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

(A) Details on the survey

We designed a survey to evaluate how a user perceives mathematical formulas exhibiting different features, in terms of simulatability and decomposability.

In doing so a major problem was that, as already noticed, no established protocol exists to measure these proxies. For this reason, in formulating the questions we focused on being consistent with the definitions and maintaining those approaches that had already shown good results in past works (mainly [2] for simulatability). As a result, we proposed two kinds of simulatability questions that forced the user to replicate the whole computation mechanism 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 influences the behaviour of the whole formula, qualitatively.

Another problem was generating a set of formulas apt to cover a sufficiently wide range of different features. In order to solve this, we decided to generate the formulas randomly. However, since a purely random generation could result in a great number of uninteresting formulas, far from our potential applications (e.g. complex-valued formulas), we introduced some further restrictions, listed below.

We believe that the survey obtained in this way can represent a good attempt to measure simulatability and decomposability of a wide range of mathematical formulas.

Formulas generation

The questions are based upon 1000 random formulas that were generated as follows. The formulas were encoded by trees which were generated with the Half-and-Half initialization method of tree-based GP [1], 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 performed rejection sampling, to avoid presenting the users with formulas that:

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

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.

We verified that generating 200 000 formulas is sufficient to guarantee a final dataset of 1000 elements that meet requirements.

Generation of simulatability questions

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

For graphical questions (involving one variable formulas) we relied on another database of one variable functions, from which we have drawn the graphs used as alternative options. Such database was obtained generating one-variable formulas and converting them into symbolic expressions with the exact same procedure used for the other formulas. In addition, we introduced further restrictions on the range of the functions in the domain $[-10, 10]$ in order to guarantee an acceptable graphical rendering.

If the original function is goniometric, graphs of other goniometric functions are selected as options. Otherwise, if the original function does not contain goniometric functions, the other options are generated choosing (1) the graphs of the opposite function, (2) a random non-goniometric function drawn from the one-variable database and (3) its opposite. This is done to guarantee a sufficient difference between the graphs, without making the correct option too obvious. The alternative functions and the correct one are then randomly assigned to the four options.

For multivariable formulas we generated a random input vector, following these criteria:

- if the formula contains expressions of the type $\sin(x+\text{const})$ or $\cos(x+\text{const})$ the associated variable is chosen randomly between 0, 1.57, 3.14, 6.28;
- if the formula contains expressions of the type $\sin(\text{const} \times x)$ or f^x the associated variable is a random integer between 0 and 9;

– in all the other cases the input vector is a vector of random integers between 0 and 9, and it is divided by 10 with a probability of 0.5.

Once the input vector has been generated, the formula is evaluated in it and a check is performed on the output. We ask for the output to be a real numerical value larger in absolute value than 0.0001 and smaller than 10 000. If the output does not satisfy these constraints three other attempts of generating a random vector are performed. After the fourth failed attempt the formula is discarded.

In all the other cases other three options are generated. If the correct output is zero the other options are chosen randomly between 1, −1, 5, −5, 10, −10. Otherwise, the other options are obtained multiplying the correct answer by three coefficients chosen randomly between −0.1,−0.5,−2, −10, 0.1, 0.5, 2, 10. As before, this is done to guarantee a sufficient difference between the proposed outputs without making the correct option too obvious. The alternative outputs and the correct one are then rounded to two decimal figures and randomly assigned to the four options.

Generation of decomposability questions

As explained in the manuscript, decomposability questions are based upon the same formulas considered in simulatability questions. For one variable formulas, the interval of variation for the variable to consider is always set to $[-1, 1]$. In the case of multivariable formulas, the input vector used for the respective simulatability question is retrieved, then the question is formulated letting the first variable varying in an interval of amplitude 2 centered in the original value of the variable. All other variables remain fixed to the previous input values.

An automatic check the behavior of the formula so obtained is performed. In particular, we check:

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

If the function is not defined on the whole interval, our automatic check is only able to asses definedness and boundedness of the function. Due to this fact we use two different kind of questions.

If the function is not defined on the whole interval of variation of $x_\star$, the user is 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 is instead defined on the whole interval, then half of the times the question is different.

In particular we formulate a question in which the possible options cover all the features retrieved with the automatic computation. Only one of the option proposed is correct, while the others are formulated by denying the correct property. For example if a function is defined on the whole interval, is not bounded, is zero at some point and is never negative, a possible set of options is:

```
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
```

where the only correct answer is `c`.

Dissemination and results

We implemented the survey as a webpage[4], consisting of an introductory section (to assess familiarity with the topic), followed by eight questions, four about simulatability, and four about decomposability. The eight questions are randomly selected when the webpage is loaded, from the pre-generated databases containing 1000 questions each.

We distributed the survey by emailing research groups and departments within the institutes of the authors, targeting both students and faculty members. We further shared the survey on Reddit (subreddit CasualMath) and Twitter.

After shortly longer than one month, we collected a total of 334 completed surveys, corresponding to 2672 answers. All data collected from the survey answers are available online[5].

References

1. Poli, R., Langdon, W.B., McPhee, N.F., Koza, J.R.: A field guide to genetic programming. Lulu. com (2008)
2. 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)

[4] `http://mathquiz.inginf.units.it/`
[5] `https://github.com/MaLeLabTs/GPFormulasInterpretability`