Applying Learning Classifier Systems to Acoustic Scene Classification: DCASE 2017 Challenge

CITS4404 Artificial Intelligence & Adaptive Systems Team Project

Yiyang Gao (21263128), Aaron Hurst (21325887), Kevin Kuek (21307006), and Scott McCormack (21875529)

School of Computer Science and Software Engineering

3rd November, 2017

Abstract

Sound classification is an emerging field of research with multifarious technology and big-data applications. To promote research in this space, the DCASE Challenge was instigated with multiple tasks for competitors to attempt. Here, we have investigated the efficacy of Learning Classifier Systems applied to the problem of Acoustic Scene Classification. This involves predicting the environment in which a sound file was captured. Results indicate that even on a highly abbreviated feature set, Learning Classifier Systems can achieve modest accuracy in comparison to standard solutions.

1 Introduction

Write introduction

2 Background

This section provides a brief review of learning classifier systems (2.1), the DCASE Challenge (2.2) and acoustic scene classification (2.3).

2.1 Learning Classifier Systems

First introduced in the mid-1970s, Learning Classifier Systems (LCSs) are a rule-based machine leaning algorithm with a unique combination of learning mechanisms, including a genetic algorithm (GA) [1]. The core of an LCS is a population of rules, or *classifiers*, which collectively form the solution to the given problem [2]. This population of classifiers is gradually evolved toward an optimal and maximally general set [2].

The motivation for this structure is that, when modelling and attempting to predict the outcome of complex systems, a desirable approach is to develop a distributed population of classifiers – in the form of rules – that together form an accurate model [2, p. 2]. Each classifier, then, spans a subspace of the problem, with the population spanning the entire problem. Individual classifiers consist of a condition-action rule which says: 'If a problem instance matches this *condition*, perform this *action*'. Classifier fitness is evaluated based on feedback from the problem (generally referred to as the 'environment').

The learning process of a LCS includes a rule discovery method known as covering, a generalisation-pressure effect called subsumption, a GA, fitness-based deletion to maintain a finite-sized population and, in some applications, reinforcement learning [3].

The rule discovery method, covering, is used to initialise the population by adding a new classifier whenever a problem instance matches no existing classifier. Subsumption is used to eliminate classifiers with more specific conditions that are no more accurate than others with equivalent, but more general conditions. The GA is used as a secondary rule discovery method that only operates on high performing rules. Deletion is employed to maintain a finite population size by removing poorly performing – i.e. low fitness – classifiers. Reinforcement learning may be used as a final step in the learning cycle for problems where feedback from the environment is delayed.

A seminal work in the field of LCSs was the introduction of the eXtended Classifier System [4, 5]. This incorporated a number of features which substantially improved the performance of LCSs [4].

A key distinction amongst LCS applications is between supervised learning, in which feedback from the environment is delayed (such as robot navigation), and offline learning, where feedback is immediate and the correct action known in advance (such as classification tasks). This distinction determines whether reinforcement learning is necessary and affects how classifier accuracy is calculated.

2.2 DCASE Challenge

Sound classification, or machine listening, is seen as a promising research field with wide-ranging applications [6]. The DCASE challenge has been established to encourage and support work in this space. The challenge provided participants with standard development (training) and evaluation (testing) datasets [7] and a baseline system for comparison and/or extension [6, 8].

The Challenge spans four sound recognition tasks: acoustic scene classification, detection of rare sound events, sound event detection in real life audio and large-scale weakly supervised sound event detection [6]. This paper focuses on acoustic scene classification. For this task, the Challenge provides datasets containing sound files obtained across 15 different contexts, such as in a car, library or office [8]. Each sound file in 3-5 minutes long natively, but was split into multiple 10 second long segments for the datasets [8]. Overall, 312 segments are provides for each context.

A baseline system for this task is provided in Python and uses a neural network with two hidden layers of 50 neurons each to classify sound files [6]. The baseline system also provides support for extracting features from the datasets using what is known as log mel-band energies, as described in Section

Add reference to Kevin's section

[6].

