Post-processing clustering to reduce XCS variability

Flavio Baronti
baronti@di.unipi.it

Alessandro Passaro
passaro@di.unipi.it

Antonina Starita
starita@di.unipi.it

Dipartimento di Informatica
Università di Pisa
Largo B. Pontecorvo, 3
56127 Pisa, Italy

ABSTRACT

XCS is a stochastic algorithm, so it does not guarantee to produce the same results when run with the same input. When interpretability matters, obtaining a single, stable result is important. We propose an algorithm to join the rules produced from many XCS runs, based on a measure of distance between rules. We also suggest a general definition for such a measure, and show the results obtained on a complex data set.

Categories and Subject Descriptors

I.2.6 [**Artificial Intelligence**]: Learning—*Learning classifier systems*

General Terms

Algorithms

Keywords

Learning classifier systems, clustering

1. INTRODUCTION

Randomness of the search process is one of the chief characteristics of evolutionary algorithms, and indeed one of its strong points. It is randomness which allows them to escape local optima, and ensures a broader portion of the search space to be explored.

This non-deterministic behaviour is however a double-edged weapon; in fact, for non-trivial problems, it is likely that many repetitions of the algorithm will produce many different final solutions. In some cases joining these results can be easy, either picking the best one, or by merging them together. When this is not possible, a viable alternative is to keep all of them, and set up a voting mechanism to take every one into account. Sometimes however joining the results can be problematic, and voting undesirable (for instance, if we want to maintain interpretability).

GECCO'05, June 25–29, 2005, Washington, DC, USA.

XCS [7] too suffers from this problem; even after employing a ruleset reduction algorithm (like CRA [8]), on non-trivial problems many runs will produce different rulesets, each one with its own rules, with similar performance but no evident way to build (or to choose) a single merged set. The problem is exacerbated by the fact that the same rule can come in many slightly variated forms, from whence the need to define a measure of similarity between rules.

We present a post-processing algorithm which tries to solve this problem, or at least mitigate it. The basic assumption is that good rules will be preferred by XCS, and will then appear more often in the output sets, although with slight variations. We repeat an XCS experiment a number of times; then a clustering algorithm is performed on all the resulting rules, putting together similar ones. Bigger clusters contain more frequent rules, so a representative from each of the biggest clusters is chosen; finally, a reduced version of XCS is executed again on this set of rules, in order to train them together and to set their working parameters (accuracy, fitness, etc.).

2. PROBLEM DEFINITION

In the following, we will make use of a measure of distance $\mathcal{D}$, and will assume a corresponding measure of similarity $\mathcal{S}$ can be defined (which decreases as distance increases). The measure $\mathcal{S}$ must lie in the range from 0 (maximally different items) to 1 (equal items); if no assumptions can be made on the source measure of distance, one possible definition for $\mathcal{S}$ given $\mathcal{D}$ can be (for any $\alpha > 0$)

$$\mathcal{S} = (1 + \mathcal{D})^{-\alpha} \tag{1}$$

The basic function required by the algorithm is a measure of distance between rules. As the shape of rules is completely problem-dependent, this measure could be thought to share this characteristic. We suggest however a way to define it which makes it independent, as long as the problem provides a set of n input data to be learned; this is not always the case, as XCS works by reinforcement learning, where the existence of such a set is just a particular situation.

In case this set exists, we define the S-*signature*[1] of a rule r as the set of input patterns the rule applies to. We then define two rules to be similar if they apply mostly to the same inputs (with a maximum when their signatures are equal); they are defined to be diverse instead when they apply to different inputs (with a minimum when the signatures do not have common elements). A suitable measure

[1]This S is for "set"; the $\mathcal{S}$ in Eq. 1 was for "similarity".

of distance can then be the Jaccard coefficient [4]:

$$D(r_1, r_2) = 1 - \frac{|S(r_1) \cap S(r_2)|}{|S(r_1) \cup S(r_2)|} \quad (2)$$

$$= 1 - \frac{|S(r_1) \cap S(r_2)|}{|S(r_1)| + |S(r_2)| - |S(r_1) \cap S(r_2)|} \quad (3)$$

This measure ranges from 0 (maximally different rules) to 1 (equal rules), and has been demonstrated to be a metric [5]. The corresponding similarity function is simply $S(r_1, r_2) = 1 - D(r_1, r_2)$.

