Random Artificial Incorporation of Noise in a Learning Classifier System Environment

Ryan J. Urbanowicz
Dartmouth College
1 Medical Center Dr.
Hanover, NH 03755,USA
ryan.j.urbanowicz@dartmouth.edu

Nicholas A. Sinnott-Armstrong
Dartmouth College
1 Medical Center Dr.
Hanover, NH 03755,USA
nicholas.a.sinnott-armstrong@dartmouth.edu

Jason H. Moore
Dartmouth College
1 Medical Center Dr.
Hanover, NH 03755,USA
jason.h.moore@dartmouth.edu

ABSTRACT

Effective rule generalization in learning classifier systems (LCSs) has long since been an important consideration. In noisy problem domains, where attributes do not precisely determine class, overemphasis on accuracy without sufficient generalization leads to over-fitting of the training data, and a large discrepancy between training and testing accuracies. This issue is of particular concern within noisy bioinformatic problems such as complex disease, gene association studies. In an effort to promote effective generalization we introduce and explore a simple strategy which seeks to discourage over-fitting via the probabilistic incorporation of random noise within training instances. We evaluate a variety of noise models and magnitudes which either specify an equal probability of noise per attribute, or target higher noise probability to the attributes which tend to be more frequently generalized. Our results suggest that targeted noise incorporation can reduce training accuracy without eroding testing accuracy. In addition, we observe a slight improvement in our power estimates (i.e. ability to detect the true underlying model(s)).

Categories and Subject Descriptors

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

General Terms

Algorithms, Performance, Design

Keywords

Learning Classifier System, Generalization, Genetics-Based Machine Learning, Genetic Algorithm, UCS, Gene Association Study

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

1. INTRODUCTION

Learning classifier systems (LCSs) have been applied to a number of bioinformatic problems with the goal of either classification or mining attribute relationships within the data. Examples of this include BOOLE++ [16], EpiCS [17], EpiXCS [18], BioHEL [5], LCSE [15] and other attempts at medical data mining [32, 7, 1]. Recently, both Michigan and Pittsburgh LCS were applied to the detection and modeling of heterogeneous and epistatic, genetic disease associations [27, 28]. Genetic heterogeneity (GH) occurs when the same genetic disorder or phenotype is caused by any one of a multiple number genetic mechanisms or pathways (involving one or more alleles or loci). *Epistasis* refers to the interaction between multiple genetic loci wherein the effect of one or more loci is masked by one or more others. These evaluations indicated that a LCS's ability to distribute what it learns over a population of rules, as opposed to seeking a solution comprised of a single best rule, offered a promising strategy to address the problem of genetic heterogeneity.

A fairly ubiquitous concern for LCSs is that of effective generalization. The generalization property of LCSs has been referred to as the "feature which uniquely distinguishes it from classical reinforcement learning systems" [24]. Classifiers traditionally express generalizations using the "don't care" symbol (#) in their conditions. When used in place of a specified attribute value, this symbol indicates that the attribute has been deemed unnecessary for predicting class or action within that classifier. Over the years, a number of attempts have been made to understand and encourage effective generalization. Within the framework of XCS, the most successful and popular LCS to date, Wilson [30] discussed his generalization hypothesis which identifies an intrinsic pressure which implicitly improves rule fitness based on generality. Given two classifiers having the same action, where the condition of one is a generalization of the other, the more general classifier will tend to be included in more match sets. Given that the genetic algorithm (GA) occurs in match sets, this would afford the more general classifier greater reproductive opportunity, likely leading to a higher numerosity and higher fitness, and eventually displacing the less general classifier. This concept was extended by Kovacs [20] in the optimality hypothesis for XCS, which suggests that a maximally general classifier for each payoff level will eventually evolve, which will have a greater numerosity than any other in its payoff level. This hypothesis goes on to suggest that the distribution of classifier numerosities may be

used to identify a subset of rules from the evolved population which represent an optimal solution. [20] and [21] highlight how ideally, maximally general classifiers should be the target of LCS learning while over-general classifiers (observed in [29]) and sub-optimally general (unnecessarily specific) classifiers should be avoided.

Modifications geared towards generalization in XCS were introduced in [33]. These included moving the application of the GA to the action set (as opposed to match set), and the inclusion of a subsumption mechanism. The evolutionary pressures these mechanisms drove were later described and formalized as set and subsumption pressures, respectively [11, 10]. In particular, two subsumption mechanisms were introduced, "GA subsumption deletion" which hinders the insertion of more specific classifiers once an accurate, more general one evolved, and "action set subsumption" in which a more general classifier actually absorbs all more specific classifiers regardless of if it already existed or was just generated.

