifier population as a whole at user specified iterations, including assessments of training and testing accuracy. Balanced accuracy is used to avoid accuracy calculation bias in multi-class and imbalanced datasets. ExSTraCS centralizes and organizes all run parameters in a readable configuration file, required to run the algorithm. Included in these parameters is the option to deactivate major ExSTraCS mechansims as desired, such as attribute tracking/feedback, expert knowledge, and rule compaction.

3 Results and Discussion

We evaluate ExSTraCS using two separate simulation studies (with discrete or continuous attributes respectively) each including a total of 960 diverse datasets (spanning from easily solvable to currently unsolvable) with underlying predictive models that simulated patterns of epistasis and heterogeneity concurrently. Discrete attribute SNP datasets very similar to those used in [7, 9, 10] were simulated using GAMETES [18] to have; architectures at maximum and minimum detection difficulty, heritabilities (i.e. the proportion of class variance that can be attributed to modeled attributes) of (0.1, 0.2, or 0.4), a minor allele frequency of 0.2, 20 attributes (only four of which were predictive and 16 were noise), sample sizes of (200, 400, 800, or 1600) and a heterogeneous mix ratio of either (50:50 or 75:25) (e.g. 75% of instances were generated from one epistatic model, and 25% were generated from a different one). 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. These datasets were very similar to the ones used in [7, 9, 10]. 960 corresponding continuous-valued versions of

these datasets were generated by transforming discrete values into random values within specified continuous intervals (e.g. a discrete attribute with states 0, 1, or 2, was transformed to have a random continuous value within the respective ranges of 0-50, 50-100, or 100-150. ExSTraCS was run up to 200,000 learning iterations but performance was also evaluated after only 10,000 iterations. Pairwise statistical comparisons were made using the Wilcoxon signed-rank tests . All statistical evaluations were completed using R.

The focus of this analysis was two-fold; (1) comparing the performance of our previous UCS-based core algorithm to the core ExSTraCS algorithm, where 'core' refers to the learning cycle without EK, AT/AF or rule compaction activated, and (2) comparing core ExSTraCS to performance when these separate mechanisms are activated, in order to demonstrate their combined value. These comparisons are performed using the discrete attribute simulation study summarized in Table 1 over a set of key performance metrics. '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' 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. Previously, we demonstrated that UCS yielded the most promising performance on these types of simulated datsets when compared to XCS, MCS, GALE and GAssist (LCS algorithms) [5, 6]. Therefore we utilize the 'core' version of UCS used in [7, 9, 10] as the standard of comparison for ExSTraCS. Notice that in Table 1 the 'core' ExSTraCS p-values are from a comparison to 'core' UCS, while all other p-values correspond to comparisons between 'core' ExSTraCS and ExSTraCS with respective mechanisms activated. As expected, the mixed-ALKR knowledge representation added to ExSTraCS significantly and consistently reduces run time by over 30% on average, when comparing 'core' UCS to 'core' ExSTraCS. We expect this difference to be even more dramatic in datasets with > 20 attributes. Interestingly, a significant increase in testing accuracy is also observed. Next we compare ExSTraCS performance when activating major new mechanisms including (1) EK, (2) AF, (3) EK + AF, and (4) EK + AF + QRF. Performance improvements from EK and AF alone were consistent with those observed in [7, 9]. Further performance improvements were observed when combining mechanisms. As would be expected when comparing UCS with EK and AF active to ExSTraCS with both, (not shown), significant differences were similar to those observed for 'core' comparisons (i.e. ExSTraCS yielded faster run times and higher testing accuracy).

Follow up analysis evaluated ExSTraCS with EK,AF, and QRF on the continuous attribute simulation study. In short, we found that the adopted representation extends ExSTraCS to accommodate continuous attributes. Notably, the run time for 200,000 learning iterations was significantly increased by about 30%, and performance (in terms of testing accuracy and the three power metrics) while promising was significantly lower than for the discrete attribute datasets. This is likely because for continuous attributes, ExSTraCS must learn not only

Table 1. Average performance over all 960 discrete-valued datasets.

10,000 Iterations (Early Performance)											
Performance Statistics	UCS	ExSTraCS									
	Core	Core	p	EK	p	AF	p	EK-AF	p		
Training Accuracy	.8569	.8640	↑ **	.8628	↓ **	X.8634	X-	.8630	↓ **		
Test Accuracy	.5720	.5724	-	.5888	↑ **	X.5713	X-	.5898	↑ **		
Both Power	.0990	.0927	-	.2729	↑ **	X.0927	X-	.2708	↑ **		
Single Power	.4854	.4917	-	.7500	↑ **	X.4354	X↓ *	.7542	↑ **		
Co-Occur. Power	.1083	.0969	↓ *	.0896	-	X.1021	X-	.0865	-		
Generality	.6234	.6233	-	.6227	↓ **	X.6266	X↓ *	.6212	↓ **		
Macro Population	1754.3	1754.6	-	1740.5	↓ **	X1754.6	X-	1738.7	↓ **		
Run Time (min)	3.70	2.57	↓ **	2.53	↓ *	X2.64	X↑ *	2.59	↓ *		
200,000 Iterations (Ending Performance)											
Performance Statistics	UCS	ExSTraCS									
	Core	Core	p	EK	p	AF	p	EK-AF	p	+QRF	p
Training Accuracy	.8641	.8800	↑ **	.8801	-	X.8608	X↓ **	.8634	↓ **	.8537	↓ **
Test Accuracy	.5833	.5863	↑ **	.5866	-	X.5925	X↑ **	.5946	↑ **	.5965	↑ **
Both Power	.2583	.2563	-	.2625	-	X.2948	X↑ **	.2948	↑ **	.3000	↑ **
Single Power	.6146	.6156	-	.6250	-	X.5792	X↓ **	.5958	↓ *	.6062	-
Co-Occur. Power	.1750	.1656	-	.1750	-	X.2031	X↑ **	.1823	↑ *	.1875	↑ *
Rule Generality	.6946	.6945	-	.6945	-	X.7566	X↑ **	.7518	↑ **	.7601	↑ **
Macro Population	1627.3	1627.6	-	1627.7	-	X1428.1	X↓ **	1435.4	↓ **	1044.3	↓ **
Run Time (min)	73.42	49.49	↓ **	49.14	-	X44.02	X↓ **	44.26	↓ **	44.31	↓ **