An equivalent way to view a rule signature is as a boolean vector. Since the input set size is a fixed value n, the B-signature of a rule r could be represented as a vector $\mathbf{b} \in \{0,1\}^n$, where $\mathbf{b}_i = 1$ iff rule r applies to input pattern i. Jaccard distance becomes then

$$D(r_1, r_2) = 1 - \frac{\sum_{i=1}^{n} B(r_1)_i \wedge B(r_2)_i}{\sum_{i=1}^{n} B(r_1)_i \vee B(r_2)_i} \quad (4)$$

It is important to note that this measure is not strictly related with the intuitive notion of similarity between two rules. In fact, it bears no notion of how the rules actually *appear*: two rules could look completely different, and still apply to the same set of input patterns. This has experimentally been found to be common when rules are "tailored" to pick a very small subset of inputs — for instance, an outlier. In that situation, the dataset probably offers many possible ways to isolate that particular pattern with a number of conditions on its values; this will produce many different-looking rules, which for the system actually have the same meaning. Consider for instance the situation where a single condition is sufficient to isolate a pattern (say *age>90*, in a dataset where only one person is that old). All the other conditions of the classifier can then vary freely, as long as they continue to cover the pattern, maintaining the same signature — that is, the same meaning to the system.

Another advantage of this definition of distance is that it does not require to choose a weighting strategy for attributes. If we had to compute similarity on the rule appearance (that is, on its conditions), we should decide how much importance to give to mismatches in the different attributes. This becomes more challenging when attributes do not have the same type.

As final notice, also within the same attribute choosing a measure of similarity could be difficult. When modelling a real value for instance, it is entirely possible that its distribution is not uniform in the whole range of validity. Then, a little variation where the values are more frequent should be weighted more than a larger variation in areas where values are few. Recalling to the previous example, if the only person above 90 is 95, the two conditions *age>90* and *age>94* appear equal to the system — while *age>50* and *age>54*, although differing of the same amount, probably describe quite different pattern sets.

The last requirement is a method to evaluate how much the original results were different, and how much this difference changed after executing the proposed algorithm. We thus need a measure of similarity between sets of rules — that is, sets of items which have themselves a similarity measure (as opposed to a simple *equals* relationship). We extend Jaccard coefficient to this more general setting.

Starting from equation 3, we need to define the size of the intersection. Notice that maximizing this value will minimize set distance (thus maximizing set similarity). We can put the items of the two sets in a fully connected bipartite graph, where each edge is weighted with the similarity between the two nodes. We then define the size of the intersection as the value of the maximal matching in the bipartite graph. This means that we assign each item from one set at most one item from the other set; the size of intersection is then measured by taking the sum of similarities between the matched items, and by picking the matching which maximizes this sum.

Table 1: Summary of XCS parameters. Naming of the variables follows [1]

N	400	χ	0.8	ϵ_I	10
β	0.15	μ	0.04	F_I	0.01
α	0.1	θ_{del}	10	θ_{mna}	2
ϵ_0	10	δ	0.1	p_{explr}	0.5
ν	5	θ_{sub}	20	p_{GAsub}	1.0
γ	N/A	$P_\#$	0.333	p_{ASsub}	1.0
θ_{ga}	40	p_I	500		

3. THE ALGORITHM

We now describe the algorithm we propose to join the results of many runs of XCS. We recall that the underlying assumption is that more important rules will be discovered and reported by XCS more often. This means in turn that, when joining all the rules generated by many runs, they will be more numerous than less important ones. Clustering all these rules, in order to put similar rules together, should thus yield bigger clusters for more important rules, and smaller clusters for less useful rules.

