Reinforcement learning of mathematics in formal systems: a research statement

Junyan Xu

October 2018

In my view (as a mathematician), proof assistants aim to formalize, verify, organize and disseminate the vast body of mathematical knowledge. These goals certainly sound attractive to a mathematician, but since proof assistants so far are not intelligent enough for the formalization process to be well automated, and due to lack of library support, even the efforts required to formalize basic mathematics can be discouraging, let alone cutting-edge research in modern mathematics. Moreover, formalization of mathematics doesn't usually help with the understanding of and research in mathematics. These factors have prevented average mathematicians from using proof assistants regularly, formalizing their work to be automatically reviewed by proof checkers, and contributing to the development of proof assistants and their libraries. For most mathematicians, proof assistants only serve as the last resort when their research have trouble getting recognized (e.g. Hales' proof of Kepler's conjecture).

Efforts have been devoted to making proof assistants more intelligent and automated, in the hope to attract more mathematicians to enter a positive feedback loop of increased adoption and more comprehensive libraries. Tactics based on explicit hand-crafted rules are developed to enable goal-directed backward reasoning (but not exploration without specific goal). Machine learning methods have been applied to automate e.g. the selection of tactics, and libraries provide the training data. Deep neural networks are able form good intuition and learn complicated and vague rules that are hard to program and communicate, but existing libraries that cover only a small portion of mathematics don't provide enough data to train them. How to collect data without excessive human labor is the big problem. Neural machine translation from informal mathematical texts is a prospective approach to collecting a large amount of machine-readable formal data and building libraries automatically.

However, in the advent of deep reinforcement learning, I have a strong sense that it will be a more efficient approach to allow machines to explore freely by themselves, bound only by logic, gather data along the way, and learn from their own experiences. From an AI point of view, formal systems (especially proof assistants which have already collected many definitions, theories and theorems in their libraries as human supervision data to serve as starting points and anchors for curriculum learning) provide rich environments for machines to learn about reasoning and mathematics and for the marriage between connectionism and symbolism. From the point of view of mathematics, which is largely about abstraction, generality, and problem solving, it is also an important and interesting problem to identify the most essential elements that enables creative behavior in mathematical research and general problem solving.

Where libraries have not been developed, it is natural for machines to start from the axioms and definitions and take a bottom-up approach. This is maybe contrary to how humans approach mathematics, since they are inspired by real world phenomena and usually have some idea about and examples of the objects before figuring out the general definition. However, this won't prevent machines from deriving basic facts in the theory (like simulating a world from physical laws), observing naturally arising (emergent) objects, developing intuition about them, and putting down definitions. Representations of diagrammatic or geometric nature could be formed in the hidden layers of neural nets, and it might be possible to extract those.

For proof assistant operating with tactics, the system need to generate conjectures to start proof search attempts; in general, it would also be beneficial for the system to set goals for itself to guide the exploration for a while. In addition to synthesizing definitions and propositions as goals, the next steps would be allowing the system to bundle propositions as axioms to form new theories, to synthesize tactics and actions of lower granularity (e.g. programs to realize algorithms for efficiently solving certain types of problems to maximize reward of type (1) below), and thereby to exhibit more sophisticated mathematical research behavior. The synthesized items, embedded into a vector space, should be organized into certain data structure (e.g. a ball tree to facilitate nearest neighbor algorithms) that reflect connections and analogies among them.

To start reinforcement learning, we need proper designs of a neural net for the agent and a reward function to provide feedback from the environment. In my opinion, the reward could come from three sources: (1) efficiency in solving "natural" problems in the formal system (e.g. determining the truth/falsity of the shortest propositions); (2) proving a "nontrivial", or surprising, proposition, and the "surprisingness", or novelty, could be measured by the probability of being true (or equivalently, information gain from proving this proposition) as predicted by a discriminative component of the neural net (as in curiosity-driven approaches to reinforcement learning; the fluctuation in "winrate" during a proof search can also serve as a measure); (3) alignment with human-specified curriculum, e.g. proving a theorem in the library or solving a problem of predetermined form (cf. assigning heroes to lanes and punishing deviations from the lanes in OpenAI Five).

