

A Learning Classifier Systems Approach to Clustering

Learning Classifier Systems Group Technical Report – UWELCSG06-003

Kreangsak Tamee^{1,2}, Larry Bull¹

¹Faculty of Computing, Engineering & Math. Sciences
University of the West of England
Bristol BS16 1QY, U.K.
kreangsakt@yahoo.com, larry.bull@uwe.ac.uk

Ouen Pinnern²

²Faculty of Engineering
King Mongkut's Institute of Technology
Bangkok, Thailand 10520
kpouen@kmitl.ac.th

Abstract- This paper presents a novel approach to clustering using a simple accuracy-based Learning Classifier System. Our approach achieves this by exploiting the evolutionary computing and reinforcement learning techniques inherent to such systems. The purpose of the work is to develop an approach to learning rules which accurately describe clusters without prior assumptions as to their number within a given dataset. Favourable comparisons to the commonly used k -means algorithm are demonstrated on a number of datasets.

1 Introduction

This paper presents initial results from a rule-based approach to clustering through the development of a Learning Classifier System (LCS)(Holland, 1976). LCS can achieve this by using both reinforcement learning (Sutton & Barto, 1998) and genetic algorithms (GA)(Holland, 1975), and offer the automated rule development of neural networks together with the transparency of production system rules. A number of studies have indicated good performance for LCS in classification tasks (e.g., see (Bull, 2004) for examples). We are interested in the utility of such systems to perform unsupervised learning tasks.

Clustering is an important unsupervised classification technique where a set of data are grouped into clusters in such a way that data in the same cluster are similar in some sense and data in different clusters are dissimilar in the same sense. For this it is necessary to first define a measure of similarity which will establish a rule for assigning data to the domain of a particular cluster centre. One such measure of similarity may be the Euclidean distance D between two data x and y defined by $D=||x-y||$. Typically in data clustering there is no one perfect clustering solution of a dataset, but algorithms that seek to minimize the cluster spread, i.e., the family of centre-based clustering algorithms, are the most widely used (e.g., Xu & Winch, 2005). They each have their own mathematical objective function which defines how well a given clustering solution fits a given dataset. In this paper our system is compared to the most well-known of such approaches, the k -means algorithm. We use as a measure

of the quality of each clustering solution the total of the k -means objective function:

$$o(X, C) = \sum_{i=1}^n \min_{j \in \{1 \dots k\}} \|x_i - c_j\|^2 \quad (1)$$

Define a d -dimensional set of n data points $X = \{x_1, \dots, x_n\}$ as the data to be clustered and k centers $C = \{c_1, \dots, c_k\}$ as the clustering solution. However most clustering algorithms require the user to provide the number of clusters (k), and the user in general has no idea about the number of clusters (e.g., see (Tibshirani et al., 2000)). Hence this typically results in the need to make several clustering trials with different values for k where $k = 2$ to $k_{max} = \text{square-root of } n$ (data points) and select the best clustering among the partitioning with different number of clusters. The commonly applied Davies-Bouldin validity index is used as a guideline to the underlying number of clusters here (Davies & Bouldin, 1979).

2 A Simple Accuracy-based LCS

In this paper we present a version of the simple accuracy-based YCS (Bull, 2005) which is derived from XCS (Wilson, 1995), here termed YCS_c. YCS_c is a Learning Classifier System without internal memory, where the rulebase consists of a number (N) of rules. Associated with each rule is a scalar which indicates the average error (ϵ) in the rule's matching process and an estimate of the average size of the niches (match sets - see below) in which that rule participates (σ). The initial random population of rules have their parameters set to 10.

On receipt of an input data, the rulebase is scanned, and any rule whose condition matches the message at each position is tagged as a member of the current match set [M]. The rule representation here is the Centre-Spread encoding (see (Stone & Bull, 2003) for discussions). A condition consists of interval predicates of the form $\{\{c_l, s_l\}, \dots, \{c_d, s_d\}\}$, where c is the interval's range centre from [0.0,1.0] and s is the "spread" from that centre from the range (0.0, s_0) and d is a number of dimensions. Each interval predicates' upper and lower bounds are calculated as follows: $[c_i - s_i, c_i + s_i]$. If an interval predicate goes

outside the problem space bounds, it is truncated. A rule matches an input x with attributes x_i if and only if $c_i - s_i \leq x_i < c_i + s_i$ for all x_i .

