Probabilistic Smoothing of Genetic Programming

Ran Wei and John A. Clark

Department of Computer Science,
University of York, United Kingdom
ran.wei, john.clark@york.ac.uk

Abstract.

1 Introduction

2 Background and Motivation

3 Symbolic Regression: Deterministic

At first we discuss a case study which is an extended version of the multivalued symbolic regression problem provided as an tutorial for ECJ (Evolutionary Computation in Java)[1].

3.1 Representation

In the symbolic regression problem, there are six nodes representing the terminals nodes **X** and **Y**, and the function nodes **add** (+), **sub** (-), **mul** (*), and **div** (/).

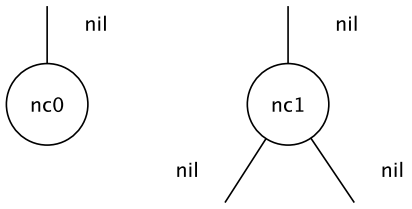

Fig. 1: Node constraints for the symbolic regression problem.

The terminal nodes conform to the node constraint *nc0*, and the function nodes conform to the node constraint *nc1* in Figure 1. *nc0* specifies that nodes

[1] https://cs.gmu.edu/~eclab/projects/ecj/docs/tutorials/tutorial4/index.html

which conform to it (Nodes **X** and **Y**) should have their return types as *nil*, which is ECJ's default type. *nc0* nodes should have 0 children. *nc1* specifies that the nodes which conform to it (Nodes **add**, **sub**, **mul**, and **div**) should have their return types as *nil*, *nc1* nodes should have two children, for which their return type should also be *nil*. The function **div** is added atop the example provided by ECJ and needs additional handling which will be discussed later.

3.2 Fitness

The objective of the symbolic regression is to evolve an equation which satisfies a set of $(X, Y, f(X, Y))$ data. In the current implementation, an upfront equation is provided: $x^2y + xy + y$. For each tree, the evaluation proceeds by giving each **X** and **Y** 20 random values and calculating their expected results (denoted by ev). Then, the tree generated by GP is evaluated (of which value is denoted by v). The *fitness* of the current generation is the sum of the the difference between ev and v. The optimal would be an equation which is equivalent to $x^2y + xy + y$. The introduction of the **div** function introduces the possibility of division by 0. To handle such situations, the *fitness* of the tree is set to the maximum value if a division by 0 is detected, such that a tree which contains division by zero is discarded very early in the evolution process.

We observe that the global optimal is obtained within 20 generations of evolution.

4 Symbolic Regression: Stochastic

With the extended symbolic regression implementation, we move on to the stochastic approach. In this approach, we introduce the concept of *stochastic node*, which is a versatile node that integrates the functions **add**, **sub**. **mul** and **div** in it. An probability is associated to each of the functions so that when a stochastic node is evaluated, the function of the node is determined (randomly) by their probabilities.

4.1 Representation

In our approach, there is only one function node: **OPNode**. There are two terminal nodes **X** and **Y**. In order to model the probability mechanism, we introduce four additional terminals: **One** (1), **Two** (2), **Three** (3) and **Four** (4). Instead of giving these nodes the *nil* type (ECJ's default type), we give these nodes the *int* type, so that they can be distinguished from the nodes of *nil* type in the tree.

To implement the stochastic node we define the node constraints shown in Figure 2. *nc0* is unchanged as it is used for terminals **X** and **Y**. *nc1* specifies that the nodes which conform to it should have no children and their return types as *int*. In our approach, nodes **One**, **Two**, **Three** and **Four** conform to *nc1*. *nc2* specifies that the nodes which conform to it should have 2 nodes of

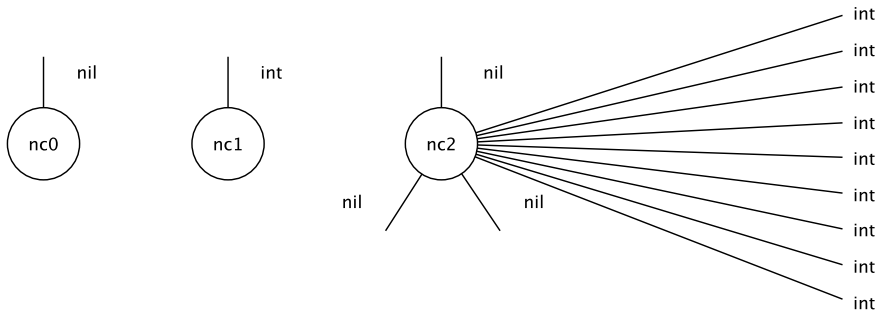

Fig. 2: Node constraints for the symbolic regression problem.

type *nil*, 10 nodes of type *int* and its own return type as *nil*. In our approach, the function nodel **OPNode** conforms to *nc2*.