In the context of GAssist (a Pittsburgh-style LCS), the principle of *Occam's razor* [23] inspired the implementation of a hierarchical selection operator [2] and the Minimum Description Length (MDL) [3] in order to promote well generalized solutions. This principle suggests that "the simplest explanation of the observed phenomena is most likely to be the correct one". Additionally, a windowing heuristic was implemented, encouraging further generalization pressure [4].

In all of these examples, the emphasis has been placed on applying pressure towards the evolution of accurate, maximally general classifiers. However this ideal balance of accuracy and generality is sometimes difficult to encourage, especially in the context of "real" data that is typically quite noisy. In comparing XCS with GAssist, the tendency for XCS to over-fit training data, (especially in smaller datasets) was observed [1]. Over-fitting often indicates the the system is learning structure that is idiosyncratic to the training set and therefore generalization is occurring sub-optimally. As utilized in [14], cross-validation offers one strategy to determine the extent of over-fitting. In addition [14] and [15] introduced LCSE, an ensemble based system demonstrated to achieve better generalization than a single LCS run, with the drawback of computational expense.

Generally speaking classification and data mining problems can exhibit two types of noise. The first, which we will refer to as classification noise, makes it impossible to develop a model or rule-set which classifies with perfect accuracy on testing data, even when all predictive attributes are accurately considered. the second, which we will refer to as attribute noise, is the inclusion of non-predictive attributes within the problem domain (i.e. attributes that have no relationship with class). For example, attribute noise is a common and important issue when dealing with bioinformatic problems, wherein the distinction between predictive and non-predictive attributes is often the goal.

A handful of LCS studies have explored the robustness of these systems to the incorporation of classification noise. In [12, 9, 13], alternating noise and noise sampled from a Gaussian distribution was added to the payoff function of multiplexor problems. [22] generated noise to LED datasets by swapping attribute values. [6] also generated classification noise in a multiplexor problem by flipping the class label with a certain probability. These problems however only include attributes which can be characterized as predictive (no attribute noise). Differently, in the present study we consider a problem that naturally possesses a great deal of both classification and attribute noise. In particular we are interested in dealing with attribute noise and encouraging efficient generalization such that rules learned by the LCS will avoid specifying noisy attributes which contribute nothing to classification.

With this goal in mind, we explore a simple, computationally inexpensive heuristic which seeks to discourage overfitting via the probabilistic incorporation of noise. This random artificial incorporation of noise, or RAIN, is designed to indirectly address the inherent noise and complexity of real data. LCSs with accuracy-based fitness measures often strive to achieve a complete-action map [19] with a respective training accuracy of 100%. In real data (e.g. disease-gene association), where the solution is only useful if it can be generalized to unseen data (e.g. testing data), achievement of this level of training accuracy is meaningless. RAIN randomly mutates probabilistically selected attributes at each learning iteration, such that each learning epoch (i.e. cycle through the entire dataset), the LCS is exposed to a randomly varied version of the original dataset. We evaluate several implementations of RAIN within an implementation of UCS described in [27]. This evaluation considers different magnitudes of noise, different models which temporally vary noise magnitude, and the application of noise which is variably targeted to different attributes in the dataset.

2. METHODS

In this section we describe (1) the LCS framework implemented to test RAIN, (2) the various implementations of RAIN and (3) our experimental evaluation of the proposed strategy.

2.1 LCS Framework

Learning classifier systems (LCS), combine machine learning with evolutionary computing and other heuristics to produce an adaptive system that learns to solve a particular problem. LCSs are closely related to and typically assimilate the same components as the more widely utilized genetic algorithm (GA). The goal of LCS is not to identify a single best model or solution, but to create a cooperative set of rules or models which together solve the task. The solution evolved by an LCS is represented as a population of rules/models which are utilized collectively to make decisions/classifications. Michigan-style LCSs, often varying widely from version to version, generally possess four basic components; (1) a population of rules or classifiers, (2) a performance component that assesses how well the population of rules collectively explain the data, (3) a reinforcement component that distributes the rewards for correct prediction to each of the rules in the population, and (4) a discovery component that uses different operators to discover new rules and improve existing ones. Learning progresses iteratively, relying on the performance and reinforcement components to drive the discovery of better rules. For a complete LCS introduction and review, see [26].