Reinforcement in YCSc consists of updating the matching error ε which is derived from the Euclidean distance with respect to the input x and c in the condition of each member of the current [M] and the niche size estimate using the Widrow-Hoff delta rule with learning rate β :

$$\varepsilon_j <- \varepsilon_j + \beta(|x - c_j| - \varepsilon_j) \quad (2)$$

$$\sigma_j <- \sigma_j + \beta(|[M]| - \sigma_j) \quad (3)$$

YCSc employs two discovery mechanisms, a niche GA and a covering operator. The general niche GA technique was introduced by Booker (1989), who based the trigger on a number of factors including the payoff prediction "consistency" of the rules in a given [M], to improve the performance of LCS. XCS uses a time-based mechanism under which each rule maintains a time-stamp of the last system cycle upon which it consider by the GA. The GA is applied within the current niche when the average number of system cycles since the last GA in the set is over a threshold θ_{GA} . If this condition is met, the GA time-stamp of each rule in the niche is set to the current system time, two parents are chosen according to their fitness using standard roulette-wheel selection, and their offspring are potentially crossed and mutated, before being inserted into the rulebase. This mechanism is also used here within match sets.

The GA uses roulette wheel selection to determine two parent rules based on the inverse of their error:

$$f_i = \frac{1}{\varepsilon^v + 1} \quad (4)$$

Offspring are produced via mutation (probability μ) where, after (Wilson, 2000), we mutate an allele by adding an amount $+$ or $-$ $rand(m_0)$, where m_0 is a fixed real, $rand$ picks a real number uniform randomly from $(0.0, m_0]$, and the sign is chosen uniform randomly. Crossover (probability χ , two-point) can occur between any two alleles, i.e., within an interval predicate as well as between predicates, inheriting the parents' parameter values or their average if crossover is invoked. Replacement of existing members of the rulebase uses roulette wheel selection based on estimated niche size. If no rules match on a given time step, then a covering operator is used which creates a rule with its condition centre on the input value and the spread with a range of $rand(s_0)$, which then replaces an existing member of the rulebase in the usual way.

Recently, Butz et al. (2004) have proposed a number of interacting "pressures" within XCS. Their "set pressure" considers the more frequent reproduction opportunities of more general rules. Opposing the set pressure is the pressure due to fitness since it represses the reproduction of inaccurate overgeneral rules. Thus to

produce an effective, i.e., general but appropriately accurate, solution an accuracy-based LCS using a niche GA with global replacement should have these two pressures balanced through the setting of the associated parameters. In this paper we show how the same mechanisms can be used within YCSc to identify clusters within a given dataset; the set pressure encourages the evolution of rules which cover many data points (via θ_{GA}) and the fitness pressure acts as a limit upon the separation of such data points, i.e., the error (via v).

3 Initial Performance

In this section we apply YCSc as described above on two datasets for the first experiment to test the performance of the system. The first dataset is well-separated as shown in Fig 1(a). We use a randomly generated synthetic dataset. This dataset has $k = 25$ true clusters arranged in a 5×5 grid in $d = 2$ dimension. Each cluster is generated from 400 data points using a Gaussian distribution with a standard deviation of 0.02, for a total of $n = 10,000$ datum. The second dataset is not well-separated as shown in Fig 1(b). We generated it in the same way as the first dataset except the clusters are not centred on that of their given cell in the grid.