After each XCS run, a ruleset reduction algorithm (we applied CRA [8]) is performed, in order to pick the most useful rules. Then the boolean signatures $\mathcal{B}(r)$ of all the m rules resulting from the runs will be computed. This produces m boolean vectors, which will be partitioned with a clustering algorithm (like k-means [3], or ROCK [2]) according to the chosen rule distance. Following the basic assumption, the cluster sizes sort rules over their importance; the l biggest clusters will then group the l most important rules. From them, one representative rule must be chosen (if the cluster has a centroid, this could be the rule closest to it). Finally, the l chosen rules should be trained again to work together; this can be done running XCS again with a fixed population made of them, and allowing only the performance, error, and fitness values to vary.

4. TESTING AND CONCLUSIONS

The first fact to verify is to check whether the basic assumption actually holds, and whether picking rules from different sets does not impair performance (XCS evolves rules to work together). The second regards clustering and retraining own variance: will it stack up with XCS' variance, thus producing again unstable results? Or does the nature of the data set to be clustered allow clustering methods to produce stable results?

We tested the whole algorithm on the HNSCC data set, a complex medical data set which produced high-variability results (described in [6]). In order to evaluate the impact on performance of clustering, we applied 10-fold cross-validation to the whole process. More in detail, for each fold XCS with CRA was run 10 times; then, the 10 resulting rule sets were

Table 2: Average results before and after clustering and retraining

	Accuracy	Specificity	Sensitivity	**Distance**
Before	$.78 \pm .02$	$.88 \pm .02$	$.60 \pm .04$	$\mathbf{.86 \pm .05}$
After	$.79 \pm .02$	$.90 \pm .03$	$.58 \pm .07$	$\mathbf{.74 \pm .04}$

clustered, and the representatives for the clusters were then re-trained together. Evaluation on the test set was done before clustering, and after retraining. This process was repeated 10 times, giving a total of 100 XCS runs on each fold; XCS evolved for 250'000 generations (full parameters are reported in table 1).

Results reported in table 2 show that the process did not impair performance, and did lead to a reduction in results variability, altough not as marked as hoped (and still quite far from results stability). Future work will include running the clustering on increasingly bigger groups of results, to see up to which point variability can be decreased, and on different data sets to assess its robustness. Other points to clear up are whether the proposed measure of distance between sets is actually a metric, and testing other measures of distance.

5. ACKNOWLEDGEMENTS

We would like to thank the following people for providing the data set and supporting us during the analysis: A. Abbondandolo, R. Barale, S. Bonatti, F. Canzian, G. Casartelli, V. Maggini, G. Margarino, P. Mereu, A. M. Rossi.

6. REFERENCES

[1] M. V. Butz and S. W. Wilson. An algorithmic description of XCS. In P. L. Lanzi and et al., editors, *IWLCS 2000*, volume 1996 of *LNAI*, pages 253–272. Springer-Verlag, 2001.

[2] S. Guha, R. Rastogi, and K. Shim. ROCK: A robust clustering algorithm for categorical attribute. *Information Systems*, 25(5):345–366, 2000.

[3] Z. Huang. Extensions to the k-means algorithm for clustering large data sets with categorical values. *Data Mining and Knowledge Discovery*, 2(3):283–304, Sep 1998.

[4] A. K. Jain and R. C. Dubes. *Algorithms for Clustering Data*. Prentice Hall, NJ, 1998.

[5] A. H. Lipkus. A proof of the triangle inequality for the tanimoto distance. *Journal of Mathematical Chemistry*, 26(1–3):263–265, March 1999.

[6] A. Passaro, F. Baronti, V. Maggini, A. Micheli, A. M. Rossi, and A. Starita. Exploring relationships between genotype and oral cancer development through XCS. In *Proceedings of MEDGEC 2005*. ACM, 2005.

[7] S. W. Wilson. Classifier fitness based on accuracy. *Ev. Comp.*, 3(2), 1995.

[8] S. W. Wilson. Compact rulesets from XCSI. In P. L. Lanzi and et al., editors, *IWLCS 2001*, volume 2321, pages 197–210. Springer-Verlag, 2001.