Project Proposal:
Neural Modelling of Mathematical Structures*

Martin Smolík and Josef Urban

[1] Charles University
[2] Czech Technical University, Prague,

Introduction and planned experiments

In this project we will focus on the usage of deep learning in building a "mathematical intuition" for a machine. The ultimate goal is to make a neural network that will attempt to distinguish between true and false statements about a mathematical structure, based on training on a provided knowledge base of true and false statements. For more complicated mathematical structures such as natural and real numbers, set-theoretical universes, or just any very large finite object, our knowledge base may be incomplete, infinite and/or non-computable. However, we will first mainly focus on a few specific cases, where we will try to build a network that "understands" group operations or real numbers.

Initially, we will focus on small finite structures, in particular groups, and try to build networks that will emulate the group operations $\cdot$ and $^{-1}$, as well as the constant e. The sequence of planned experiments is roughly as follows:

- Learn finite structures directly from their multiplication/interpretation table(s).
- Learn finite structures from sets of (universally) quantified sentences.
- Learn finite structure from sets of randomly chosen or generated sentences.
- Learn infinite structures in a similar way.

At each point we can evaluate the usefulness of the trained neural approximations directly on a knowledge base of true/false statements, but also in more complex tasks such as estimating the truth of conjectures generated automatically by various conjecturing methods.

Experiments done on finite groups so far

Short introduction to groups: A group is a mathematical structure that has 2 operators (in our case called "composition" - denoted $\cdot$ and "inverse" - denoted $^{-1}$) and one constant, called "unit" - e. These operators and the constant must satisfy some axioms (for any elements).

- Associativity: $(x \cdot y) \cdot z = x \cdot (y \cdot z)$
- Unit: $x \cdot e = x = e \cdot x$
- Inverse: $x \cdot x^{-1} = e = x^{-1} \cdot x$

Learning of the composition operator: Our main focus are now on small permutation groups. Composition operator is learned from the multiplication table. We construct a multilayer feedforward neural network that tries to estimate the result. For this we choose a grounding - a representation of the group elements as vectors of a chosen size. We construct training data by taking groundings of pairs of random elements and we pre-compute the result of the composition. The network processes the input pairs, we measure the error and backpropagate. Because we use the multiplication table, our data will always satisfy the associativity axiom.

*Supported by the ERC Consolidator grant no. 649043 AI4REASON and by the Czech project AI&Reasoning CZ.02.1.01/0.0/0.0/15_003/0000466 and the European Regional Development Fund.

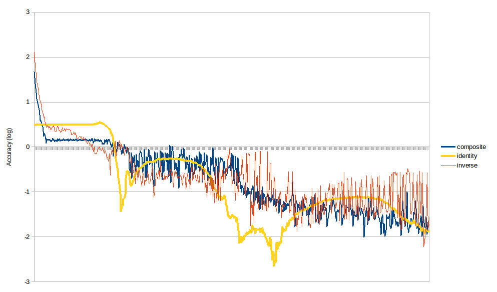

Figure 1: Accuracy during training for S_3.

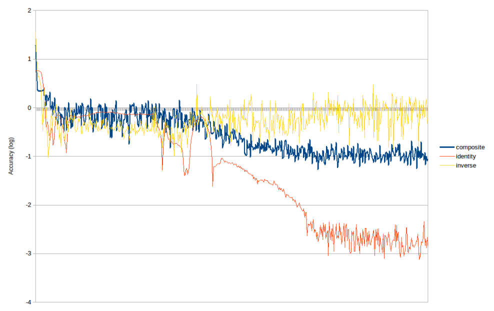

Figure 2: Accuracy for S_4.

Learning of the unit and inverse operator: The unit and the inverse operator are trained differently. We try to learn them by trying to satisfy universally quantified formulas. We pre-train the composition operator, and we use it to train the networks for the unit and the inverse operator. Unit is represented as a single (learned) tensorflow variable. Using the composition operator and a random element x, we estimate $x \cdot e$. This should always be equal to x. We measure the error and adjust our estimation of e accordingly. Here we do not change any variables in the pre-trained composition operation. For the inverse operator we also use this method with the inverse axiom. Inverse operator, like composition, is a multilayer feedforward neural network. For a random element x we estimate $x \cdot x^{-1}$. This should be equal to our estimated value of e. We therefore measure the error and backpropagate in the inverse's network. We once again do not touch the composition nor unit.

Results

We have built a framework in tensorflow [1] that supports learning from both tables and universally quantified formulas. In Figure 1 we can see the accuracy of operations in a group S_3 in one of the early implementations. Training consisted of almost 100 000 epochs. All operations were trained at the same time, although optimizers were able to alter only their specific variables. Because of implementation difficulties in tensorflow the identity and inverse were trained in batches of 1. That is why in case of the inverse operator the accuracy has such a high variation.

So far we have used a very basic grounding, where each permutation is represented in $\mathbb{R}^3$ by its one-line notation, e.g. the identity is represented by the vector $[0, 1, 2]$. The loss function that was used was $(|v_c - v_e|_{L_1})^2$ where v_c is computed and v_e is the expected value. Such a grounding and loss function are however not optimal and may result in large inaccuracies when working with larger groups. The accuracy is computed as $|v_c - v_e|_{L_1}$ with v_c and v_e as above. The metric used is the L_1 metric. We use base 10 logarithm, which means that in the end the error was close to 0.01.

Figure 2 shows the graph for training of the group S_4 with similar groundings and loss functions as for S_3. The training went for 1 000 000 epochs. We can see that the inverse operator is severely limited by the size of the training batch. We can also see that the identity is very accurate despite having the same disadvantage. The talk will discuss further experiments with training groups and other mathematical structures.

The networks described here use 4 hidden layers with 9 nodes each. The activation function is leaky ReLu. The optimizer is a default tensorflow Adam optimizer, therefore the learning rate is quite low. Our project's goal also includes finding better architectures.

References

[1] Martín Abadi, Paul Barham, Jianmin Chen, Zhifeng Chen, Andy Davis, Jeffrey Dean, Matthieu Devin, Sanjay Ghemawat, Geoffrey Irving, Michael Isard, Manjunath Kudlur, Josh Levenberg, Rajat Monga, Sherry Moore, Derek Gordon Murray, Benoit Steiner, Paul A. Tucker, Vijay Vasudevan, Pete Warden, Martin Wicke, Yuan Yu, and Xiaoqiang Zheng. Tensorflow: A system for large-scale machine learning. In Kimberly Keeton and Timothy Roscoe, editors, *12th USENIX Symposium on Operating Systems Design and Implementation, OSDI 2016, Savannah, GA, USA, November 2-4, 2016.*, pages 265–283. USENIX Association, 2016.