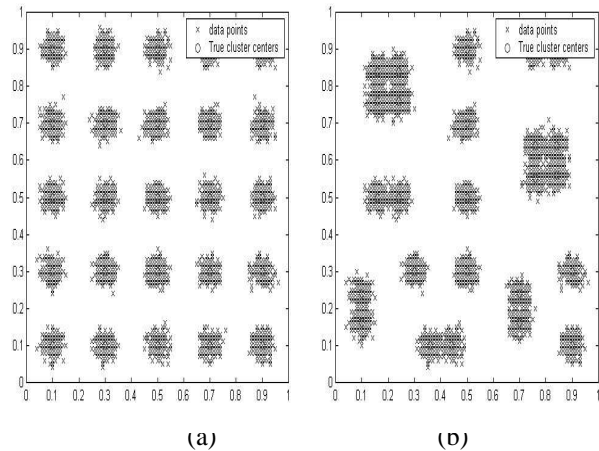
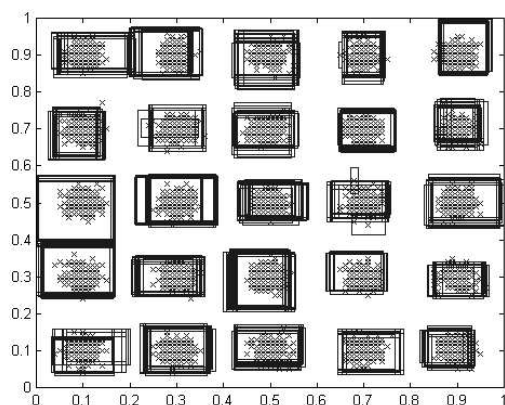


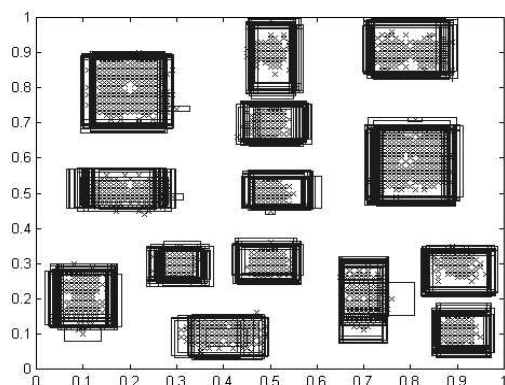
Figure 1: The well-separated (a) and less-separated (b) data sets used.

The parameters used were: $N=800$, $\beta=0.2$, $v=50$, $\chi=0.8$, $\mu=0.04$, $\theta_{GA}=50$, $s_0=0.05$, $m_0=0.01$. All results presented are the average of ten runs. Learning trials consisted of 250,000 presentations of a randomly sampled data point.

Figure 2 shows a typical example solutions produced by YCSc on both data sets. That is, the region of the 2D input space covered by each rule in the final rule-base is plotted along with the data. As can be seen, in the well-separated case the system roughly identifies all 25 clusters whereas in the less-separated case contiguous clusters are covered by the same rules.



(a)



(b)

Figure 2: Typical solutions for the well-separated (a) and less-separated (b) data sets.

As expected, solutions contain many overlapping rules around each cluster. The next section presents a rule compaction algorithm which enables identification of the underlying clusters.

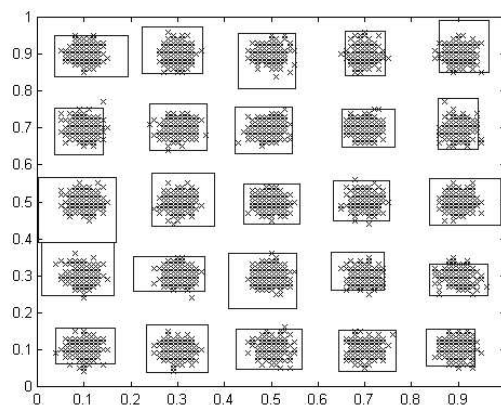
4 Rule Compaction

Wilson (2002) introduced a rule compaction algorithm for XCS to aid knowledge discovery during classification problems (see also (Fu & Davis, 2002)(Dixon et al., 2003)(Wyatt et al., 2004))). We have developed a compaction algorithm for clustering with LCS:

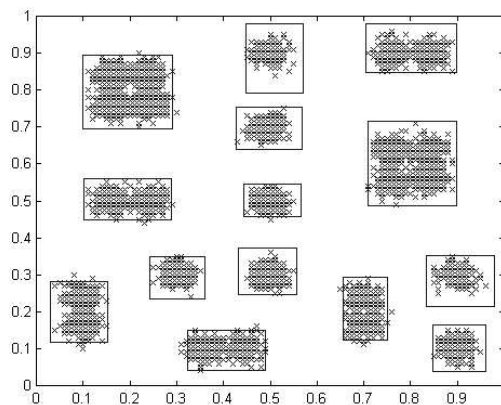
Step 1 Delete the useless rules: The useless rules are identified and then deleted from the ruleset in the population based on their coverage. Low coverage means that a rule matches a small fraction (10%) of the dataset.