As for the agent, a single LSTM net seems sufficient for our purpose, as it encapsulates short-term memory about the exploration process; the long-term memory for synthesized items will be stored outside of the net. The actions of the agent include picking two (or more) atomic or synthesized items and combining or performing substitution to get a new item; the items can be chosen according to a policy determined the inner product of the LSTM's outputs with vector embeddings of the items (as in OpenAI Five, which operates on a continuous action space). If the synthesis produces a new theorem, the agent will be given a reward according to novelty. The new item will be given an entry in the collection of synthesized items, indexed by its vector embedding, and the way it is synthesized will be recorded in the entry. The actions also include adding or removing assumptions, changing the goal, and possibly deciding when to enter a MCTS proof search and allotting playouts. The theory, assumptions, goal, and the action chosen at the last time step are given as inputs to the LSTM (the embedding of these should be trainable); since we eventually want the LSTM to solve problems for us, we could give it an additional parameter as input which specifies the level of exploration versus focus (e.g. it could determine how long in the future the agent is disqualified from other rewards until the current goal is solved), and the value of the parameter could be randomly set whenever the goal is changed. Then, the LSTM will be trained using some optimization algorithm (e.g. PPO used for OpenAI Five) to maximize the cumulative reward (with decay). Several copies of the same net can be run in parallel to speed up production of training data.

I am planning to implement such a system and apply it to set theory, finite group theory[1], and game solving (Hex and Go)[2] as first experiments; elementary number theory, combinatorial group theory, point-set topology, synthetic (plane) geometry, etc. are some other possibilities. I have not decided on the proof assistant to use: available proof state extraction tools like in GamePad (for Coq) would be convenient, but we probably want set theory as the foundation; if we need to develop the framework from scratch, systems with a flexible

[1]The classification of finite simple groups have always been on the top of things people want to see formalized, and that will be our eventual goal; symbolic manipulation and algebra may be easier for machines to handle.

[2]Hex engines have traditionally adopted solvers as a component which perform pattern matching and decomposition of boards, but solving Go is not common practice and there has been no progress since 2009 when the empty 5×6 board was solved; no one has attempted formalization of solutions as far as I know. Since the goal is explicit, and it is straightforward to make progress (search deeper), it should be easy to achieve good result in game solving. The hope is that a stronger deduction system (than mere tree search) can compensate for speed loss from formalization (at least partly) and capture complicated dependencies among different groups of pieces, allowing machines to approach these games in a more human way; after some optimization, it is hopeful that the system can serve as the basis of a competitive engine.

and minimalistic design like Metamath with only one kind of "action" may be the easiest choice.

It will be interesting to see how far the system can go and how much of its work will be appreciated by mathematicians, and tune the hyperparameters and try different reward functions and neural net architectures (attention, NTM/DNC, etc.) accordingly; at least, I expect it to produce meaningful and rich data to train proof searchers. Ideally, a full-fledged such system will exhibit creative mathematical research behavior, hopefully reaching human level, and organize gained knowledge through vector embedding and other means. Many people believe that mathematics is universal and mathematics of aliens would be similar to ours; it should then be reasonable to expect that machine-developed mathematics will also be similar. If that is the case, it should not be hard to draw parallels and build dictionary between the two bodies of knowledge and to instruct machines to fill in nontrivial details to develop our favorite mathematical theories with just a few hints. Machines will then gradually absorb all past mathematics, quickly formalize and verify new mathematics as they come out to resolve any doubt or dispute, and provide universal access to most advanced mathematical knowledge to any level of detail you want. And, as Art Quaife wrote, "the time will come when such crushers as Riemann's hypothesis and Goldbach's conjecture will be fair game for automated reasoning programs. For those of us who arrange to stick around, endless fun awaits us in the automated development and eventual enrichment of the corpus of mathematics." Go players around the world have been experiencing such fun ever since the innovative "move 37", with the release of AlphaGo self-play games and opening book, and with the development of the open-source implementation Leela Zero. AlphaGo proved many heuristics humans have been using in gameplay and analysis unreliable, and opening in professional games is completely revolutionized. The prospects of deep reinforcement learning in mathematics in much more exciting, and I can't wait to make things happen.