CSCI 5922: Deep Learning Project Proposal

"XAI in Deep Sequence Models by State Machine Extraction"

Nicholas W. Renninger*

I. Problem Overview

As an aerospace engineer, my interest in new, amazing machine learning technology like deep learning is somewhat tempered by the lack of explainability of Deep Neural Networks (DNNs), especially as pertaining to the solution of sequential decision-making problems. This means that while we have developed extensive methodologies in machine learning, motion planning, and formal methods for approaching humanity's desire for autonomous systems, but we currently lack robust ways to provide performance guarantees for autonomous, safety-critical systems. We are in dire need of more principled and guaranteed design methodologies than many "black-box" methods in use today if we want to realize trusted autonomy for safety-critical systems in our society.

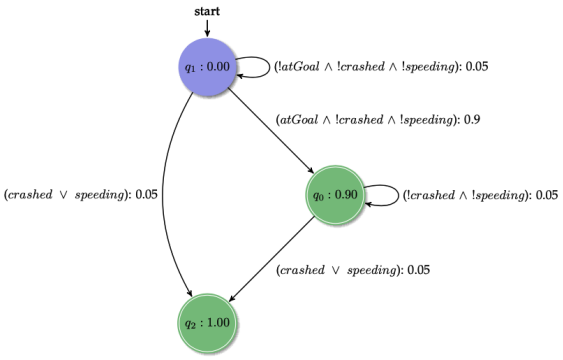

Fig. 1 An example of a probabilistic deterministic finite automaton (PDFA) specification for one of my toy autonomous driving scenarios.

To bring us closer to provably safe, interpretable decision-making algorithms for autonomous systems, one approach I am investigating is to use machine learning to learn high-level human goals from demonstrations in a format that is amenable to formal control synthesis. Autonomous vehicles exemplify an industry that currently faces many challenges in the field of decision making and control for a safety-critical system. In his annual review in 2018, Schwarting outlined the progress made in the Verification and Synthesis of decision-making and motion planning algorithms [1]. Black box methods (i.e. DNN representing policy / value networks in Deep Reinforcement Learning) are struggling to be verified [1], which makes their use questionable in such a safety-critical environment. It was particularly noted that while formal control synthesis was of great interest, it does not see wide-spread use cause due to its expensive and the due to the difficulty in formally specifying desired system behavior.

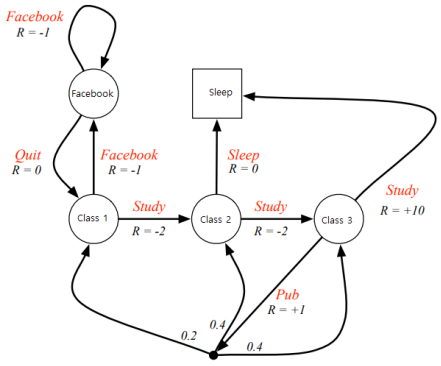

Fig. 2 An example of a Markov Decision Process (MDP) for which we might want to learn a specification. (source)

My research consists of first using probabilistic state-machine learning algorithms (e.g. ALERGIA, MDI [2], or RTI(+) [3]) to learn an automaton model for human demonstrator's specification for the operation of an autonomous system based on observed symbol sequences (traces) representing of system behavior. See Fig. 1 for an example of such a specification. These probabilistic state machines allow are generative in nature, and when sampled from produce traces in the language of the probabilistic automaton. Then, using tools from formal methods, we compose this specification with a model for the autonomous system's dynamics (e.g. transition system or (PO)MDP – see Fig. 2) to obtain a correct-by-construction policy to control the system model to follow the learned specification.

For this project, I would like to focus on the **specification learning** aspect of the project. While the methods mentioned earlier (ALERGIA, MDI, RTI(+)) work with a classical state-merging to merge the observed traces from the system's operation into a statistically consistent probabilistic state machine, I would like to investigate new works that leverage the amazing power of neural networks as probabilistic sequence models. This means that instead of directly learning a state machine

*Graduate Student, Ann & H. J. Smead Department of Aerospace Engineering, 3775 Discovery Drive Boulder, CO 80303

from the observed traces, **I would like to experiment with using a learned DNN (probably a recurrent or attention model) to represent the language distribution over the given symbol alphabet, and then extract the language distribution from the DNN as a state machine.**

This sort of process is of interest to researchers working on explainable AI (XAI) as well as researchers like me working on formal control synthesis for uncertain sequential decision-making systems.