Step 2 Find the required rules: The population $[P]_{N[deleted]}$ is sorted according to the numerosity of the rules. Then $[P]_M$ ($M < N$) is formed by selecting the minimum sequential set of rules that covers all data.

Step 3 Remove redundant rules: This step is an iterative process. On each cycle it selects the rule in $[P]_M$ that maximum number of match set. This rule is removed into the final ruleset $[P]_F$ and the data that it covers deleted from the dataset. The process continues until the dataset is empty.



(a)



(b)

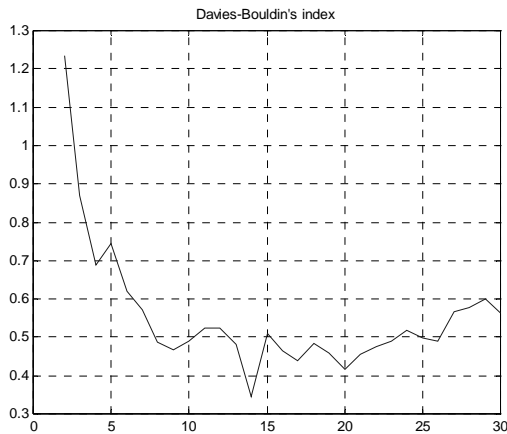
Figure 3: Showing the effects of the rule compaction on the typical solutions shown in Figure 2 for the well-separated (a) and less-separated (b) data sets.

Figure 3 shows the final set $[P]_F$ for both the full solutions shown in Figure 2. YCSc's identification of the clusters is now clear. Under the (simplistic) assumption of non-overlapping regions as described by rules in $[P]_F$ it is easy to identify the clusters after compaction. In the case

where no rules subsequently match new data we could of course identify a cluster by using the distance between it and the centre of each rule.



(a)



(b)

Figure 4: *K*-means algorithm performance using the Davies-Bouldin index for the well-separated (a) and less-separated (b) data sets.

We have examined the average quality of the clustering solutions produced during the ten runs by measuring the total objective function described in equation (1) and checking the number of clusters defined. The average of quality on the well-separated dataset is 8.39 ± 0.39 and the number of clusters is 25 ± 0 . That is, it correctly identifies the number of clusters every time. The average quality on the not well-separated dataset is 24.58 ± 0.63 and the number of clusters is 14 ± 0 . Hence it is not correct every time due to the lack of clear separation in the data.

For comparison, the *k*-means algorithm was applied to the datasets. The *k*-means algorithm (assigned with the known $k=25$ clusters) averaged over 10 runs gives a quality of 32.42 ± 9.49 and 21.07 ± 5.25 on the well-separated and less-separated datasets respectively. The low quality of solutions in the well-separated case is due to the choice of the initial centres; *k*-means is well-known for becoming less reliable as the number of underlying clusters increases. In the less-separated case there is no significant difference in performance between YCSc and *k*-means. For estimating the number of clusters we ran, for 10 times each, different *k* (2 to 30) with different random initializations. To select the best clustering with different numbers of clusters, the Davies-Bouldin validity index is shown in Figure 4. The result on well-separated dataset has a lower negative peak at 23 clusters and the less-separated dataset has a lower negative peak at 14 clusters. That is, it is not correct on both datasets, for the same reason as noted above regarding quality. Thus YCSc performs as well or better than *k*-means whilst also identifying the number of clusters during learning.