[27] previously implemented a version of UCS designed to accommodate a disease-gene association problem domain. Similarly, this study tests RAIN in the context of data mining which concurrently examines epistasis and GH, modeled

as they might simultaneously occur in a single nucleotide polymorphism (SNP) genetic association study. This problem domain is particularly challenging and the findings of [27, 28] suggest that LCSs (particularly UCS) may be adapted at effectively address it. UCS, or the sUpervised Classifier System [8], has many similarities to the architecture of XCS [30] but differs in that it replaces reinforcement learning with supervised learning, encouraging the formation of best action maps (rule sets of efficient generalizations) and alters the way in which accuracy (and thus fitness) is computed. UCS was designed specifically to address single step problem domains such as classification and data mining, where delayed reward is not a concern. The implementation of UCS used in this study was the same as used in [27] with some minor control modifications leaving the original learning algorithm unchanged.

2.2 RAIN Implementation

As previously described, Michigan LCSs learn iteratively from the environment (i.e. one sample instance per learning iteration). At the beginning of a supervised learning iteration, the state and class of a sample is obtained from the environment. Our incorporation of random artificial noise takes place here, before rule matching takes place. We will begin by describing our strategy for RAIN in which each attribute has, within a given iteration, an equal current probability (P_c) of being permuted. Initially, we specify a maximum probability of attribute permutation (P_m). We considered four different temporal models in which P_c varies over the course of the specified maximum number of learning iterations (I_m). The first, is *uniform* as defined by;

$$P_c = P_m \tag{1}$$

The second is *linear* as defined by;

$$P_c = I_c/I_m * P_m \tag{2}$$

where I_c is the current learning iteration. The third is *inverse linear* as defined by;

$$P_c = (1 - I_c/I_m) * P_m \tag{3}$$

The fourth is *gaussian* as defined by;

$$P_c = a * e^{-\frac{x-b^2}{2*c^2}} \tag{4}$$

where $x = \frac{I_c}{I_m}$, the function's peak height $a = 1$, its center $b = 0.5$, and it's width $c = 0.2$. Illustrations of these models are given in Figure 1. The permutation of any attribute(s) within the current dataset instance are performed non-destructively, such that the dataset is not permanently altered.

In addition to the above models, we also implement RAIN such that it is variably targeted to different attributes in the dataset. For this implementation of 'targeted' RAIN we only consider the 'uniform' temporal model, such that $P_c = P_m$ throughout. However, future implementations could easily combine targeted noise with the other temporal dynamics explored above. With targeted RAIN, the individual probability of permutation for each attribute in the dataset varies according to a list of attribute weights. The premise of this alternate strategy is to strategically, and automatically avoid destructively adding noise to attributes likely to be important to classification and instead target noise towards attributes are more likely to be noise themselves. We implement and test two different targeted strategies; *Targeted Generality* (TG) and *Targeted Fitness Weighted Generality* (TFWG). First we will describe how these lists of weights are obtained, and updated every learning epoch.

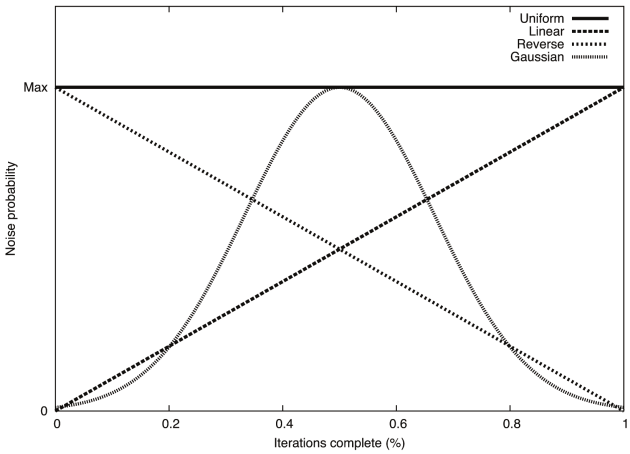

Figure 1: A simple visualization of the temporal noise incorporation patterns implemented in this study.