II. Possible Approaches

I have two approaches that I might explore for the final project:

- Purely studying RNN language models and the ability of existing methods to extract a state machine from an RNN trained for my problem.
- Studying how RNN language models used as policy representations in a deep reinforcement learning setting fair as targets for state machine extraction.

In the case of pure RNN language model study, there are a large variety of methods that have had varying levels of success in extracting state machines from RNNs trained to be language models [4] [5] [6] [7] [8] [9] [10]. This process would consist of using the generative target models described in Sec. III to build the training corpus, and then experimenting with some of the more advanced RNN / transformer language models presented. Then, I would choose one of the more recent, salient methods to extract an appropriate PDFA from the citations listed (right now I'm leaning towards [8]), and compare the extracted state machine language model's performance to that of the learned DNN language model.

In the case in which we might learn and extract a state machine from a recurrent DNN (RNN) used as a policy representation using deep reinforcement learning, things would be pretty different. The idea here would be based off of current work to extract state machines from an RNN representing the policy in the case of either model-based POMDP policy iteration [11] or model-free reinforcement learning [12].

In this case, I would primarily be interested in implementing the work done by Koul [12], as this was the original work on extracting a moore-machine from an RNN encoding an RL agent's policy. This approach is really interesting:

1) Use a baseline RL algorithm (e.g. DDPG, A3C, SAC, TRPO) to train a RNN-LSTM policy for an atari game given as an RL environment. This would involve designing a good LSTM architecture that trains well for the given environment.
2) Next, we build the state machine *into* the policy network by quantizing both the feature space of the LSTM input and the hidden state space of the LSTM into a small, finite number of discrete states. This is accomplished by training two encoder-decoder pairs – one for the feature quantization and one for the hidden state quantization. These encoder / decoder pairs are trained by feeding features and traces from the replay buffer through these compression networks and minimizing the quantization / compression loss.
3) Then, these encoder / decoder pairs are inserted back into the original policy network before and after the recurrent layers. As the quantization networks have already been trained, only fine tuning of this new policy network should be necessary with more RL episodic training.
4) You can now extract a moore machine from this combined network by simply sampling properties from the encoder stage of each quantization network. From these quantized values, you can reconstruct a moore machine policy that has a small number of discrete observations and states, and that has encoder / decoder networks that allow this fully discrete controller to interface with a continuous observation / control RL environment (e.g. observing frames of an atari game and sending controls).

This is quite a bit of implementation, but the paper is well written and it should be fairly straightforward to follow along. One thing that could decrease the difficulty of the project would be to use a simpler RL environment that does not require the training of a CNN for video feature extraction.

III. Feasibility Analysis

As we are most interested in comparing the structure, parameters, and language distribution of the extracted state machine model with that of a more traditional state machine learning algorithm, the training / testing data is most usefully generated at will from a target state machine (this is very common in any study done on a state machine learning algorithm). Besides generating training / test data from desired

language model state machine targets, I can also utilize widely available language modeling competition datasets (i.e. PAutomaC [3]).

This project will focus on the ability to extract a state machine from a DNN language model which means there is actually a quite straightforward way to evaluate the feasibility of this sort of technique in general. I can not only evaluate standard language model measures like perplexity score [3], but also information theoretic measures of model performance like coding rate, model distortion (e.g. Kullback-Leibler (KL) Divergence or basic predictive accuracy), and mutual information [13]. These comparisons will allow the information theoretic properties of traditionally inferred probabilistic state machines to be readily compared to that those of learned DNNs and those of state machines extracted from these same learned DNNs.

