Score-based Program Synthesis

Supervisors: Théo Matricon, Nathanaël Fijalkow

Location: LaBRI, Level: M2, Email: `theomatricon@gmail.com`

Programs are a representation in-between human and machines. Humans write them, understand them, whereas machines can check and execute them. Furthermore, they also provide a language with great expressivity which is not the case of competitors such as decision trees. These properties make them likely candidates for automatic systems that aim to provide a generic solution to a problem. Most recent works focus on large language models [CTJ+21, LCC+22] for code generation, while this approach seems to works quite well on some cases, most models fail to do some simple unseen tasks. Another approach, ours [FLM+22], is to describe the space of programs with a grammar, use a neural network as an oracle to obtain a probability distribution over programs in this grammar and then enumerate them in non-increasing order of probability.

But generating a program does not guarantee that, for large solutions, the program obtained is easily explainable to a human. It is possible that the program contains obscure data structures or expressions such as the famous Doom fast inverse square root code. This begs the question of introducing a human in the loop, to whom we could give a program to score, the goal would be to generate a correct program that maximises the given score. In other words, we have some evaluation function on programs which we want to maximise, this function is a black box. But we cannot ask for each program the human's input even if we would like to do so, therefore we consider the case where we evaluate the score for each program and want to maximise the score.

The candidate will first get used to the existing python framework. Then we will explore the idea of program synthesis with rewards by creating a domain specific language to represent policies in simple reinforcement learning environment easily available thanks to Gym [BCP+16]. We will try different methods possibly inspired from the bandit literature [LS20] or reinforcement learning techniques [SB05].

References

[BCP+16] Greg Brockman, Vicki Cheung, Ludwig Pettersson, Jonas Schneider, John Schulman, Jie Tang, and Wojciech Zaremba. Openai gym, 2016. `arXiv:arXiv:1606.01540`.

[CTJ+21] Mark Chen, Jerry Tworek, Heewoo Jun, Qiming Yuan, Henrique Ponde de Oliveira Pinto, Jared Kaplan, Harrison Edwards, Yuri Burda, Nicholas Joseph, Greg Brockman, Alex Ray, Raul Puri, Gretchen Krueger, Michael Petrov, Heidy Khlaaf, Girish Sastry, Pamela Mishkin, Brooke Chan, Scott Gray, Nick Ryder, Mikhail Pavlov, Alethea Power, Lukasz Kaiser, Mohammad Bavarian, Clemens Winter, Philippe Tillet, Felipe Petroski Such, Dave Cummings, Matthias Plappert, Fotios Chantzis, Elizabeth Barnes, Ariel Herbert-Voss, William Hebgen Guss, Alex Nichol, Alex Paino, Nikolas Tezak, Jie Tang, Igor Babuschkin, Suchir Balaji, Shantanu Jain, William Saunders, Christopher Hesse, Andrew N. Carr, Jan Leike, Joshua Achiam, Vedant Misra, Evan Morikawa, Alec Radford, Matthew Knight, Miles Brundage, Mira Murati, Katie Mayer, Peter Welinder, Bob McGrew, Dario Amodei, Sam McCandlish, Ilya Sutskever, and Wojciech Zaremba. Evaluating large language models trained on code. *CoRR*, abs/2107.03374, 2021. URL: `https://arxiv.org/abs/2107.03374`, `arXiv:2107.03374`.

[FLM+22] Nathanaël Fijalkow, Guillaume Lagarde, Théo Matricon, Kevin Ellis, Pierre Ohlmann, and Akarsh Nayan Potta. Scaling neural program synthesis with distribution-based search. *Proceedings of the AAAI Conference on Artificial Intelligence*, 36(6):6623–6630, Jun. 2022. URL: `https://ojs.aaai.org/index.php/AAAI/article/view/20616`, `doi:10.1609/aaai.v36i6.20616`.

[LCC+22] Yujia Li, David Choi, Junyoung Chung, Nate Kushman, Julian Schrittwieser, Rémi Leblond, Tom Eccles, James Keeling, Felix Gimeno, Agustin Dal Lago, Thomas Hubert, Peter Choy, Cyprien de Masson d'Autume, Igor Babuschkin, Xinyun Chen, Po-Sen Huang, Johannes Welbl, Sven Gowal, Alexey Cherepanov, James Molloy, Daniel J. Mankowitz, Esme Sutherland Robson, Pushmeet Kohli, Nando de Freitas, Koray Kavukcuoglu, and Oriol Vinyals. Competition-level code generation with alphacode, 2022. URL: `https://arxiv.org/abs/2203.07814`, `doi:10.48550/ARXIV.2203.07814`.

[LS20] Tor Lattimore and Csaba Szepesvári. Bandit algorithms. 2020.

[SB05] Richard S. Sutton and Andrew G. Barto. Reinforcement learning: An introduction. *IEEE Transactions on Neural Networks*, 16:285–286, 2005.