Power, or the success rate, is typically estimated in these types studies by tracking the frequency with which an algorithm successfully identifies the correct underlying model or attribute(s), across some number of data set replicates. This type of estimation is not applicable for LCSs, which evolve a solution made up of an entire population of rules. In [27] a strategy for estimating power in LCSs was outlined. Power is estimated by counting the number of times an attribute is specified across all rules comprising the population, as opposed to how often a '#' is used. This value is weighted by the numerosity of each rule, and is referred to as the attribute generality. Specifically, success is achieved if the total number of rules specifying '#' for each of the predictive attributes is lower than for any other noise attribute. This was used as an indicator of which attributes were important to making accurate classification. This strategy hypothesized that if a LCS is truly learning the underlying model(s) in a dataset, attributes that are important to the underlying model(s)(predictive attributes) will tend to be specified more frequently within rules of the population. Conversely, attributes that are not involved in the underlying model(s) (noise attributes) will tend to be generalized ('#'/don't-care symbol used) more frequently. Similarly, here we use this attribute generality to direct a higher probability of noise to attributes which the LCS has learned to be less important to classification, and less (or no) noise towards those that are specified more frequently (and are presumably important for making accurate classifications). In TG, our weight list is simply this list of attribute generalities. In TFWG the attribute generalities are weighted by both the numerosity and fitness of each respective rule in the population. During the very first epoch, no noise is added. Only after the algorithm has seen the whole training set does targeted RAIN turn on. The process of calculating and updating the targeted weight list occurs at the end of each epoch (i.e. after each cycle through the training data).

Next we describe how this list of attribute weights is utilized by RAIN to probabilistically determine which attribute

will be permuted. This description is the same for both TG and TFWG. As with our temporal models we specify P_m. This value is used to randomly determine how many attributes will be permuted in the present dataset instance. Next, we find the minimum weight in the weight list and subtract that value from all weights in the list. This ensures that the attribute most often specified in the rule population has a zero probability of being permuted. Each time an attribute is to be permuted, roulette wheel selection is employed to choose that attribute based on the adjusted weights of the list.

2.3 Experimental Evaluation

To evaluate RAIN we used a subset of the simulated data sets described in [27, 28] which concurrently model GH and epistasis as they might appear in a SNP gene association study of common complex disease. All data sets were generated using a pair of distinct, two-locus epistatic interaction models, both utilized to generate instances (i.e. case and control individuals) within a respective subset of each final data set. Each two-locus epistatic model was simulated without Mendelian/main effects, as a penetrance table using [25]. The simulated models in this study each possessed a heritability of 0.2, minor allele frequencies of 0.2, and an architectural "difficulty" designation of "moderate" [27]. Balanced datasets (i.e. datasets with an equal number of cases and controls) simulated from these models were generated as having four different sample sizes (200, 400, 800, 1600) and a heterogeneous mix ratio of 50:50. Additionally 10-fold cross validation (CV) was employed to measure average testing accuracy and account for over-fitting. Together a total of 80 simulated datasets x 10 fold CV (or 800 runs of the algorithm) were accomplished for each of the 24 experimental configurations of RAIN and a control run. These datasets were selected based on the results of [27] such that our control evaluation (without RAIN) would demonstrate varying success.

In this study we specified a maximum micro population size of 1600 and allowed the algorithm to run up to 500000 learning iterations with interval evaluations at 50000, 100000, and 200000. All other UCS run parameters were left at their default settings as in [27]. For each run we track the following characteristics; training accuracy, testing accuracy, generality, macro population size, the power to find both underlying models, the power to find at least one underlying model, and run time. Here, generality is the average proportion of "don't care" symbols within the entire rule population (taking numerosity into account). Power is a reflection of our ability to mine knowledge from the evolved rule population. Each of these values represent an average over the 10 CV runs. For the four temporal models and two targeted implementations of RAIN we ran evaluations at P_m values of 0.001, 0.01, 0.05, and 0.1.

Statistical comparisons between implementations of RAIN and our control runs were made using the non-parametric Kruskal-Wallis paired with the Mann-Whitney test due to a lack of normality in the value distributions. All statistical evaluations were completed using R. Comparisons were considered to be significant at $P \leq 0.05$.

3. RESULTS

Figure 2 summarizes the average run characteristics for all experimental configurations after 500,000 iterations. Additionally, Figure 3 illustrates the distribution of macro population sizes and Figure 4 illustrates the distribution of generality for these same experimental configurations after 500,000 iterations. A clear initial observation is the poor performance of the first four temporal RAIN models we examined (i.e. uniform, linear, inverse linear, and gaussian) each of which permute attributes with equal probability. While at $P_m = 0.1$, both uniform and linear models significantly reduced training accuracy, this was accompanied by a significant reduction in testing accuracy. A significant reduction in generality was observed for intermediate values of P_m for all temporal models except inverse linear. At $P_m = 0.1$ we observed a dramatic reduction in the algorithm's ability to detect the attributes of both underlying models. While this difference was not statistically significant, power comparisons for all temporal models suggested that adding noise with equal probability to all attributes in the dataset tended to harm our ability to identify the attributes involved in the underlying genetic models.

