Transparent, Online Image Pattern Classification Using a Learning Classifier System

No Author Given

No Institute Given

Abstract. Image pattern classification in computer vision problems is challenging due to large, sparse input spaces with the added demand for generalisation and accuracy of results. The Evolutionary Computation technique of Learning Classifier Systems (LCS) addresses such problems, but has not been applied previously to this domain. Instead, offline, supervised techniques on fixed data sets have been shown to be highly accurate. This paper shows that LCS enable online, reinforcement learning on datasets that may change over time and produce transparent (human readable) classification rules. Further work is needed in domains applicable to offline, supervised learning to achieve benchmark accuracy, but the promising initial results auger well for domains, such as mobile robotics, where compact, accurate and general rules learnt in a graceful manner are required.

Keywords: Learning Classifier Systems, Online Learning, Image Classification.

1 Introduction

In the field of Evolutionary Computation, the concept of *Learning Classifier Systems* (LCS) results in a powerful machine learning technique that successfully combines genetic algorithms and reinforcement learning to evolve populations of decision rules. Since the introduction of the accurate, maximally general XCS system [12], LCS has established itself as a flexible and robust learning technique with the highly desired benefit of transparency, as in typical use the decision rules operate in the problem domain explicitly and are therefore human-interpretable.

Having its roots in cognitive systems [5], LCS models an agent interacting with an environment and use concepts of state, action and reward plus predictions at its core. While this enables learning in temporal multi-step problem scenarios, LCS are capable of dealing with stimulus-response classification problems where they can learn maximally general rules in complex multi-class models.

Computer vision problems such as object detection have historically been considered one of the more challenging applications for machine learning, with many learning techniques being applied in the field. High-dimensionality of the image input spaces combined with typically high sparseness of available training data demand learning techniques that have excellent generalisation properties, yet are very efficient in both training and application stages.

In this paper we start exploring the use of LCS for image classification, as so far the learning concept has seen little application in this field. We argue that LCS have the necessary properties to make it a suitable approach for image pattern classification, discuss the practical aspects of such application, and present some encouraging experimental results, which reinforce our belief that by adapting LCS for such demanding problems it can become better established as a competitive learning technique in the field of computer vision.

Often LCS are utilised in problems with the parameter space that has a directly corresponding feature space, e.g. data mining where each recorded variable may be considered a feature [3]. The task of the LCS is to generalise by identifying uninformative features (labelling them as 'don't care'), whilst linking feature ranges to output classes (it is noted that computed predictions are also possible with approximated feature combinations linked to actions). In computer vision and pattern classification applications, the 'best' features to identify a class/pattern are highly sparse compared with the total possible number/type of features, e.g. $730,000$ possible haar-like features used in a 36x36 image [11]. A standard LCS would not function on $730,000$ features due to a combinatorial explosion in the search space size. (It is noted that GAssist [1]/ BioHEL [2], which are Pittsburgh LCS adapt to large search spaces in the BioInformatics domain). Thus, the aim of this work is to adapt the LCS concept to the problem of pattern classification in images for the first time.

The objectives of the work are to:

1. assess the performance capabilities of a standard LCS in vision processing tasks by identifying suitable test problems, performance criteria and comparison with benchmark results.
2. identify the methods within the standard LCS that require adjustment to this problem domain.
3. introduce novel mechanisms or adapt existing methods into the LCS framework in order to improve performance towards the level of benchmark techniques, e.g. it is hypothesised that a 'messy' encoding that enables the number of features to be autonomous adapted will be necessary [7].

The benefits of utilising the LCS concept extend beyond the hypothesised improved performance, which is unlikely to be achieved in this initial investigation. The primary benefit is transparency in the rules created enabling a human to select, and if necessary, modify the selected features [4]. Initial, offline training is likely to be slower than existing techniques due to the computational overhead of the Evolutionary Computation basis of LCS. However, LCS can be configured as online, reinforcement learning systems, which quickly and gracefully learn and adapt to new instances in the problem domain, e.g. once the system has learnt to recognise generic handwritten characters, it could quickly adapt online to a new user rather than having to be retrained from scratch. The system can also function in reinforcement learning so that the user does not have to correct all specific examples, instead simply give an overall level of performance. Another example domain where the benefits of this research is anticipated to prove useful is in visual object detection in autonomous robotics.