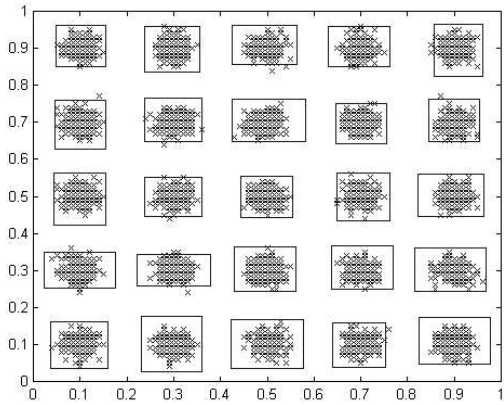
5 Local Search

Previously, Wyatt and Bull (2004) have introduced the use of local search within LCS for continuous-valued problem spaces. Within the classification domain, they used the Widrow-Hoff delta rule to adjust rule condition interval boundaries towards those of the fittest rule within each niche on each matching cycle, reporting significant improvements in performance. The same concept has also been applied to a neural rule representation scheme in LCS (O'Hara & Bull, 2005). We have examined the performance of local search for clustering. Initial results using Wyatt and Bull's scheme gave a reduction in performance, typically more specific rules, i.e., too many clusters, were identified (not shown). Instead, we here introduce a scheme which uses the current data as the target for the local learning to adjust the centres of the rules:

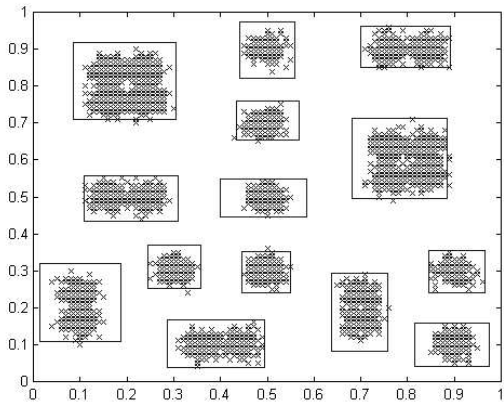
$$c_{ij} < -c_{ij} + l(x_j - c_{ij}) \quad (4)$$

Where c_{ij} represents centre of gene j of rule i in the current match set, x_j represents the value in dimension j of the current input data, and l is the learning rate, here set to 0.1. This is applied on every match cycle before the GA trigger is tested, as in (Wyatt & Bull, 2004).

Figure 5 shows typical solutions once the rule compaction described in the previous section is applied. In the well-separated case, the quality of solutions is increased to 6.45 ± 0.09 , whereas there is no significant difference in the less-separated case with an average quality of 24.24 ± 0.53 . The same number of clusters were identified as before, i.e., 25 and 14 respectively.



(a)



(b)

Figure 5: Typical solutions using local search, after compaction, for the well-separated (a) and less-separated (b) data sets.

6 Increased Complexity

Finally, we have begun to examine the performance of YCSc compared to k -means over randomly generated datasets in several d dimensions with varying numbers k clusters. A Gaussian distribution is generated around each centre, their standard deviation is set to 0.01. Each centre coordinate is s generated from a uniform distribution over the hypercube $[0,1]^d$, the expected distances between cluster centers is set to 0.2 (i.e., well-separated). We test dataset with d -dimensions 2, 4 and 6. The true k clusters are 9 and 25, where we generate 100 data points for each cluster. Then we determine the average quality of clustering and number of clusters from YCSc with local search from 10 runs as before. We also determine for k -

means (the number of k groups was known) the quality and Davies-Bouldin index as before.

Table 1 shows how YCSc always gives superior quality and gives an equivalent or closer estimate of the number of clusters compared to k -means.

Table 1: YCSc with local search vs. k -means on harder datasets.

dataset	d	method	k found	Quality
$k=9$	2	YCSc	9 ± 0	0.52 ± 0.01
		k -means	9	6.27 ± 2.10
$k=9$	4	YCSc	9 ± 0	1.02 ± 0.03
		k -means	9	6.77 ± 2.19
$k=9$	6	YCSc	9 ± 0	1.54 ± 0.03
		k -means	9	8.67 ± 3.82
$K=25$	2	YCSc	25 ± 0	1.81 ± 0.03
		k -means	22	8.23 ± 2.51
$K=25$	4	YCSc	25.40 ± 0.66	3.60 ± 0.04
		k -means	24	8.26 ± 2.68
$K=25$	6	YCSc	25.60 ± 0.80	5.37 ± 0.04
		k -means	22	9.30 ± 2.56

7 Conclusions

In this paper we have presented initial results from a rule-based approach to clustering which uses both evolutionary computing and reinforcement learning techniques - YCSc. Our findings suggest that the approach performs as well or better than the widely used k -means algorithm with the added advantage of not requiring an *a priori* estimate of the underlying number of clusters expected in the data. We are currently applying the approach to a number of larger data sets.