2.3 Acoustic Scene Classification

The first task of the 2017 DCASE Challenge, and the focus of this paper, is acoustic scene classification. Barchiesi et al. define this as "the task of associating a semantic label to an audio stream that identifies the environment in which it has been produced" [9, p. 17]. Potential applications in this domain revolve around context aware smart devices, such as smartphones and hearing aids, that adjust their functioning based on the environments in which they find themselves [9].

The typical approach taken to acoustic scene classification is to segment the original sound file into many, small 'frames', calculate a set of features over each frame, use the resulting feature array to train a statistical model and finally apply some decision rule for assigning classification labels [9, pp. 18–19]. Many approaches submitted to the 2017 DCASE challenge trained some form of neural network as their 'statistical model' [8].

The results of the 2017 DCASE Challenge show that the baseline solution provided achieved an accuracy of 74.8% on the development dataset and 61.0% on the evaluation set averaged across all sound classifications [8]. Many entrants successfully outperformed the baseline on both the development and evaluation datasets; however, all algorithms performed worse on the evaluation set compared to the development set [8]. The best performing solution achieved an accuracy of 87.1% and 83.3% on the development and evaluation datasets, respectively [8, 10].

3 Feature Extraction

Import Kevin's section

4 Experiment 1: Baseline System

Preliminary investigation into LCSs built by others, led us to sourcing from a sample system hosted on GitHub titled *eLCS*. This Michigan-style classifier was developed as an educational tool which assisted with demonstrating the intricacies and components that contribute towards building a typical LCS. Five demo folders are utilized to show the steps towards building a typical LCS and a dataset consisting of discrete attributes and a starting rule population is provided to run the LCS. The last folder in this system, titled *Demo 5* was utilized as a starting point for our baseline system.

5 Experiment 2: XCSR Implementation

In response to the positive results obtained from the baseline system described above, a separate LCS algorithm was coded by the authors. This provided much greater ability to control various aspects of the algorithm and more clarity as to the overall system's functioning. This section presents the general approach to developing this algorithm (Section 5.1), a description of each of the key parameters (Section 5.2) and the key modifications made to the standard LCS design (Section 5.3).

5.1 Overview of Approach

The LCS algorithm developed by the authors was closely based on the structure and pseudocode provided by Butz and Wilson in their paper entitled "An Algorithmic Description of LCS" [3]. This paper provides detailed descriptions for each of the key modules within XCS, which is one of the most well-used LCS implementations [5]. The decision to embark on developing our own LCS was largely based on the availability of the pseudocode in this paper.

Two fundamental changes made to the algorithm presented by Butz and Wilson were the change to continuous data, and hence continuously-valued classifier rules, and the switch from reinforcement learning to offline learning. Both changes necessitated a significant number of alterations throughout the algorithm. Various papers were consulted for how to make these changes. In particular, a number of papers on XCSR – the standard continuous version of XCS – were used [11, 12, 13, 14].

The GA required a number of additional design decisions. These included the use of two-point crossover, roulette-wheel selection of parents and employing GA subsumption (parents can subsume children if more general). The GA was also applied in a 'niched' manner, acting only on the correct set. Niched GA operation has been noted as a significant contributor to LCS performance [4]. Accuracy-based fitness, another key contributor to LCS performance [4], was also used.

As discussed in Section

Add reference to Kevin's section

, the mean and standard deviation of log mel-bands were used as the features for this experiment. All 312 sound files from all of the 15 sound contexts were considered. That is, the LCS *environment* consisted of 312 instances (sound files) per endpoint (context), giving a total of 4,680 instances, each containing 80 attributes (features; 40 means, 40 standard deviations). Data was split into training and testing datasets with a 60:40 ratio.