2 Classification with LCS

Traditionally, a learning classifier system represents an agent enacting in an unknown environment via a set of sensors for input and a set of effectors for actions. After observing the current state of the environment, the agent performs an action, and the environment provides a reward. LCS have been adapted to supervised learning through the UCS framework [9], which would be plausibly be a better starting system for image classification if only offline learning was required. However, if online adaptive learning with reinforcement only available from the domain is required, then the approach adopted here is more flexible.

The agent in LCS has two modes of operation, explore (training) and exploit (application), that can be described as follows (for a more detailed description refer to [12]).

In the explore mode the agent attempts to obtain information about the environment and describe it by creating decision rules:

1. observes the current state of the environment, $x \in X$.
2. selects classifiers from the classifier population $[P]$ that have conditions matching the state x, to form the match set $[M]$.
3. performs covering: for every action $a_i \in A$ in the set of all possible actions, if a_i is not represented in $[M]$, a random classifier is generated that matches x and advocates a_i, and added to the population.
4. forms a system prediction array, $P(a_i)$ for every $a_i \in A$ that represents the system's best estimate of the payoff should the action a_i be performed in the current state x. Commonly, $P(a_i)$ is a fitness weighted average of the payoff predictions of all classifiers advocating a_i.
5. selects an action a to explore (probabilistically or randomly) and selects all the classifiers in $[M]$ that advocated a to form the action set $[A]$.
6. performs the action a, recording the reward from the environment, r, and uses r to update the predictions of all classifiers in $[A]$.
7. when appropriate runs a genetic algorithm (GA) to introduce new classifiers to the population. In XCS, two parent classifiers are selected from $[A]$ and two offsprings are produced by applying crossover and mutation on their conditions, such that both offsprings match the currently observed state.

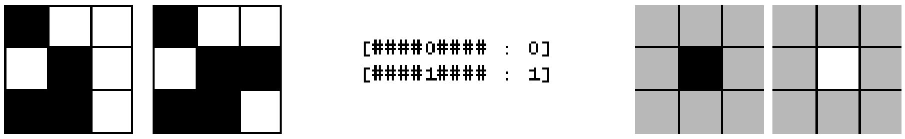
Additionally, the explore mode may perform subsumption, to merge more specific classifiers into more general, accurate ones, and deletion, if the classifier population size grows larger than the specified limit. In contrast, in the exploit mode the agent does not attempt to learn and simply performs the action with the best predicted payoff.

The Markov property of the environment is assumed, meaning that the same action in the same state will result in the same reward. LCS have been shown to be robust to small amounts of noise and are often more robust than most machine learning techniques with increasing amounts of noise [4], The generalisation property in LCS allows a single rule to cover more than one state provided that the action-reward mapping is similar.

3 Approaches to Image Pattern Classification

3.1 Naïve Pixel-based Conditions

To learn compact and general models, LCS utilises generalising condition rules in the individual classifiers. In simple ternary encoding schemes, generalising conditions are achieved using a special 'don't care' symbol (#). Consider simple binary 3×3 pixel black (0) and white (1) images, where every image can be encoded as a string of 9 bits. To learn the distinction of images where the centre pixel is white from images where the centre pixel is black, two classifiers would be sufficient (see also Figure 1): Notice that the two classifiers above are maximally



```
[####0#### : 0]
[####1#### : 1]
```

Fig. 1. Simple pattern classification problem of distinguishing 3×3 patterns based on the colour of the centre pixel from examples (left) can be solved with two classifiers (middle) using 'don't care' encoding (depicted as gray, right).

accurate and general, and cover the entire problem domain. However as soon as the problem gets a little more complicated, generalising at the pixel level becomes difficult. Consider learning to recognise images that have a horizontal line of 3 white pixels on any of the rows. Three classifiers are needed to model the positive class:

$$[111\#\#\#\#\#\#] \rightarrow 1,$$
$$[\#\#\#111\#\#\#] \rightarrow 1,$$
$$[\#\#\#\#\#\#111] \rightarrow 1,$$

and yet another 9 rules are needed to fully cover the negative class. While learning such problem is still fully possible, when we get to useful image pattern sizes (hundreds of pixels) and more pixel states (e.g. 256 gray-scale values), there are typically thousands of example instances of every pattern class, and in turn those commonly represent only a very sparse sampling of the underlying problem, e.g. all the possible images representing the object of interest. At pixel level, the different images depicting the same object will often be so different, that generalisation with a 'don't care' pixel is not effective, and the LCS is forced to keep one classifier for every example instance it has seen, resulting in poor pattern recognition performance.