Acknowledgments

This work was partially supported by the Government of Thailand.

Bibliography

- Booker, L.B. (1989) Triggered Rule Discovery in Classifier Systems. In J.D. Schaffer (ed) *Proceeding of the Third International Conference on Genetic Algorithms*. Morgan Kaufmann, pp265-274
- Bull, L. (2004)(ed.) *Applications of Learning Classifier Systems*. Springer.
- Bull, L. (2005) Two Simple Learning Classifier Systems. In L. Bull & T. Kovacs (eds) *Foundations of Learning Classifier Systems*. Springer.
- Butz, M., Kovacs, T., Lanzi, P-L & Wilson, S.W. (2004) Toward a Theory of Generalization and Learning in XCS. *IEEE Transactions on Evolutionary Computation* 8(1): 28-46
- Davies, D. L. & Bouldin, D. W. (1979) A Cluster Separation Measure. *IEEE Trans. On Pattern Analysis and Machine Intelligence*, vol. PAMI-1 (2): 224-227.
- Dixon, P., Corne, D., Oates, M. (2003) A Ruleset Reduction Algorithm for the XCS Learning Classifier System. In Lanzi, Stolzmann & Wilson (eds.), *Proceedings of the 5th International Workshop on Learning Classifier Systems*. Springer, pp.20-29.
- Fu, C. & Davis, L. (2002). A Modified Classifier System Compaction Algorithm. In Banzhaff et al. (eds.) *Proceedings of GECCO 2002*. Morgan Kaufmann, pp 920-925.
- Holland, J.H. (1975) *Adaptation in Natural and Artificial Systems*. Univ. of Michigan Press.
- Holland, J.H. (1976) Adaptation. In Rosen & Snell (eds) *Progress in Theoretical Biology*, 4. Plenum
- O'Hara, T. & Bull, L. (2005) A Memetic Accuracy-based Neural Learning Classifier System. In *Proceedings of the IEEE Congress on Evolutionary Computation*. IEEE Press, pp2040-2045.
- Stone, C. and Bull, L. (2003) For real! XCS with continuous-valued inputs. *Evolutionary Computation*, 11(3):299-336.
- Sutton, R. & Barto, R. (1998) *Reinforcement Learning*. MIT Press.
- Tibshirani, R., Walther, G., & Hastie, T. (2001) Estimating the Number of Clusters in a Dataset Via the Gap Statistic. *Journal of the Royal Statistical Society*, B, 63: 411-423.
- Wilson, S.W. (1995) Classifier Fitness Based on Accuracy. *Evolutionary Computation* 3(2):149-76.
- Wilson, S. W. (2000) Get real! XCS with continuous-valued inputs. In P. L. Lanzi, W. Stolzmann and S. W. Wilson (eds.) *Learning Classifier Systems. From Foundations to Applications*. Springer, pages 209-219.
- Wilson, S. (2002). Compact Rulesets from XCSI. In Lanzi, Stolzmann & Wilson (eds.), *Proceedings of the 4th International Workshop on Learning Classifier Systems*. Springer, pp. 197-210.
- Wyatt, D. & Bull, L. (2004) A Memetic Learning Classifier System for Describing Continuous-Valued Problem Spaces. In N. Krasnagor, W. Hart & J. Smith (eds) *Recent Advances in Memetic Algorithms*. Springer, pp355-396.
- Wyatt, D., Bull, L. & Parmee, I. (2004) Building Compact Rulesets for Describing Continuous-Valued Problem Spaces Using a Learning Classifier System. In I. Parmee (ed) *Adaptive Computing in Design and Manufacture VI*. Springer, pp235-248.
- Xu, R & Winch, D. (2005) Survey of Clustering Algorithm. *IEEE Transactions on neural networks* 16 (3).