Neural Program Synthesis with Soft Actor Critic

Lucas Kabela
Department of Computer Science, The University of Texas

Abstract

In a programming by example (PBE) neural program synthesis framework, supervised learning has shown to be an effective approach in Delvin et al. 2017. However, supervised learning optimizes for exact program matches to a reference program, ignoring programs which are consistent with the input/output examples provided.

Thus, we investigate the effects of directly optimizing for consistency through reinforcement learning in the domain of string transforming programs. We propose using Soft Actor Critic (SAC) due to its low sample complexity and maximization of entropy, which explores a larger space of viable candidate solutions, an important feature in PBE. We observe reinforcement learning techniques improve consistency metrics by up to 10.4% above a supervised baseline at no significant drop in exact match accuracy. Further, SAC yields 4.36% higher consistency than REINFORCE, and produces a greater number of consistent candidates.

Introduction

In neural program synthesis, a neural network is trained on input/output pairs to predict program tokens. This task can be viewed as a form of supervised sequential generation which Delvin et al. 2017 showed can achieve up to 56% accuracy with a basic seq-to-seq model.

However, in a PBE framework, we are only given a small handful of examples which underspecify the task. As such, a training objective which maximizes the likelihood of generating programs which are consistent is preferred to one which maximizes exact matches. Such an objective would theoretically output a larger space of potentially satisfying programs, which further user supplied input/output examples can refine.

One objective which enables us to encourage consistency is reinforcement learning (RL), which uses a reward function to train models. With the further goal of outputting a space of satisfying programs, we look to Soft Actor Critic (SAC) by Haarnoja et al. 2018, which adds an extra term of maximizing entropy to the loss function, effectively encouraging exploration over many candidate solutions. Therefore, in this work we implement RL algorithms to investigate if reinforcement learning improves consistency in neural program synthesis.

Methods

We began by replicating results from Delvin et al. 2017, constructing the seq-2-seq Attention A architecture. We further incorporated teacher learning for training. This model was trained on 2.56 million programs on a P100. This is network serves as a pretrained policy for further supervision and RL training (see Discussion). We trained this network another 1.28 million programs with supervised learning to establish the supervised baseline.

Next, we implemented a simple MDP for the reinforcement learning algorithms, where states are tuples of the user program so far, and the i/o examples, actions are the next token in the program, and the reward is 0 or 1 if the previous token was an EOS and the program is consistent with the examples. We use a curriculum of growing program sizes for stability.

We then implement REINFORCE with a value network baseline to establish an RL baseline, and next SAC, following the architecture in Figure 1. We use a replay buffer, and optionally HER to improve learning by increasing density of reward signals. Models were trained for another 1.28 million programs on a P100 while holding policy constant for the first 100,000 programs to adjust value networks. We used early stopping to achieve maximal results.

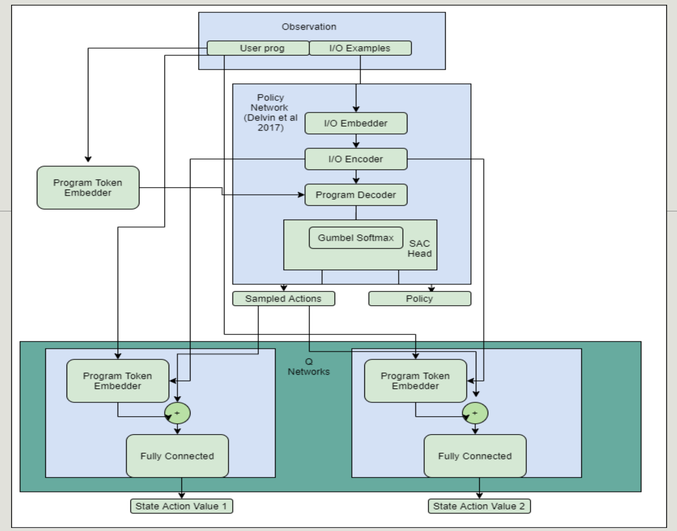

Figure 1. Soft Actor Critic Network Architecture

Results

We ran two sets of experiments over 5 batches of 1000 programs. These evaluate the exact match accuracy (EM), and the percent of beams which contain at least 1 program consistent with the i/o examples, recorded in Tables 1 and 2. All RL techniques improved upon the supervised baseline, with up to 10.4% higher consistency at no significant difference in EM.

Finally, to test our hypothesis of the effects of entropy on the number of candidate solutions, we evaluated the number of consistent programs in beams of size 10 and record the results in Figure 2. We observed SAC contained significantly more beams with 4+ consistent programs.