Significant image differences at the pixel level are a well known problem in computer vision, and it is commonly tackled using some form of feature extraction. In the next section we show how a popular Haar-like feature can be used to enable LCS learning for image classification.

3.2 Haar-like Feature Conditions

One popular type of features that is used in state of the art image classification systems [11] is the Haar-like rectangular features. The features efficiently compute differences in pixel intensity sums of neighbouring rectangular regions (see Figure 2) by utilising a so called *integral image*, where each pixel is replaced by a sum of all pixels to the left and above:

$$II(x,y) = \sum_{i,j=1}^{x,y} I(i,j),$$

and the integral image II can be computed with a single pass over the image I. The value of feature f at location $l = (x,y)$ and scale $s = (width, height)$, $f(x,l,s)$ can be computed with just a few lookup calls in the integral image II.

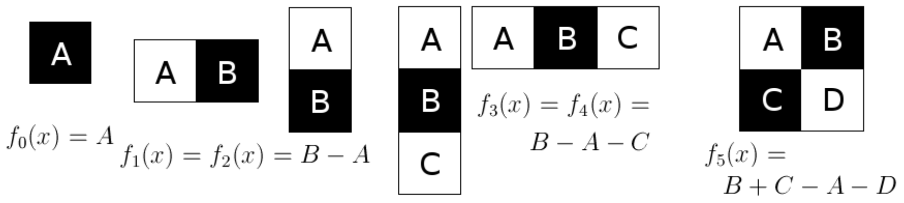


Fig. 2. Haar-like features. The feature values are computed by subtracting sums of pixel intensities in neighbouring rectangular regions A, B, C, D. Note that when applied to an image, the position and scale of the feature is important.

By applying a threshold t and comparison direction d on the outputs of the Haar-like features, binary decision rules can be formed that detect the presence (or absence) of contrast in neighbouring regions in the image. We thus propose the following conditions for use in the LCS decision rules:

$$c = c(f,l,s,d,t),$$

$$c(x) = \begin{cases} \text{true, if } f(x,l,s) * d < t * d \\ \text{false, otherwise} \end{cases}$$

Notice that the Haar-like features are weak, in the sense that a single feature is insufficient to describe a complex pattern. We therefore utilise a ‘messy’ encoding [7]: by allowing multiple feature conditions to be joined using a logical ‘*and*’ operator, the resulting decision rule conditions are sufficiently complex to make learning feasible:

$$c(x) = c_1(x) \wedge \ldots \wedge c_m(x).$$

For complete maximally general learning to occur, the encoding that is effectively provided by the classifier conditions must have an important property of allowing both generalisation and accuracy—the classifiers should be general, but not over-general. We argue that the suggested Haar-like multi-feature conditions exhibit that property:

- **Generalisation** that the symbolic encoding gained with the 'don't care' symbol (#). Haar-like features achieve this by ignoring the image information outside of the feature positions and by thresholding the feature values. An extreme case of generalisation (all #) can be achieved by setting a threshold on a feature such that every feasible image pattern will match.
- **Accuracy/specificity**. Every condition can be made more specific by adding more features to it. Essential for ensuring this property is the 'type-zero' Haar-like feature that simply returns the sum of the pixel intensities within a rectangular region, which effectively enables very precise thresholding of individual pixel values if needed. An extreme case of specificity, when no generalisation is possible, is a set of 'type-zero' single pixel features that completely describe a single unique image/state.

In practice, the LCS learning attempts to select a good trade-off between the two extremes, as it has evolutionary pressures [4] for both accuracy and generalisation, and the Haar-like multi-feature conditions provide sufficient flexibility for the search along this front, as our experimental results suggest.

4 Experimental Results

4.1 Datasets

We applied the Haar-like multi-feature LCS to the problem of handwritten digit classification. The widely known MNIST benchmark database [8] contains a training set of $60,000$ example images of all ten handwritten digits $0 \ldots 9$, collected from 250 individuals. The examples are presented as 28×28 pixel grayscale images, centred around the pixel intensity 'centre of mass'. The proposed system does no preprocessing of the training data, which is known to improve results. The database also contains a separate evaluation set of $10,000$ examples, sampled from a separate group of 250 individuals.