– No significant change

* $p < 0.05$ (Direction of change given by arrows)

** $p < 6.94x10^{-4}$ (Cutoff assumes Bonferroni multiple test correction based on 72 comparisons)

'which' attributes to specify, but appropriate interval ranges as well. Addressing these shortcomings will be a target for ongoing research.

4 Conclusions

While ExSTraCS has been developed with biomedical, epidemiological, bioinformatics, and genetics problem domains in particular, we expect this new algorithm to be translatable to many related domains, and hopefully inspire new mechanisms and improvements based on this core architecture. Through extensive simulation studies we have demonstrated the value of bringing successful mechanisms together in ExSTraCS in order to improve the key objectives of a successful data mining algorithm including speed, learning efficiency, flexibility, ease of use, scalability, and interpretability. In addition to improving continuous attribute performance future work will address (1) expanding ExSTraCS to also accommodate continuous endpoints (e.g. quantitative traits), (2) further scalability (3) reassessment of fitness and deletion metrics to improve learning efficiency and (4) accessibility and usability through the development of an ExSTraCS GUI, and adaptive run parameters.

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

References

1. Urbanowicz, R.J., Moore, J.H.: Learning classifier systems: a complete introduction, review, and roadmap. Journal of Artificial Evolution and Applications (2009)

2. Thornton-Wells, T., Moore, J., Haines, J.: Genetics, statistics and human disease: analytical retooling for complexity. TRENDS in Genetics **20**(12) (2004) 640–647
3. Wilson, S.: Classifier fitness based on accuracy. Evo. Comp. **3**(2) (1995) 149–175
4. Bernadó-Mansilla, E., Garrell-Guiu, J.: Accuracy-based LCS: models, analysis and applications to classification tasks. Evo. Comp. **11**(3) (2003) 209–238
5. Urbanowicz, R., Moore, J.: 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, ACM (2010) 195–202
6. Urbanowicz, R., Moore, J.: The application of pittsburgh-style lcs to address genetic heterogeneity and epistasis in association studies. Parallel Problem Solving from Nature–PPSN XI (2011) 404–413
7. Urbanowicz, R., Granizo-Mackenzie, A., Moore, J.: 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, ACM (2012) 927–934
8. Urbanowicz, R.J., Andrew, A.S., Karagas, M.R., Moore, J.H.: Role of genetic heterogeneity and epistasis in bladder cancer susceptibility and outcome: a LCS approach. Journal of the American Medical Informatics Association (2013)
9. Urbanowicz, R.J., Granizo-Mackenzie, D., Moore, J.H.: Using expert knowledge to guide covering and mutation in a michigan style LCS to detect epistasis and heterogeneity. In: Parallel Problem Solving from Nature-XII. Springer (2012) 266–275
10. Tan, J., Moore, J., Urbanowicz, R.: Rapid rule compaction strategies for global knowledge discovery in a supervised learning classifier system. In: Advances in Artificial Life, ECAL. Volume 12. (2013) 110–117
11. Bacardit, J., Krasnogor, N.: A mixed discrete-continuous attribute list representation for large scale classification domains. In: Proceedings of the 11th Annual conference on Genetic and evolutionary computation, ACM (2009) 1155–1162
12. Urbanowicz, R.J., Granizo-Mackenzie, A., Moore, J.H.: An analysis pipeline with statistical and visualization-guided knowledge discovery for michigan-style learning classifier systems. Computational Intelligence Magazine, IEEE **7**(4) (2012) 35–45
13. DeJong, K.A., Spears, W.M.: Learning concept classification rules using genetic algorithms. Technical report, DTIC Document (1990)
14. Greene, C., Penrod, N., Kiralis, J., Moore, J.: Spatially uniform relieff (surf) for computationally-efficient filtering of gene-gene interactions. BioData mining **2**(1) (2009) 1–9
15. Kononenko, I.: Estimating attributes: analysis and extensions of relief. In: Machine Learning: ECML-94, Springer (1994) 171–182
16. Greene, C.S., Himmelstein, D.S., Kiralis, J., Moore, J.H.: The informative extremes: using both nearest and farthest individuals can improve relief algorithms in the domain of human genetics. In: Evolutionary Computation, Machine Learning and Data Mining in Bioinformatics. Springer (2010) 182–193
17. Granizo-Mackenzie, D., Moore, J.H.: Multiple threshold spatially uniform relieff for the genetic analysis of complex human diseases. In: Evolutionary Computation, Machine Learning and Data Mining in Bioinformatics. Springer (2013) 1–10
18. Urbanowicz, R.J., Kiralis, J., Sinnott-Armstrong, N.A., Heberling, T., Fisher, J.M., Moore, J.H.: Gametes: a fast, direct algorithm for generating pure, strict, epistatic models with random architectures. BioData mining **5**(1) (2012) 16