A framework for measuring the generalization ability of Geometric Semantic Genetic Programming (GSGP) for Black-Box Boolean Functions Learning

Andrea Mambrini[1], Yang Yu[2], Xin Yao[1]

[1] University of Birmingham, Birmingham, UK
[2] Nanjing Univeristy, Nanjing, China

Abstract. Moraglio et al. proposed GSGP operators for learning boolean functions [1]. The work provides upper bounds of the expected time for the algorithm to fit the training set but it doesn't give any guarantees on how the learned functions will evolve on unseen input. In this work we provide a framework to analyse GSGP as learning tool. This can be used to obtain lower bounds on the generalisation error of the boolean functions evolved by the algorithm.

1 A framework for measuring Generalization

In this section we define the learning problem which GSGP using the operators from [1] tries to solve and the *generalization ability* as a way to measure the performance of a GP algorithm as a learning tool.

1.1 The learning problem

Given a complete truth table $C = \{(x_1, y_1), ..., (x_N, y_N)\}$ consisting in the complete description of the input-output behaviour of a fixed boolean function $h : \{0,1\}^n \to \{0,1\}$ in n variables ($N = 2^n$), given an underlying distribution D over the rows of the truth table, i.e., $D_i \geq 0$ for all row i and $\sum_{i=1}^{N} D_i = 1$. A training set T consisting in $\tau \leq N$ test cases $T \subset C = \{(x_1, y_1), ..., (x_\tau, y_\tau)\}$ is sampled from the truth table according to the distribution D. The aim is to use just the training set T to learn a boolean function $h : \{0,1\}^n \to \{0,1\}$ matching as better as possible the input-output behaviour described by C with respect to the distribution D, i.e., to minimise the *generalisation error*,

$$\epsilon_g(h) = \sum_{i=1}^{N} D_i \cdot I[h(x_i) \neq y_i]$$

where $I[\cdot]$ is the indicator function that is 1 if its inner expression is true and 0 otherwise.

The problem has the additional constraint of being black-box. Which means that the learning algorithm has no direct access to T. It has instead access to an oracle that, given a candidate boolean expression X will return how well it match the input-output behaviour described by T.

1.2 Fitness functions

In order to highlight the black-box constraint we will define two different fitness functions. The *fitness on the training set*, $f_T(X)$ is the training error, which measures how well a given boolean expression X matches the training set T. It is defined as

$$f_T(X) = \epsilon_t(X) = \frac{1}{\tau} \sum_{i=1}^{\tau} I[X(x_i) \neq y_i] \ , \ \forall (x_i, y_i) \in T$$

and it is equal to zero when an individual perfectly match the training set T, while it has its worst value of 1 when the training set is completely unmatched. This fitness function represents the oracle described above and it is the fitness function used by the learning algorithm for selection.

The *real fitness* $f_C(X)$ is instead the generalisation error, measuring how well a given boolean expression X matches the complete truth table C. It is defined as follow

$$f_C(X) = \epsilon_g(X) = \sum_{i=1}^{N} D_i \cdot I[X(x_i) \neq y_i] \ , \ \forall (x_i, y_i) \in C$$

Its value is equal to 0 when the individual perfectly match the complete truth table C, while it has its worst value of 1 when the truth table is completely unmatched. Notice that the learning algorithm has *no access* to $f_C(X)$. It can use the fitness on the training set, $f_T(X)$, to measure the fitness of the current individual X but it cannot get the value of $f_C(X)$. This fitness function will be used in the following to define the generalisation ability of an algorithm.

1.3 Learning performance measures

Being f_T the fitness on the training set, and being f_C the real fitness we define the *generalization ability* of an evolutionary algorithm $\mathcal{A}$ as

$$G(\mathcal{A}) = E[1 - f_C(\widetilde{X})]$$

, where $\tilde{X}$ is the individual obtained after $\mathcal{A}$ has converged. The generalisation ability is in the range $[0, 1]$, and is the larger the better.

Notice that the definition of the generalisation ability depends just on f_C while the algorithm has just access to f_T. This means that the best generalisation performance will not necessarily be achieved by the algorithm which better optimise f_T. This is common knowledge in machine learning, where the effect of maximising f_T regardless of f_C would be considered as *overfitting*.

Previous work on the analysis of GSGP [1] have provided upper bounds for the expected runtime, which in our framework can be defined as:

$$R(\mathcal{A}) = E[\mathcal{T}(f_T(X) = 0)]$$

which is the expected time for the algorithm to find an individual with zero error on the training set since $\mathcal{T}(f_T(X) = 0)$ is the number of generations until the first individual with zero training error is produced. This is important to get guarantees on the convergence time of the algorithm but it has two important problems: it doesn't give any indication on the goodness of the algorithm as a learning tool and it wrongly bias the design of the algorithm to minimise $R(\mathcal{A})$ despite of $G(\mathcal{A})$. In fact good learning algorithm should have low convergence time, while still guarantee good values for $G(\mathcal{A})$.

2 Application of the framework

This framework can be applied to measure the generalization ability of the GSGP presented in [1]. In order to that one has to answer the question "Given a GSGP using $f_T(X)$ for selection and being $\widetilde{X}$ the best individual when the algorithm has stopped, what is the expected $f_C(\widetilde{X})$?". We did the calculations and we obtained that the Boolean Semantic Mutation (SGMB, or point mutation) presented at [1] needs a training set comprising at least half of all the possible inputs ($N/2$) in order to have a $G(SGMB) > 0$. On the other hand, Fixed Block Mutation (FBM) with a proper size of the blocks, can obtain a $G(FBM) > 0.5$ with a polynomial training set for a large enough problem size (n).

References

1. Alberto Moraglio, Andrea Mambrini, and Luca Manzoni. Runtime analysis of mutation-based geometric semantic genetic programming on boolean functions. In Frank Neumann and Kenneth A. De Jong, editors, *FOGA*, pages 119–132. ACM, 2013.