4.2 Experimental Setup

We used an implementation of XCS based on XCSJava package by Butz [4], adapted to work with image patterns. The necessary adjustments include: **Features**. 6 types of Haar-like features were defined: a single rectangle sum ('type-zero' feature that is not known to be used in other Haar-related approaches), two rectangle difference (horizontal, vertical), three rectangle difference (horizontal, vertical), and a four rectangle difference feature (see Figure 2). We allowed up to 8 individual features in a single condition to form 'messy' classifiers.
Covering. When performing covering, a random number of features were generated for the condition, randomly selecting feature type, position, scale and direction, but setting the threshold to the current value of the feature, ensuring that the condition matches the currently observed state.
Crossover. During crossover of two classifier conditions, for every feature in the 'longer' condition, a random feature was selected from the 'shorter' condition, and uniform crossover was performed between the two features.

Mutation. Every property of every feature was allowed to mutate, except for the threshold values, which after the mutation step were set to the values of mutated features in the current state, ensuring that the condition matches the currently observed state.
Condition validation. In the cases where crossover or mutation moved the feature window to an infeasible region in the image, the offspring was subsumed by the parent classifier by increasing the numerosity of the latter.
Population and generations. The population size was limited to $60,000$ classifiers, and the experiments were run for $4,000,000$ generations.
Significance. The experiments were repeated 30 times, and the reported numbers are averages, with standard deviation demonstrated where applicable.

4.3 Results

Figure 3 shows the behaviour of the classification performance and relative population size in an 'offline' scenario, with $4,000,000$ generations taking $15-20$ hours for each of the 30 runs. The system reaches $91 \pm 1\%$ overall classification rate on the unseen test set. Additionally, Table 1 contains the confusion matrix of the LCS when applied to the unseen evaluation set, showing that the system does indeed generalise and over-fitting was not evident.

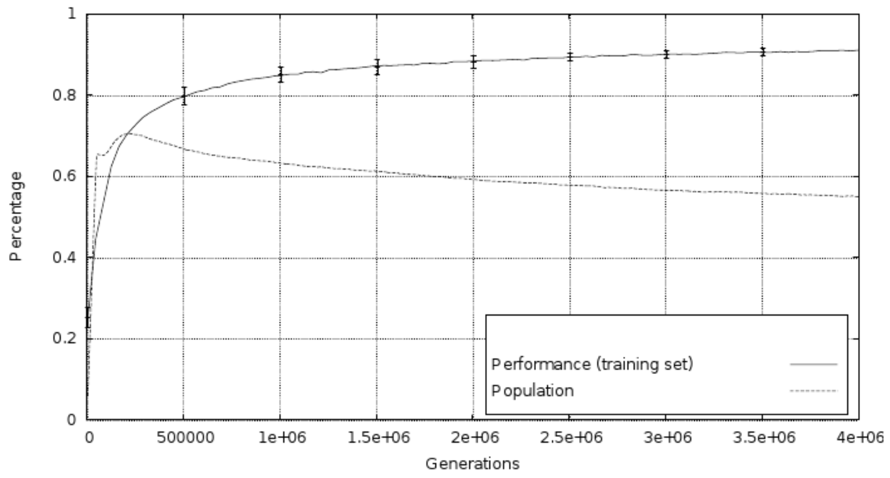


Fig. 3. Performance of the LCS with error bars on 30 repetitions as measured internally on the training set, and average population size (number of unique classifiers compared with 60,000 limit)

Figure 4 shows example classifiers represented as feature images. Note that some classifier conditions are intuitively interpretable and target the regions of high contrast where the curves of handwritten digits will consistently pass through, while others are harder to interpret yet are useful to the system due to its cooperative nature.

