Deep Learning for symbolic mathematics

presentation by Mikhail Liz

March 2020

Content

- Introduction.
- Deep Learning model.
- Mathematical expressions as sequences.
- Data Generation.
- Results.

Introduction

Goals and definitions

Goals

- Integrating a function
- Solving a first order or second order ordinary differential equation (ODE)

Typically the following notation is used

- f is an arbitrary function, its integral is F
- F is an arbitrary function, its derivative is f

Introduction

The particular problems

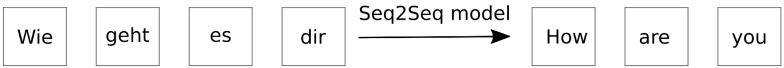
Answers must be completely correct or incorrect.

$$
\begin{aligned}
\int 3 \cdot x \quad dx &\longrightarrow \frac{3}{2} \cdot x \cdot x + c && ✓ \\
&\longrightarrow \frac{3 \cdot x^2}{2} + c && ✓ \\
&\longrightarrow \frac{3}{2} \cdot x + x + c && ✗
\end{aligned}
$$

Deep Learning model
Seq2Seq

Seq2Seq model was used for this task. The most important properties of this model for the task of function integration are:

- Both input and output sequence can have arbitrary length and those lengths can differ
- A one-to-one relation from item in input sequence to item in output sequence is not necessary



Input sequence in german, Output sequence translated to english.

Mathematical expressions as sequences
Infix notation

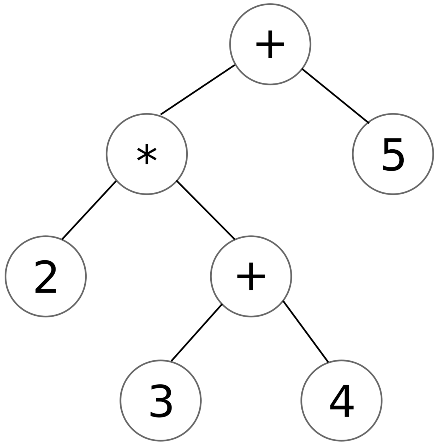
Mathematical expressions already are in sequence form, the so-called infix notation. However, this notation is not very well suited for processing in a seq2seq model. This is because to make the order of the operations unambiguous, a lot of parenthesis are needed:

$$2 \cdot (3 + 4) + 5$$
$$(2 \cdot (3 + 4)) + 5$$

Top: Commonly used infix-notation. Bottom: Parenthesis for unambiguous order of operations

Mathematical expressions as sequences

Expressions as trees



Mathematical expressions as sequences

Prefix notation

The tree is traversed from top to bottom and from left to right with the following rules:

- If the current node is a primitive value (a number), add it to the sequence string
- If the current node is a binary operation, add the operations symbol to the sequence string. Then, add the representation of the left child node (could be recursive). Then, add the representation of the right child node.

$$\underbrace{+\underbrace{*2\underbrace{+3\ 4}_{3+4}}_{2*(3+4)}\ 5}_{2*(3+4)+5}$$

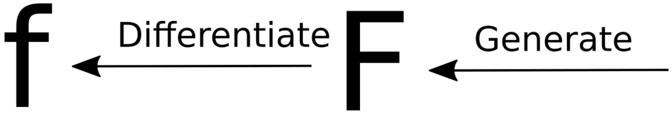
Data Generation
Forward Generation

Randomly generate a symbolic function f, then use an external tool to compute the symbolic integral F. Pairs (f, F) produced by this method have the property, that f often consists of much fewer symbols than F.

$$\xrightarrow{\text{Generate}} \mathrm{f} \xrightarrow{\text{Integrate}} \mathrm{F}$$

Data Generation
Backward Generation

Randomly generate a function F, then automatically differentiate it to receive function f. The pair (f, F) is then as training sample. Pairs (f, F) produced by this method have the property, that F most often consists of fewer symbols than f.



Data Generation

Integration By Parts

The algorithm for generating functions with integration by parts:

1. Randomly generate functions F and G
2. Automatically differentiate to obtain f and g
3. If Fg is already part of the training set, we know its integral. Compute the integral of fG using the formula above
4. If fG is already part of the training set, we know its integral. Compute the integral of Fg using the formula above

$$\int Fg = FG - \int fG$$

Data Generation

Additional steps

Simplification

Expressions are simplified to enforce that the model also outputs simplified expressions.

Removing invalid expressions

Due to the randomly generated nature of the expressions, they might contain invalid sub-terms. If they do, remove those terms from the expressions tree.

Results

Testing on single generators

Using training data that was generated by only one of FWD, BWD or IBP and evaluating the model on data generated from the same method gives very good results:

- FWD: 96,2% accuracy
- BWD: 99,7% accuracy
- IBP: 99,5% accuracy

Results

Comparing to popular mathematical frameworks

Comparison the results with popular mathematical frameworks using only the BWD generator:

- Mathematica (with 30 second timeout): 84.0%
- Matlab: 65.2%
- Maple: 67.4%
- Seq2Seq: 99.6%

Results

Comparing across generators

Models were evaluated on one generator and tested on other generators:

- Trained on FWD: 17.2%(BWD), 88.9%(IBP)
- Trained on BWD: 27.5%(FWD), 59.2%(IBP)
- Trained on BWD+IBP: 56.1%(FWD)
- Trained on FWD+BWD+IBP: 94.3%(FWD), 99.7%(BWD), 99.7%(IBP)

Thank you for your attention!