	1	10	100
Supervised	18.12	**26.94**	**34.30**
REINFORCE	18.94	25.82	32.20
SAC	**19.58**	26.02	31.60
SAC + HER	18.40	25.34	29.8

Table 1. Exact Match Accuracy (%) vs Beam Size

	1	10	100
Supervised	29.36	42.60	54.00
REINFORCE	33.32	51.12	63.8
SAC	37.68	**51.92**	**64.4**
SAC + HER	**37.76**	51.46	61.12

Table 2. Consistency (%) vs Beam Size

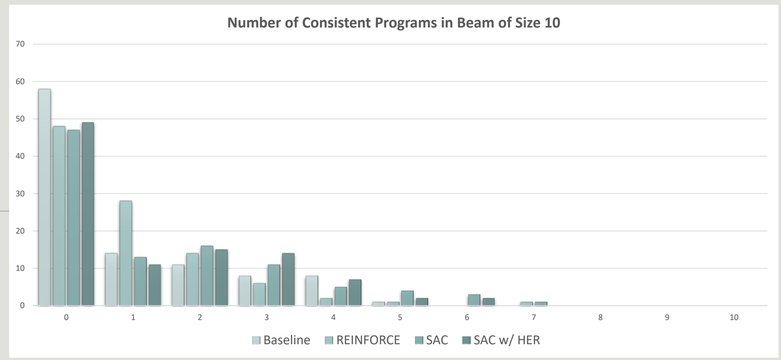

Figure 2. Number of Consistent examples in Beam Size 10.

Discussion

The consistency improvement of RL methods is inline with our initial hypothesis and indicates RL can be used to fine tune neural program synthesis. Furthermore, SAC and SAC + HER had significantly ($p = .000106$) higher consistency for beam size 1 than REINFORCE, which we believe to be due to the sample efficiency of off policy learning.

Furthermore, consistent with our hypothesis on entropy encouraging exploration, we observed that SAC and SAC + Her had a greater number of beams with greater than 4 consistent programs. However, we note that this could be a property of off policy learning in general and believe more work should be done with off policy deep learning algorithms to investigate the true effectiveness of the entropy term.

One disappointing note was that HER did not lead to any significant improvement over SAC, and further we were not able to train an RL algorithm from scratch (hence why we used a pretrained policy network). We attribute this to the sparsity of the reward and high dimensionality of the action space, and reliance on the i/o examples, which did not change with HER. Modifications such as hallucinations by Sahni et al 2019 or Prioritized Experience Replay by Brittian et al. 2019 may lead to better results for training from scratch, but for now our results are consistent with Zohar and Wolf 2018, which experienced RL approaches as intractable in this domain. Finally, we could not reach the accuracy of prior work due to resource constraints (we trained only $1/10^{th}$ of the number of programs as in prior work).

Conclusions

In conclusion, we demonstrate that reinforcement learning algorithms can successfully be applied to neural program synthesis as a tool for fine tuning program consistency. We observed an improvement over a supervised baseline by up to 10.4% with a beam decoder of size 100. Furthermore, we observe the entropy term of soft actor critic indeed improves the number of consistent programs found over both the supervised baseline and REINFORCE on policy gradient technique.

While we were not able to train a model from scratch despite sophisticated techniques such as HER, we believe there are many viable modifications in the reinforcement learning literature which can further improve upon our results and encourage more reinforcement learning work in neural program synthesis.

Contact

Lucas Kabela
Department of Computer Science, University of Texas
Email: lucaskabela@utexas.edu
Code available online at https://github.com/Lucaskabela/prog-synth-SAC

References

1. Marcin Andrychowicz, Filip Wolski, Alex Ray, Jonas Schneider, Rachel Fong, Peter Welinder, BobMcGrew, Josh Tobin, Pieter Abbeel, and Wojciech Zaremba. Hindsight experience replay.CoRR,abs/1707.01495, 2017. URL http://arxiv.org/abs/1707.01495.
2. Marc Brittain, Joshua R. Bertram, Xuxi Yang, and Peng Wei. Prioritized sequence experience replay.CoRR, abs/1905.12726, 2019. URL https://arxiv.org/abs/1905.12726
3. Jacob Devlin, Jonathan Uesato, Surya Bhupatiraju, Rishabh Singh, Abdel rahman Mohamed, andPushmeet Kohli. Robustfill: Neural program learning under noisy i/o, 2017.
4. Sumit Gulwani. Automating string processing in spreadsheets using input-output ex-amples.InPoPL'11, January 26-28, 2011, Austin, Texas, USA, January 2011.URL https://www.microsoft.com/en-us/research/publication/automating-string-processing-spreadsheets-using-input-output-examples/.
5. Tuomas Haarnoja, Aurick Zhou, Pieter Abbeel, and Sergey Levine. Soft actor-critic: Off-policymaximum entropy deep reinforcement learning with a stochastic actor, 2018
6. Himanshu Sahni, Toby Buckley, Pieter Abbeel, and Ilya Kuzovkin. Visual hindsight experiencereplay.CoRR, abs/1901.11529, 2019. URL https://arxiv.org/abs/1901.11529
7. Amit Zohar and Lior Wolf. Automatic program synthesis of long programs with a learned garbagecollector.CoRR, abs/1809.04682, 2018. URL http://arxiv.org/abs/1809.04682.