Figure 5 shows the performance of the suggested system in an 'online' learning scenario. Here, the training environment initially provides only two of digit

Table 1. Confusion matrix for the independent evaluation set (mean performance ± standard deviation, %). Rows correspond to actual class, columns correspond to the system classification. E Expected, A Actual

E\A	0	1	2	3	4	5	6	7	8	9
0	**96±1**	0±0	0±0	0±0	0±0	0±0	1±1	0±0	1±1	0±1
1	0±0	**99±0**	0±0	0±0	0±0	0±0	0±0	0±0	0±0	0±0
2	1±0	1±1	**90±3**	1±1	1±0	0±0	1±1	2±1	3±2	1±1
3	0±1	0±1	2±1	**89±3**	0±0	1±1	0±1	2±1	2±1	2±2
4	0±0	0±0	0±0	0±1	**93±2**	0±0	1±0	0±0	1±1	3±2
5	1±1	0±0	0±0	4±2	0±0	**87±4**	2±1	1±1	2±1	2±2
6	1±0	1±0	0±0	0±0	1±1	2±1	**92±1**	0±0	2±1	0±1
7	0±0	1±0	3±1	1±0	1±0	0±0	0±0	**90±2**	1±1	4±2
8	1±2	1±1	1±1	3±3	1±1	2±2	1±2	2±1	**85±6**	3±3
9	1±0	1±0	0±0	1±1	4±3	1±1	0±0	3±2	2±2	**87±5**

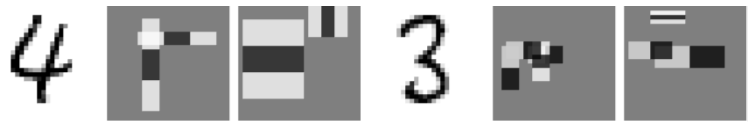


Fig. 4. Example training patterns and matching classifiers. First row shows an example of digit '4' and conditions of two corresponding classifiers. Second row shows an example of digit '3' and its matching conditions that are harder to interpret.

classes, '0' and '1', which are learnt easily, then subsequent digit classes are added to the training every 200, 000 generations. The 'on-line' nature of LCS enables the system to recover from the performance drop effected by unseen classes of examples.

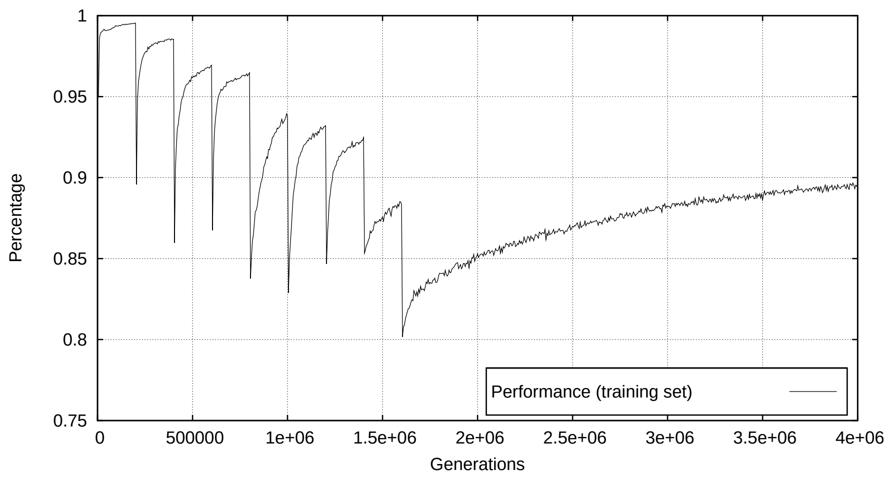


Fig. 5. Online learning with LCS. In the first half of the training, a new digit class (sequentially 0 to 9) is introduced every 200, 000 generations.

Finally, Table 2 shows the comparison of the proposed method to the performance of other known systems on the MNIST dataset.

Table 2. Performance of some classification systems on the MNIST dataset.

System	Method	Processing	Error rate
LeCun et al. 1998 [8]	Linear classifier (1-layer NN)	None	12.0
LeCun et al. 1998 [8]	SVM, Gaussian Kernel	None	1.4
Kégl et al. 2009 [6]	Product of stumps on Haar features	None	0.87
Proposed System	LCS	None	9.0 ± 1

5 Discussion and Future Work

While this straightforward approach of using Haar-like features to enable LCS classification of image patterns does not reach the best reported performance of other machine learning methods, we believe that the results presented here are promising and encourage further research into an improvement of learning classifier systems for image pattern classification.