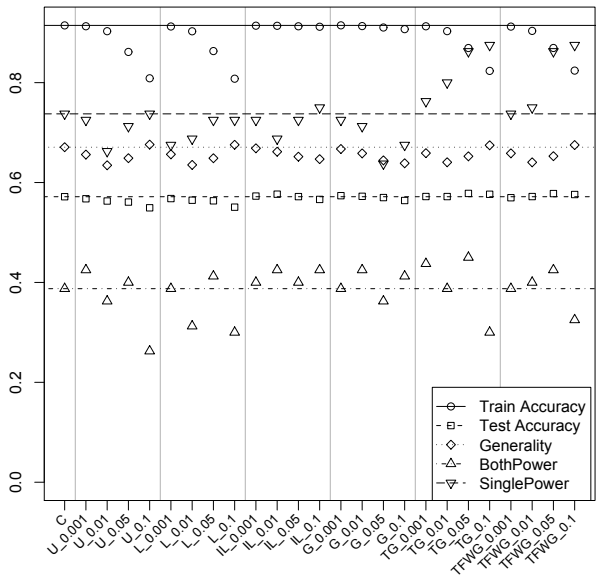

Figure 2: Comparing the RAIN test configurations after 500,000 learning iterations. Each data point represents an average over 80 datasets, and a total of 800 UCS runs (taking CV into consideration). Grey vertical lines delineate our different RAIN implementations where "C" stands for control, "U" for uniform, "L" for linear, "IL" for inverse linear, "G" for gaussian, and 'TG' and 'TFWG' are the two targeted implementations described. The horizontal lines are drawn through the average control group values for quick comparison to RAIN tests.

These trends were generally consistent across all interval evaluations (data not shown). Notably however, as noise diminished over time in the inverse linear and gaussian models, training accuracy and macro population size recovered. In other word the application of high P_m during earlier iterations, diminishing to negligible P_m by the final itera-

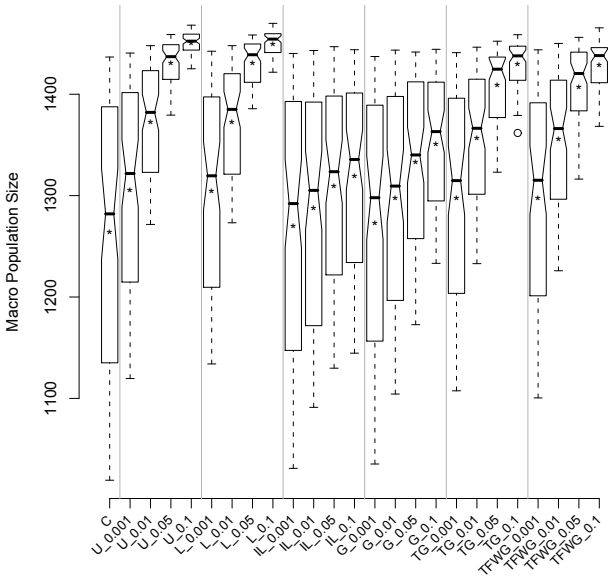

Figure 3: Comparing macro population size distributions after 500,000 learning iterations. Each box plot includes 80 training accuracy averages taken over 4 different data set samples sizes. The star within each box plot indicates the average of the 80 values. RAIN configurations on the x-axis are the same as in Figure 2.

