Algorithm Selection for Symbolic Integration using Machine Learning

Rashid Barket[1], Matthew England[1], and Juergen Gerhard[2]

[1] Coventry University, Coventry, United Kingdom
[2] Maplesoft, Waterloo, Canada

Abstract

Computer Algebra Systems (e.g. Maple) are used in many contexts including research, education, and in industrial settings. One of the key algorithms, symbolic integration, has various sub-algorithms used that can affect the form of the output integral or the runtime. We first discuss new data generation methods for integrals that generate a richer variety of training data. Then, various ML models are trained to select a sub-algorithm that outputs the optimal expression for the integral. Not only is the model able to produce more optimal outputs compared to Maple's state-of-the-art meta-algorithm, but it also outputs substantially more uniquely optimal answers compared to Maple. Furthermore, we show that the models are able to outperform Maple on an external dataset showing the model's generalisation abilities.

1 Introduction

A Computer Algebra System (CAS) is known for its precise calculations. If a CAS were to make an incorrect calculation, it would be detrimental to both the user and the CAS itself. On the other hand, ML has a probabilistic nature and tends to predict what an answer to a problem may be. Thus, it seems as though ML and CASs would have little to no interaction. However, for many implemented CA algorithms in a CAS, there is usually a choice that can be made. When an algorithm has to make a choice, this is either done in a pre-set order or a deterministic heuristic. With the way these choices are made, computation time could be wasted or non-optimal answers may be produced.

This is certainly the case for Maple's symbolic integration algorithm. Maple's integration algorithm is essentially a meta-algorithm: there are 12 different possible sub-algorithms to use for indefinite integration. Each can produce very different, but mathematically equivalent answers. Some can also take much longer to execute than others.

We train a TreeLSTM and LSTM model to select a sub-algorithm that produces the optimal length answer. This is important to a Maple user as some answers produced look much more complex/convoluted compared to a simpler answer that are available from different sub-algorithms. We show that the TreeLSTM model has an improvement over Maple's meta-algorithm despite training on a small dataset. Even though we do not optimise over the runtime, we also show that both the LSTM and TreeLSTM are competitive in execution time with Maple's meta-algorithm.

2 Related Work

Perhaps the most relevant experiment to our problem was done by Lample & Charton [1] who trained a transformer to calculate integrals and solve differential equations directly. A dataset is created with three different data generators and the transformer is trained on a large quantity of (integrand, integral) pairs and the differential equations and their solutions. One major breakthrough of their approach is the fact that they are able to calculate integrals that CASs like Maple and Mathematica were not able to do on their testing set. Critiques have been brought up regarding their data generation methods [2] and while the authors are able to achieve good results, this approach essentially remains a black box, giving no indication into why some integrals are easier to integrate than others.

A prime example of ML being used to do algorithm selection is shown by Simpson et al. [3] who train an ML model to select the best of multiple algorithms for calculating the resultant of two polynomials in Maple and Mathematica. The ML model trained to select a resultant algorithm reduced the computation time in Maple by 67% in Maple and 49% in Mathematica compared to the implementations for selecting a resultant algorithm done in each CAS.

3 Algorithm Selection

We train both LSTMs and TreeLSTMs to select the best sub-algorithm that outputs the smallest length integral. The outputs from the model are then compared to the answer that Maple's meta-algorithm selects. To evaluate how well our models perform compared to Maple, we conduct two experiments. The first experiment is done on a hold-out test set that was separate from the training data. The TreeLSTM was able to predict 85.1% of optimal answers compared to 75.9% and 73.2% from the LSTM and Maple's meta-algorithm respectively. We see even more substantial results when we look at unique optimal predictions: The TreeLSTM was able to produce almost 8x more unique optimal answers compared to Maple and the LSTM.

While it is great that the models succeeded on the test set, it is important to look at how they do on externally provided data. This would show if the model is only good at classification for data coming from the same data generation method, or if the model is generalisable. To this end, we test the models on a dataset provided by Maplesoft, creators of Maple. The Maple test suite paints a slightly different picture compared to the test data from the generators. In terms of total optimal answers, TreeLSTM very slightly outperforms the LSTM and Maple's meta-algorithm. However, Maple can produce slightly more uniquely optimal answers compared to the other two models. One interesting fact to note is that if we compare the TreeLSTM model to Maple directly (leaving LSTM out), the TreeLSTM produces more uniquely optimal answers to Maple at 633 to 578 examples. The TreeLSTM model is able to slightly outperform Maple's meta-algorithm on both metrics, demonstrating the model's generalisation abilities on a small training set.

References

[1] Guillaume Lample and François Charton. Deep learning for symbolic mathematics. In *Proc. International Conference on Learning Representations (ICLR)*, 2020.

[2] Bartosz Piotrowski, Josef Urban, Chad E. Brown, and Cezary Kaliszyk. Can neural networks learn symbolic rewriting? In *Proc. Artificial Intelligence and Theorem Proving (AITP)*, 2019.

[3] Matthew C. Simpson, Qing Yi, and Jugal Kalita. Automatic algorithm selection in computational software using machine learning. In *15th IEEE International Conference on Machine Learning and Applications (ICMLA)*, pages 355–360, 2016.