Training time is slow compared with supervised techniques, e.g. 4M generations processed in 15-20 hours. However, as a single feature can be evaluated with only a few operations, and even with the entire population needing to be processed to find the match set, operational time is only a few milliseconds. Future work could focus on efficiency of the LCS application using cascade concepts.

Online learning removes the need for a human operator to decide when to retrain supervised offline learning algorithms and negates the need to know what dip in performance level necessitates retraining when new instances of data/classes become available.

The LCS does deliver on the promise to form a generalising model using human-interpretable rules. The current Haar-like multi-features approach learns descriptions of patterns that are comparable to other Haar related approaches, like the Viola&Jones AdaBoost cascade [11] or the Haar-SVM of Rätsch et al. [10], and with further extensions could be adapted to training very efficient classifier cascades. Furthermore, other image feature types should be explored to determine which ones can be most effectively used with LCS.

Finally, observation of the behaviour of LCS in large complex scenarios, like those posed by computer vision and image processing problems, presents a good opportunity to further improve and tune the LCS technique itself, which is essential to further establish learning classifier systems as a competitive machine learning methodology.

6 Conclusions

The technique of Learning Classifier Systems can be applied to the field of pattern classification demonstrating novel functionality and promising results. The transparent nature of production rules, which is anticipated to be required in many real world domains, was assisted by the flexible encoding so appropriate features were autonomously learnt. The generalisation capability of LCS, when coupled with the 'messy' encoding enabled compact, accurate and general classifiers to form. Online learning was possible, so provided the utility of recognition is available, an exact supervisory feedback is not required. Further, online graceful learning has been demonstrated, which negates the need for retaining from start on the presentation of novel classes from the domain. Currently, LCS training is slow (compared with alternative supervised techniques), albeit the operational times once trained were 'real-time'. The promising initial results encourage the proposed future work in order to raise the accuracy of classification. Therefore, the LCS technique should be considered in domains where transparent, online learning of patterns is required, e.g. mobile robotics.

References

1. Bacardit, J., Butz, M.V.: Data mining in learning classifier systems: Comparing XCS with GAssist. In: Advances at the frontier of Learning Classifier Systems, vol. 4399, pp. 282–290. Springer-Verlag (2007)
2. Bacardit, J., Krasnogor, N.: Biohel: Bioinformatics-oriented hierarchical evolutionary learning. Nottingham eprints, University of Nottingham (2006)
3. Blind:
4. Butz, M.V.: Rule-based evolutionary online learning systems: A principled approach to LCS analysis and design. Springer Verlag, Berlin Heidelberg (2006)
5. Holland., J.: Adaptation in Natural and Artificial Systems: An Introductory Analysis with Applications to Biology, Control, and Artificial Intelligence. MIT Press (1975)
6. Kégl, B., Busa-Fekete, R.: Boosting Products of Base classifiers. In: Proceedings of the 26th Annual International Conference on Machine Learning. pp. 497–504. ICML '09, ACM, New York, NY, USA (2009)
7. Lanzi, P.L., Perrucci, A.: Extending the representation of classifier conditions part ii: From messy coding to s-expressions. In: Banzhaf, W., Daida, J., Eiben, A.E., Garzon, M.H., Honavar, V., Jakiela, M., Smith, R.E. (eds.) Proceedings of the Genetic and Evolutionary Computation Conference. vol. 1, pp. 345–352. Morgan Kaufmann, Orlando, Florida, USA (13-17 July 1999)
8. LeCun, Y., Bottou, L., Bengio, Y., Haffner, P.: Gradient-based Learning Applied to Document Recognition. Proceedings of the IEEE 86(11), 2278–2324 (1998)
9. Orriols-Puig, A., Bernadó-Mansilla, E.: A further look at ucs classifier system. In: Proceedings of the 9th International Workshop on Learning Classifier Systems - IWLCS2006. Springer - to appear (2006)
10. Rätsch, M., Romdhani, S., Vetter, T.: Efficient Face Detection by a Cascaded Support Vector Machine Using Haar-like Features. Pattern Recognition pp. 62–70 (2004)
11. Viola, P., Jones, M.: Rapid Object Detection Using a Boosted Cascade of Simple Features. In: IEEE Computer Society Conference on Computer Vision and Pattern Recognition. vol. 1 (2001)
12. Wilson, S.: Classifier Fitness Based on Accuracy. Evolutionary Computation 3(2), 149–175 (1995)