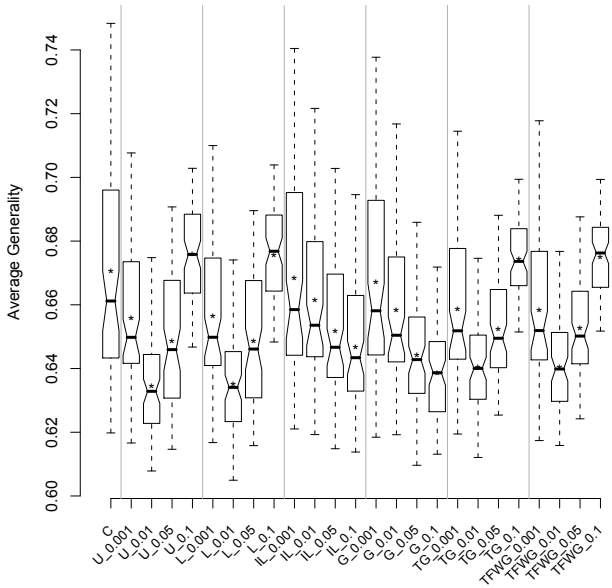

Figure 4: Comparing generality distributions after 500,000 learning iterations. Each box plot includes 80 training accuracy averages taken over 4 different data set samples sizes. The star within each box plot indicates the average of the 80 values. RAIN configurations on the x-axis are the same as in Figure 2.

tion yielded no significant impact on overall performance. This finding is consistent with how learning classifier systems adaptively evolves it's rule population to accommodate changing environments.

These findings led us to the development of our targeted RAIN strategies (i.e. TG and TFWG) in an attempt to implement noise such that it probabilistically targeted attributes less likely to be important to classification, and therefore avoid damage to power or testing accuracy. The results of our two targeted RAIN strategies were encouraging, especially at $P_m > 0.01$. At these noise magnitudes, both TG and TFWG yielded significantly lower training accuracies while preserving testing accuracy and increasing the power to find one or both of the underlying models (although not statistically significant). Additionally, generality recovered at these higher P_m values, while at $P_m = 0.01$ it was significantly lower. Also, dramatically illustrated in Figure 3, the macro population size is significantly increased in TG and TFWG. Figures 3 and 4 also suggest that, in general, higher magnitudes of P_m have a tendency to consistently increase macro population size, while also decreasing the variability observed in these values across our experimental runs. No significant differences between TG and TFWG were observed. Overall, high P_m values consistently required a small but statistically significant run time (the worst case took approximately 15% more time). These results were largely consistent at all iteration checkpoints.

4. CONCLUSIONS AND FUTURE WORK

In the present study we explore the implementation of the random artificial incorporation of noise (RAIN) in a learning classifier system environment in an effort to improve efficient generalization and deter over-fitting in a complex genetics problem domain. Our results indicate that while the incorporation of RAIN with equal attribute probability is ineffective, attribute targeting of RAIN was able to reduce over-fitting, evidenced by a significant decrease in training accuracy without reducing testing accuracy. Additionally, improvements in power (successful attribute identification) offer evidence that RAIN may improve the mining of predictive attributes in LCSs. We also observed increases in macro population size with the addition of higher magnitudes of noise. Intuitively, since noise introduces more variety and less regularity during training, an increased diversity of classifiers in the population is needed to cover that ever changing set. Further exploration is required to explore the drop in generality observed with intermediate noise magnitudes, and the recovery of that generality (when compared to the control), when higher noise magnitudes were applied. In addition, our future work will explore a broader range of P_m values, implement a hybrid version of rain with combines attribute targeting with a temporal model, and implement RAIN with adaptive P_m values so that the frequency of noise incorporation may adjust itself according to the systems performance over time.

Acknowledgement

We thank Casey Greene for his input on the development of this concept. This work was supported by NIH grants AI59694, LM009012, and LM010098.

5. REFERENCES

[1] J. Bacardit and M. Butz. Data mining in learning classifier systems: comparing XCS with GAssist. *Urbana*, 51:61801, 2004.

[2] J. Bacardit and J. Garrell. Métodos de generalización para sistemas clasificadores de Pittsburgh. In *Proceedings of the Primer Congreso Espanol de Algoritmos Evolutivos y Bioinspirados (AEB'02)*, 486–493, 2002.

[3] J. Bacardit and J. Garrell. Bloat control and generalization pressure using the minimum description length principle for a pittsburgh approach LCS. *Learning Classifier Systems*, pages 59–79, 2007.

[4] J. Bacardit, D. Goldberg, M. Butz, X. Llora, and J. Garrell. Speeding-up pittsburgh learning classifier systems: Modeling time and accuracy. In *Parallel Problem Solving from Nature-PPSN VIII*, 1021–1031. Springer, 2004.

[5] J. Bacardit and N. Krasnogor. Biohel: Bioinformatics-oriented hierarchical evolutionary learning. *Computer Science and IT*, 2006.

