SUPPLEMENTARY MATERIAL for Learning a Formula of Interpretability to Learn Intepretable Formulas

Marco Virgolin[1], Andrea De Lorenzo[2], Eric Medvet[2], and Francesca Randone[3]

[1] Centrum Wiskunde & Informatica, Amsterdam, the Netherlands
[2] Department of Engineering and Architecture, University of Trieste, Trieste, Italy
[3] IMT School for Advanced Studies, Lucca, Italy

Details on the survey

In this supplementary material, we provide full details on the way we generated the survey aimed at collecting human-feedback on the Proxies of Human Interpretability (PHIs) called *simulatability* and *decomposability* [1].

As mentioned in the manuscript, no established protocol exists to measure simulatability and decomposability. In formulating the questions, we attempted to be consistent with the definitions and type of measurements used in previous work (mainly [3] for simulatability), while adapting them to the context of an online survey. We proposed two kinds of simulatability questions that were aimed at making the user replicate the computations of the function presented. Decomposability questions, on the other hand, were designed to asses if the user was able to evaluate how a single variable influenced the behaviour of the whole formula in a qualitative manner.

Together with the challenge of designing proper questions, another challenge was represented by the generation of a set of formulas apt to cover a sufficiently wide range of different features. We began with generating the formulas at random, and followed by automatic refinements, to get rid of uninteresting formulas, formulas with non-real output, etc. We begin by explaining formula generation.

Formulas generation

We initially generated a large number (200 000) of random formulas, as follows. The formulas were encoded by trees which were generated with the half-and-half initialization method of tree-based GP [2], with max depth of 4. Resulting formulas can have up to 4 variables and use operations taken from the set $\{+, -, \times, \div, ^\wedge, \sqrt{\cdot}, \sin, \cos\}$.

We subsequently performed rejection sampling and simplifications, to ultimately obtain a set of 1000 formulas, as follows. We used rejection sampling to avoid presenting the users with formulas that:

- contained no variable, as they represented uninteresting constant functions;
- with more than two power ($^\wedge$) operations, to avoid potentially requiring the user to calculate very large numbers;

– led to complex (i.e, not real) numbers.

Next to rejection sampling, we applied the following simplifications to avoid the presence of cumbersome computations:

– we replaced the arguments of goniometric functions with either $x + \text{const}$ or $x \times \text{const}$, with $\text{const} \in \{1.57, 3.14, 6.28\}$ (note that the maximum depth of the tree encoding became 5);
– we set all constants to integer numbers or 0.5;
– we excluded expressions such as $f(x)^{g(x)}$ if f and g are both non-constant and g is not the identity.

Finally, we made further use of rejection sampling to partition the formulas evenly based on input dimensionality, i.e., ultimately 25 % formulas with 1 variable, 25 % with 2 variables, 25 % with 3, and 25 % with 4.

Generation of simulatability questions

As stated in the main paper, simulatability questions can be of two types depending on the dimensionality of the formula.

For one variable formulas, we prepared *graphical* questions. Given the formula, we showed 4 possible interactive plots of its behavior for $x \in [-10, 10]$.

If the original function represented by the formula was (i.e., the formula contained goniometric operations), the alternative (wrong) graphs were taken by considering other goniometric functions. Otherwise, the other options were:

– the graph of the opposite function;
– a random one variable non-goniometric function;
– the opposite of a random one variable non-goniometric function.

This is done to guarantee a sufficient diversity between the graphs, while not exposing the correct option as an obvious one. The order of appearance of the graphs is randomized.

For multivariable formulas we generated a random input vector of values for the variables, and asked the user to pick the option that most closely resembled the actual output of the function. The input of values for the variables was initially generated at random, as a vector of random integers between 0 and 9, and divided by 10 with a probability of 0.5. Moreover, the following rules were applied:

– if the formula contained expressions of the type $\sin(x_i + \text{const})$ or $\cos(x_i + \text{const})$, the input value for the variable x_i was chosen uniformly at random between 0, 1.57, 3.14, 6.28;
– if the formula contained expressions of the type $\sin(\text{const} \times x_i)$ or f^{x_i}, then the input value for x_i was set to a random integer between 0 and 9.

We discarded the cases in which the rules would conflict (i.e., when both applied to some x_i).

Once the input vector was generated, the formula was evaluated on it, and a check was performed on the output. Specifically, we constrained the output to be a real number with absolute value larger than 0.0001 and smaller than 10 000. If the output did not satisfy this, three other attempts of generating the input vector were performed. If all failed, the formula was discarded.

Regarding the three wrong options for the answer, the following rules were used to generate them:

– if the correct output was zero, the other options were chosen uniformly at random between 1, −1, 5, −5, 10, −10;
– otherwise, the other options were obtained by multiplying the correct output by a coefficient chosen uniformly at random between −0.1,−0.5,−2, −10, 0.1, 0.5, 2, 10.

As before, this was done to guarantee a sufficient difference between the proposed outputs without making the correct option too obvious. The alternative outputs and the correct one were finally rounded to two decimal figures, and their order of diplay shuffled.

Generation of decomposability questions

Decomposability questions were generated starting from the same formulas considered in simulatability questions.

For one variable formulas, the interval of variation for the variable to consider was always set to $[-1, 1]$. In the case of multi-variable formulas, the input vector used for the respective simulatability question was retrieved, then the question was formulated by letting the first variable vary in an interval of amplitude 2 centered in the original value of the variable as present in the input vector.

An automatic check the behavior of the function so obtained was performed. In particular, we checked:

– if the function was defined in all points of the given interval;
– if the function was bounded on the interval;
– if the function was zero at some points of the interval;
– if the function was negative at some points of the given interval;
– what the range of the function was on the given interval.

If the function was not defined on the whole interval, our automatic check was only able to asses definedness and boundedness of the function. Because of this, we actually used two different kind of questions. If the function was not defined on the whole interval of variation of $x_\star$, the user was asked to pick the correct answer among the following options:

```
a. The function is bounded and always defined
b. The function is bounded but not always defined
c. The function is not bounded but always defined
d. The function is not bounded and not always defined
```

If the function was instead defined on the whole interval, then the above choices were only supplied for half of the times. The other half, we proposed a question in which the possible options covered all the features retrieved with the automatic check. As for all other questions, we wanted only one of the proposed options to be correct. Hence, we formulated wrong options by negation of the correct property. For example if a function was defined on the whole interval, was not bounded, was zero at some point and was never negative, a possible set of options was:

```
a. The function is never zero
b. The function is not defined on the whole interval
c. The function is not bounded
d. The function is negative at some point
```

Note that the only correct answer is `c`.

References

1. Lipton, Z.C.: The mythos of model interpretability. Queue **16**(3), 31–57 (Jun 2018)
2. Poli, R., Langdon, W.B., McPhee, N.F., Koza, J.R.: A field guide to genetic programming. Lulu. com (2008)
3. Poursabzi-Sangdeh, F., Goldstein, D.G., Hofman, J.M., Vaughan, J.W., Wallach, H.: Manipulating and measuring model interpretability. arXiv preprint arXiv:1802.07810 (2018)