Code Fragments: Past and Future use in Transfer Learning

Will N. Browne
Victoria University of Wellington, New Zealand,
will.browne@vuw.ac.nz

ABSTRACT

Code Fragments (CFs) have existed as an extension to Evolutionary Computation, specifically Learning Classifiers Systems (LCSs), for half a decade. Through the scaling, abstraction and reuse of both knowledge and functionality that CFs enable, interesting problems have been solved beyond the capability of any other technique. This paper traces the development of the different CF-based systems and outlines future research directions that will form the basis for advanced Transfer Learning in LCSs.

CCS Concepts

•**Computing methodologies** → **Machine learning algorithms;**

Keywords

Code Fragments, Transfer Learning

1. CODE FRAGMENTS

The need for Code Fragments (CFs) arose when applying the evolutionary computation (EC) technique of Learning Classifier Systems (LCSs) to data mining, when it was not possible to discover rules linking higher-order 'abstracted' information about input variables. Follow-on work showed that abstract rules in the toy problem of 'Connect 4' could be learned in a bottom-up manner, but this required two interacting systems.

The ability to manipulate low-level schema has been considered implicit in the successful functioning of EC approaches. Such building-blocks of knowledge have been explicitly identified through Automatically Defined Functions (ADFs) in Genetic Programming. However, ADFs are formed in a *top-down* manner, i.e. complete solutions first, which can be hard to discover. **CFs are** similar to a ADFs in terms of being **small sub-trees of GP-like syntax**, but they are **formed in a bottom-up manner** and restricted to be of depth two initially.

GECCO'16 Companion July 20-24, 2016, Denver, CO, USA

ACM ISBN 978-1-4503-4323-7/16/07.

DOI: http://dx.doi.org/10.1145/2908961.2931737

CFs were not the first approach to adopt GP-like syntax (including S-expressions) into the LCS framework, but the main purpose is different from feature selection or output computation. The purpose is to reuse CFs as *building-blocks of knowledge* from small problems to improve learning in larger-scale problems of the same domain or in a related domain. Furthermore, CFs can now include *building-blocks of functionality* through reusing learned rule-sets as functions, which also include their own CFs.

It is noted also that multimodal and dynamic domains reuse learnt solutions, but these keep whole solutions for reuse. Here, CFs are *part* solutions or even whole solutions (rule-sets) form *sub-parts* of other CFs.

CFs are embedded into the LCS framework due to its cognitive systems roots for learning and practical considerations. Thus the flexible reinforcement learning LCS, XCS, is the base framework, but alternatives, such as the supervised approaches, can be adapted.

XCSCFC is the primary CF-based system, replacing traditional conditions with CFs to indicate a matched state. It scales to problems in a similar domain by reusing learnt CFs as blocks of knowledge, e.g. scaling from 6-bits up to 135-bit Multiplexer problems (MUX).

XCSCFA utilises CFs in the action for a computed action-like system. This demonstrated the importance of consistency underlying CF-based learning that leads to compact solutions avoiding bloat and redundant classifiers.

XCSSMA encoded Finite State Machines as actions, which can encode 'loop' patterns, enabling generic (n-bit) solutions to problems, e.g. even parity.

XCSCF2 was the first to include CFs and Code-functions through reusable rule-sets, with associated CFs, bootstrapping learning. Long chains of rules, CFs, associated functions and so forth can be reduced using distilled rules.

XCSCF* using a layered-learning approach where instead of an experimenter specifying the problem & system parameters & system functions, they specify a sequence of problems & system parameters only as the needed functions are sub-problems in the sequence. This generated a general, i.e. any n-bit, solution for MUX for the first time.

Although more functions than required can exist, it is still a human ordered sequence. Future work will parallelise unordered problems such that when their learning is complete the learnt functionality and CFs will be transferred (become available) to currently uncompleted problems.

1.1 Acknowledgments: Co-authors

Dan Scott, Charalambos Ioannides, Muhammad Iqbal, Mengjie Zhang, Isidro M. Alvarez, Syed Saud Naqvi, Yi Liu.