[6] J. Bacardit and N. Krasnogor. Performance and efficiency of memetic pittsburgh LCSs. *Evolutionary computation*, 17(3):307–342, 2009.

[7] E. Bernadó, X. Llora, and J. Garrell. XCS and GALE: A comparative study of two learning classifier systems on data mining. *Advances in learning classifier systems*, 115–132, 2002.

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

[9] M. Butz. *Rule-based evolutionary online learning systems: A principled approach to LCS analysis and design*. Springer Verlag, 2006.

[10] M. Butz, T. Kovacs, P. Lanzi, and S. Wilson. Toward a theory of generalization and learning in XCS. *Evolutionary Computation, IEEE Transactions on*, 8(1):28–46, 2004.

[11] M. Butz and M. Pelikan. Analyzing the evolutionary pressures in XCS. In *Proceedings of the Third Genetic and Evolutionary Computation Conference (GECCO-2001)*, volume 935, page 942, 2001.

[12] M. Butz, K. Sastry, and D. Goldberg. Tournament selection: Stable fitness pressure in XCS. In *Genetic and Evolutionary Computation GECCO 2003*, 215–216. Springer, 2003.

[13] M. Butz, K. Sastry, and D. Goldberg. Strong, stable, and reliable fitness pressure in XCS due to tournament selection. *Genetic Programming and Evolvable Machines*, 6(1):53–77, 2005.

[14] Y. Gao, J. Huang, H. Rong, and D. Gu. Learning classifier system ensemble for data mining. In *Proceedings of the 2005 workshops on Genetic and evolutionary computation*, pages 63–66. ACM, 2005.

[15] Y. Gao, J. Huang, H. Rong, and D. Gu. LCSE: learning classifier system ensemble for incremental medical instances. *Lecture Notes in Computer Science*, 4399:93, 2007.

[16] J. Holmes. A genetics-based machine learning approach to knowledge discovery in clinical data. In *Proceedings of the AMIA Annual Fall Symposium*, page 883, 1996.

[17] J. Holmes. Discovering risk of disease with a learning classifier system. In *Proceedings of the Seventh International Conference of Genetic Algorithms (ICGA97)*, pages 426–433. Citeseer, 1997.

[18] J. Holmes and J. Sager. Rule discovery in epidemiologic surveillance data using EpiXCS: an evolutionary computation approach. *Lecture Notes in Computer Science*, 3581:444, 2005.

[19] T. Kovacs. *A comparison of strength and accuracy-based fitness in LCSs*. PhD thesis, PhD thesis, University of Birmingham, 2002.

[20] T. Kovacs. XCS classifier system reliably evolves accurate, complete and minimal representations for Boolean functions. *Cognitive Science Research Papers, University of Birmingham CSRP*, 1997.

[21] T. Kovacs. Strength or accuracy? Fitness calculation in learning classifier systems. *Learning Classifier Systems*, pages 143–160, 2000.

[22] X. Llorà and D. Goldberg. Bounding the effect of noise in multiobjective learning classifier systems. *Evolutionary Computation*, 11(3):279–298, 2003.

[23] T. Mitchell. Machine learning. 1997. *Burr Ridge, IL: McGraw Hill*.

[24] O. Sigaud and S. Wilson. LCSs: a survey. *Soft Computing-A Fusion of Foundations, Methodologies and Applications*, 11(11):1065–1078, 2007.

[25] R. Urbanowicz, J. Kiralis, Sinnott-Armstrong, T. N. Heberling, J. Fisher, and J. Moore. A Fast, Direct Algorithm for Generating Pure, Strict, Epistatic Models with Random Architectures. In preparation.

[26] R. Urbanowicz and J. Moore. Learning Classifier Systems: A Complete Introduction, Review, and Roadmap. *Journal of Artificial Evolution and Applications*, 2009, 2009.

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

[28] R. Urbanowicz and J. Moore. The application of pittsburgh-style learning classifier systems to address genetic heterogeneity and epistasis in association studies. In *Proceedings of the 11th annual international conference on Parallel Problem Solving in Nature*, 404–413. Springer, 2010.

[29] S. Wilson. ZCS: A zeroth level classifier system. *Evolutionary computation*, 2(1):1–18, 1994.

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

[31] S. Wilson. Mining oblique data with XCS. *Advances in Learning Classifier Systems*, 283–290, 2001.

[32] S. Wilson, S. Wilson, G. Xcs, et al. Generalization in the XCS classifier system. 1998.