4.2 Function weightings and stochastic costs

The probability and the selection of the function of an *OPNode* is explained as follows;

- Each *OPNode* keeps track of the *weight*s of functions **add**, **sub**, **mul** and **div**, denoted by W_{add}, W_{sub}, W_{mul} and W_{div}. Initially they are all 0.
- When an *OPnode* is evaluated, the ten nodes of type *int* are evaluated first. When node **One** is encountered, W_{add} is increased by 1. W_{sub}, W_{mul} and W_{div} are increased by 1 if nodes **Two**, **Three** and **Four** are encountered respectively.
- The probability of the functions are calculated based on their weights. In particular, let P_{add}, P_{sub}, P_{mul} and P_{div} denote the probabilities of **add**, **sub**, **mul** and **div**. Such that

$$P_{add} = W_{add}/(W_{add} + W_{sub} + W_{mul} + W_{div})$$
$$P_{sub} = W_{sub}/(W_{add} + W_{sub} + W_{mul} + W_{div})$$
$$P_{mul} = W_{mul}/(W_{add} + W_{sub} + W_{mul} + W_{div})$$
$$P_{div} = W_{div}/(W_{add} + W_{sub} + W_{mul} + W_{div})$$

- The function of the *OPNode* is then selected randomly by the probabilities P_{add}, P_{sub}, P_{mul} and P_{div}.
- Each *OPNode* is also associated with a *stochastic cost*, denoted by $N_{stochastic}$, such that:

$$N_{stochastic} = 1 - max(P_{add}, P_{sub}, P_{mul}, P_{div})$$

to calculate the stochastic cost of a tree ($T_{stochastic}$), in our approach, we use the sum of the *stochastic cost*s of all the *OPNode*s in a tree:

$$T_{stochastic} = \sum\nolimits_{N \in OPNodes} N_{stochastic}$$

4.3 Fitness

The objective of the stochastic symbolic regression is the same as the symbolic regression. In our approach we use the same experiment settings (same upfront equations, 20 randomised values for each $\mathbf{X}$ and $\mathbf{Y}$, and their expected results ev). In addition, we add another search objective: to minimise the *stochastic cost* ($T_{stochastic}$) of the tree. Let v denote the value retured by evaluating a tree, we define our secondary fitness:

$$fitness = abs(ev - v) + T_{stochastic} \tag{1}$$

During the evolutionary process, for each pair of $\mathbf{X}$ and $\mathbf{Y}$, the tree is evaluated 100 times and the lowest fitness cost is kept.

Through experimentation, we observe that the weightings of the objectives are significant to the evolutionary process. We observe that if the two objectives share the same weighting (as in Equation 1), the stochastic cost dominates the evolution, causing the main tree (the tree containing *OPNode*s only, not the subtrees of each *OPNode*) to stop evolving from early generations. We then search for the weightings of the two objectives and observe that the factor of 0.01 should be applied to the stochastic cost in order to balance the two objectives. Thus, we refine our fitness:

$$fitness = abs(ev - v) + 0.01 * T_{stochastic} \tag{2}$$

With the refined fitness, we observe that the global optimal is obtained within 80 generations of evolution. The stochastic cost of the final tree is zero as a result of evolution, this means that each node in the tree is deterministic, i.e. it only has one function every time it is evaluated.

5 The Occupancy Classification Problem: Deterministic

In this section we adapt our approach to the study of the classification problem presented in [1]. In this study, data collected from sensors in a room, such as temperature, humidity, light, and CO_2 levels (and other fields derived from these statistics) are used to determine if the room is occupied. The data and the occupancy are used to train the classification models. In this section, we first present the classification model we implemented using ECJ, we then adapt our approach to the classification problem and evaluate our hypothesis.

5.1 Representation

We apply our approach used in the symbolic regression problem to the classification problem. This is an inconventional approach as we are trying to solve a classification problem using regression methods. In this section, we discuss the problem representation.

In the occupancy classification problem, there are eight different types of data that are used in the study: *temperature*, *humidity*, *light*, CO_2, *humidity ratio*, *number of seconds from midnight* (denoted by *nsm*), week status (denoted by *ws*) and *occupancy*. In our representation, there are seven terminal nodes **T** (temperature), **H** (humidity), **L** (light), **HR** (humidity ratio), **NSM** (number of seconds from midnight) and **WS** (week status). There are also four function nodes **add** (+), **sub** (-), **mul** (*), and **div** (/).