In the case that the DNN is learned through a deep (possibly inverse) RL algorithm, there will obviously be ample data to train the DNN policy / value model and well studied reward evaluation models from the field of uncertain decision-making; thus the problem is still feasible from this RL angle.

As outlined in Section II, the methods I will be trying out are not incredibly computationally intensive, as far as deep learning goes.

Thus, there is ample evidence to suggest that there is enough data to learn and evaluate the proposed models, so this project does not have any immediately obvious feasibility problems.

IV. Project Logistics

For this project I will be working along, and I am using a working title of: "`XAI in Deep Sequence Models by State Machine Extraction`".

References

[1] Schwarting, W., Alonso-Mora, J., and Rus, D., "Planning and Decision-Making for Autonomous Vehicles," *Annual Review of Control, Robotics, and Autonomous Systems*, Vol. 1, No. 1, 2018, pp. 187–210. doi:10.1146/annurev-control-060117-105157, URL https://doi.org/10.1146/annurev-control-060117-105157.

[2] de la Higuera, C., *State Merging Algorithms*, 2004, Cambridge University Press, New York, NY, USA, 2013, Chap. 16.3, pp. 333–339. doi:10.1017/CBO9781139194655.

[3] Verwer, S., Eyraud, R., and Higuera, C., "PAutomaC: A Probabilistic Automata and Hidden Markov Models Learning Competition," *Mach. Learn.*, Vol. 96, No. 1-2, 2014, pp. 129–154. doi:10.1007/s10994-013-5409-9, URL https://doi.org/10.1007/s10994-013-5409-9.

[4] Weiss, G., Goldberg, Y., and Yahav, E., "Extracting Automata from Recurrent Neural Networks Using Queries and Counterexamples," *Proceedings of the 35th International Conference on Machine Learning*, Proceedings of Machine Learning Research, Vol. 80, edited by J. Dy and A. Krause, PMLR, Stockholmsmässan, Stockholm Sweden, 2018, pp. 5247–5256. URL http://proceedings.mlr.press/v80/weiss18a.html.

[5] Giles, C. L., Lawrence, S., and Tsoi, A. C., "Noisy Time Series Prediction using Recurrent Neural Networks and Grammatical Inference," *Machine Learning*, Vol. 44, No. 1/2, 2001, pp. 161–183. doi:10.1023/A:1010884214864, URL http://link.springer.com/10.1023/A:1010884214864.

[6] Weiss, G., Goldberg, Y., and Yahav, E., "Extracting Automata from Recurrent Neural Networks Using Queries and Counterexamples," Tech. rep., 2018.

[7] Michalenko, J. J., Shah, A., Verma, A., Baraniuk, R. G., Chaudhuri, S., and Patel, A. B., "Representing Formal Languages: A Comparison Between Finite Automata and Recurrent Neural Networks," *ArXiv*, Vol. abs/1902.1, 2019.

[8] Le, T.-D. B., and Lo, D., "Deep specification mining," *ISSTA*, 2018.

[9] Dong, G., Wang, J., Sun, J., Zhang, Y., Wang, X., Dai, T., and Dong, J. S., "Analyzing Recurrent Neural Network by Probabilistic Abstraction," *ACM Reference format*, 2016. doi:10.1145.

[10] Ayache, S., Eyraud, R., and Goudian, N., "Explaining Black Boxes on Sequential Data using Weighted Automata," 2018.

[11] Carr, S., Jansen, N., Wimmer, R., Serban, A., Becker, B., and Topcu, U., "Counterexample-guided strategy improvement for POMDPs using recurrent neural networks," *IJCAI International Joint Conference on Artificial Intelligence*, Vol. 2019-Augus, 2019, pp. 5532–5539. doi:10.24963/ijcai.2019/768.

[12] Koul, A., Fern, A., and Greydanus, S., "Learning finite state representations of recurrent policy networks," *7th International Conference on Learning Representations, ICLR 2019*, 2019, pp. 1–15.

[13] Marzen, S. E., and Crutchfield, J. P., "Probabilistic Deterministic Finite Automata and Recurrent Networks, Revisited," *ArXiv*, 2019.