5.2 Parameters

Import Yiyang's section

5.3 Problem-Specific Modifications

Mutation scheme (plus note on probabilities: Sometimes a child will just be copies of their parents: P(no crossover) = 0.3, P(no mutation) = (0.98))

Correct set subsumption – criteria for most general classifier determined as most wildcards or equal most and largest range

Subsumption conditions – tolerance and removed wildcard criteria

Deletion experience threshold – changed deletion criteria so that classifiers with little utility (low match count) can be deleted, motivation was the really low average match count – show graph of function

$$\frac{exp_{min}}{3.05}\left(\pi/2 - \arctan\left(\frac{\text{age} - age_{min}}{20}\right)\right)$$

6 Results

Yiyang: feel free to mess around with the sub-headings under Results if you wish

6.1 Environment Representations

Alternative feature processing investigated

6.2 Parameter Tuning

Reason why standard deviation and mean are used, what information they carry

Car and office confusion matrix Plot for std dev and mean (for car and office) to explain Car, office and city centre (with confusion and plots) Then add metro station and show results (worse) - explain why it is worse Show a classifier on a graph to show how it can match instances Overall accuracy (all 15 features)

7 Discussion

Our approach is good because it is more general (no need to construct statistical models or decision rules) and significantly reduces the number of features that need to be analysed, and therefore the complexity of the algorithm. [THIS IS COMPARING TO STANDARD APPROACH TO ACS / DCASE CHALLENGE SUBMISSIONS]

8 Conclusion

Write conclusion

References

[1] M. V. Butz, "Learning classifier systems," in *Springer Handbook of Computational Intelligence*. Springer, 2015, pp. 961–981.

[2] R. J. Urbanowicz and J. H. Moore, "Learning Classifier Systems: A Complete Introduction, Review, and Roadmap," *Journal of Artificial Evolution and Applications*, vol. 2009, pp. 1–25, 2009.

[3] M. V. Butz and S. W. Wilson, "An Algorithmic Description of XCS," in *International Workshop on Learning Classifier Systems*. Springer, 2000, pp. 253–272.

[4] P. L. Lanzi, "Learning Classifier Systems: Then and Now," *Evolutionary Intelligence*, vol. 1, no. 1, pp. 63–82, 2008.

[5] O. Sigaud and S. W. Wilson, "Learning Classifier Systems: A Survey," *Soft Computing*, vol. 11, no. 11, pp. 1065–1078, 2007.

[6] Mesaros, Annamaria and Heittola, Toni and Dimint, Aleksandr and Elizalde, Benjamin and Shah, Ankit and Vincent, Emmanuel and Raj, Bhiksha and Virtanen, Tuomas, "DCASE 2017 challenge setup: Tasks, datasets and baseline system," in *Detection and Classification of Acoustic Scenes and Events*, 2017.

[7] Mesaros, Annamaria and Heittola, Toni and Virtanen, Tuomas, "TUT Database for Acoustic Scene Classification and Sound Event Detection," in *24th European Signal Processing Conference 2016 (EUSIPCO 2016)*, 2016.

[8] A. Mesaros and T. Heittola, "Acoustic scene classification - dcase2017," 2017. [Online]. Available: http://www.cs.tut.fi/sgn/arg/dcase2017/challenge/task-acoustic-scene-classification

[9] D. Barchiesi, D. Giannoulis, D. Stowell, and M. D. Plumbley, "Acoustic scene classification: Classifying environments from the sounds they produce," *IEEE Signal Processing Magazine*, vol. 32, no. 3, pp. 16–34, 2015.

[10] S. Mun, S. Park, D. Han, and H. Ko, "Generative adversarial network based acoustic scene training set augmentation and selection using SVM hyper-plane," DCASE2017 Challenge, Tech. Rep., 2017.

[11] D. Sowden, "Investigating the Learning Classifier Systems XCSR and XCSF," 2007. [Online]. Available: http://www.cs.bath.ac.uk/~mdv/courses/CM30082/projects.bho/2006-7/Sowden-DJ-dissertation-2006-7.pdf

[12] C. Stone and L. Bull, "For real! XCS with continuous-valued inputs," *Evolutionary Computation*, vol. 11, no. 3, pp. 299–336, 2003.

[13] S. W. Wilson, "Get real! xcs with continuous-valued inputs," in *Learning Classifier Systems: From Foundations to Applications.* Springer-Verlag, 2000, pp. 209–219.

[14] M. Behdad, L. Barone, T. French, and M. Bennamoun, "On xcsr for electronic fraud detection," *Evolutionary Intelligence*, vol. 5, no. 2, pp. 139–150, 2012.