The node constraints are the same as it is illustrated in Figure 1 in the symbolic regression problem, where nodes **T**, **H**, **L**, **HR**, **NSM** and **WS** conform to *nc0* and nodes **add**, **sub**, **mul**, and **div** conform to *nc1*.

5.2 Fitness

The objective of the classification problem is to determine if a room is occupied with the data. In our implementation we use a threshold approach. In the training data provided[2], there are 8143 data entries. We obtained the threshold (denoted by θ) by summing the means of all types of data:

$$\theta = \overline{T} + \overline{H} + \overline{L} + \overline{HR} + \overline{NSM} + \overline{WS}$$

When the evaluated value of a tree (denoted by v) is obtained, we compare v with the threshold θ to form our functional cost ($cost_f$):

$$v \leq \theta \Rightarrow cost_f = 0;$$
$$v > \theta \Rightarrow cost_f = 1;$$

In order to guarantee that all seven terminals are used, we check for existence of the terminals regardless of repetitions (denoted as the parameter cost $cost_{param}$). If all seven terminals are used, $cost_{param}$ is 0, otherwise $cost_{param}$ equals the number of terminals that are not used.

We then define our fitness:

$$fitness = cost_f + cost_{param}$$

With our implementation we managed to get 99.19% accuracy within 116 generations of evolution in ECJ. However, the accuracy is not able to improve any further.

[2] https://archive.ics.uci.edu/ml/datasets/Occupancy+Detection+

6 The Occupancy Classification Problem: Stochastic

We now discuss the stochastic approach for the occupancy classification problem. We adapt our approach in solving the symbolic regression problem, as discussed in Section 4. For the classification problem, we use the same *OPNode* which integrates the functions **add**, **sub**. **mul** and **div** in it.

6.1 Representation

For the classification problem, we re-use the seven terminal nodes introduced in Section 5, i.e. **T** (temperature), **H** (humidity), **L** (light), **HR** (humidity ratio), **NSM** (number of seconds from midnight) and **WS** (week status).

The node constraints are the same as it is illustrated in Figure 2 in the symbolic regression problem, where nodes **T**, **H**, **L**, **HR**, **NSM** and **WS** conform to *nc0* and *OPNode* conforms to *nc2*.

6.2 Fitness

Same as the deterministic approach to the occupancy classification problem, we use a threshold approach. We obtain the threshold (denoted by θ) by adding the means of all types of data:

$$\theta = \overline{T} + \overline{H} + \overline{L} + \overline{HR} + \overline{NSM} + \overline{WS}$$

When the evaluated value of a tree (denoted by v) is obtained, we compare v with the threshold θ to form our functional cost ($cost_f$):

$$v \leq \theta \Rightarrow cost_f = 0;$$
$$v > \theta \Rightarrow cost_f = 1;$$

To guarantee that all seven terminals are used, we check for existence of the terminals regardless of repetitions (denoted as the parameter cost $cost_{param}$). If all seven terminals are used, $cost_{param}$ is 0, otherwise $cost_{param}$ equals the number of terminals that are not used.

We define our preliminary fitness:

$$fitness = cost_f + cost_{param} \tag{3}$$

Since we also introduce stochasticity in the problem, we also need to minimise the *stochastic cost* ($T_{stochastic}$) of the tree. Thus, we refine our fitness function:

$$fitness = cost_f + cost_{param} + T_{stochastic} \tag{4}$$

During the evolutionary process, for each set of data, the tree is evaluated 10 times (due to the huge amount of calculation) and the lowest fitness cost is kept.

In addition, we normalise the weightings of the three objectives (cost function, parameter cost function and stochastic cost function). After experimentations we observe that the best results are obtained when the factor of 0.01 is applied to both the parameter cost function and stochastic cost function. Thus, we refine our fitness:

$$fitness = cost_f + 0.01 * cost_{param} + 0.01 * T_{stochastic} \tag{5}$$

Due to the amount of calculations needed, we limit the evolution to 1000 generations. We observe that the stochastic approach is able to achieve 84.19% accuracy within 1000 generations of evolution. We believe that the accuracy will improve as the *OPNodes* in the tree are not-but-close-to deterministic. We anticipate that the stochastic approach is applicable to the occupancy classification problem if sufficient resources are provided.

References

1. Luis M. Candanedo and Veronique Feldheim. Accurate occupancy detection of an office room from light, temperature, humidity and CO_2 measurements using statistical learning models. *Energy and Buildings*, 112:28 – 39, 2016.