Automata Learning Meets State Space Machines

William Fishell[1] Mark Santolucito[2]

[1]Columbia University
[2]Columbia University, Barnard College

October 7th

State Space Models and Mealy Machines

- State Space Models have become popular in recent years as a promising alternative to Transformers [1, 2]

State Space Models and Mealy Machines

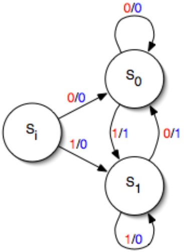
- State Space Models have become popular in recent years as a promising alternative to Transformers [1, 2]
- Unlike Mealy Machines, these models parameterize a continuous state space

State Space Models and Mealy Machines

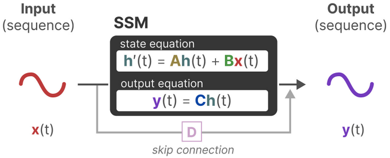
- State Space Models have become popular in recent years as a promising alternative to Transformers [1, 2]
- Unlike Mealy Machines, these models parameterize a continuous state space
- Despite this, their update dynamics are both input and state dependent

State Space Models and Mealy Machines

- State Space Models have become popular in recent years as a promising alternative to Transformers [1, 2]
- Unlike Mealy Machines, these models parameterize a continuous state space
- Despite this, their update dynamics are both input and state dependent



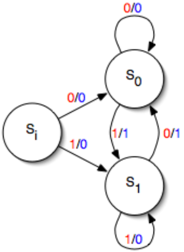
Mealy Representation



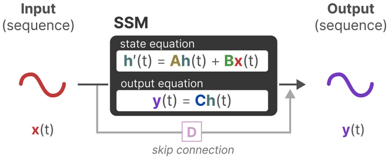
State Space Update Dynamics

State Space Models and Mealy Machines

- State Space Models have become popular in recent years as a promising alternative to Transformers [1, 2]
- Unlike Mealy Machines, these models parameterize a continuous state space
- Despite this, their update dynamics are both input and state dependent



Mealy Representation



State Space Update Dynamics

- State Space models and Mealy machines are "learned" very differently, yet much of their computational behavior is very similar

How well do these two machines "learn" various reactive systems

- Mealy machines are usually synthesized, but in a data-rich, spec-poor world, we construct them through active and passive automata learning [3]

How well do these two machines "learn" various reactive systems

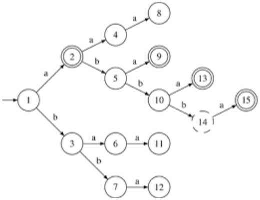
- Mealy machines are usually synthesized, but in a data-rich, spec-poor world, we construct them through active and passive automata learning [3]
- Unlike Mealy machines, SSMs learn using gradient descent

How well do these two machines "learn" various reactive systems

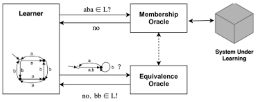
- Mealy machines are usually synthesized, but in a data-rich, spec-poor world, we construct them through active and passive automata learning [3]
- Unlike Mealy machines, SSMs learn using gradient descent
- SSMs struggle at learning regular language problems that automata learning for Mealy machines performs well in [4, 5]

How well do these two machines "learn" various reactive systems

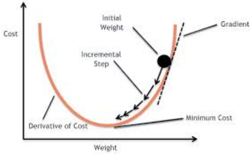
- Mealy machines are usually synthesized, but in a data-rich, spec-poor world, we construct them through active and passive automata learning [3]
- Unlike Mealy machines, SSMs learn using gradient descent
- SSMs struggle at learning regular language problems that automata learning for Mealy machines performs well in [4, 5]
- We use SyntComp benchmarks to compare automata learning and gradient descent methods for reactive systems, identifying which problem classes each learns best



RPNI Passive Learning



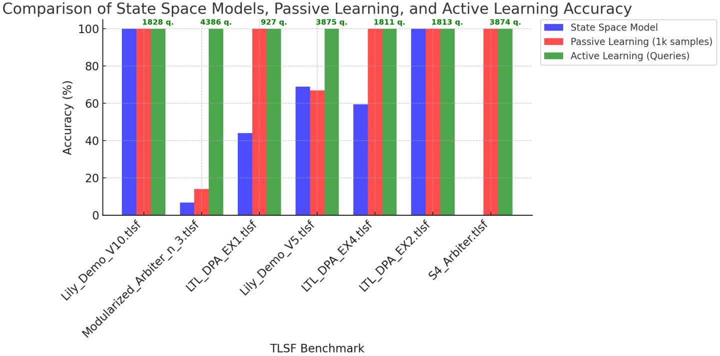
Active Learning



Gradient Descent

Analyzing sample complexity reveals how effectively each algorithm learns reactive systems

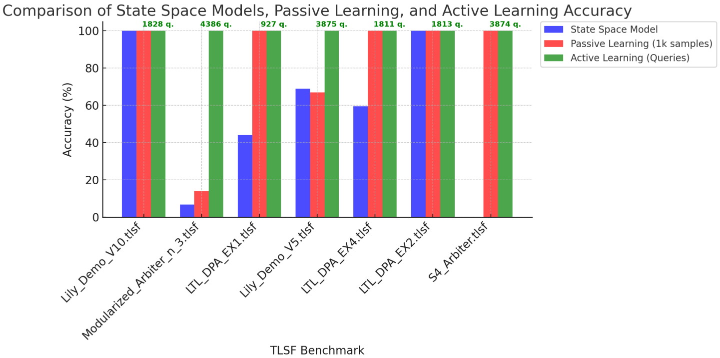
- Passive and Active learning are much more sample efficient compared to gradient based learning for SSMs



Passive Learning 1,000 samples, SSM Learning 10,000 samples

Analyzing sample complexity reveals how effectively each algorithm learns reactive systems

- Passive and Active learning are much more sample efficient compared to gradient based learning for SSMs

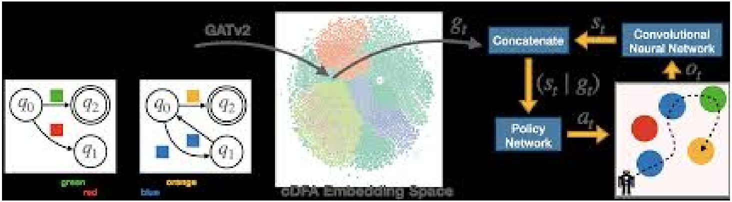


Passive Learning 1,000 samples, SSM Learning 10,000 samples

- Why should we care about this?

Projecting automata into Euclidean space enables SSMs to learn more complex reactive behaviors

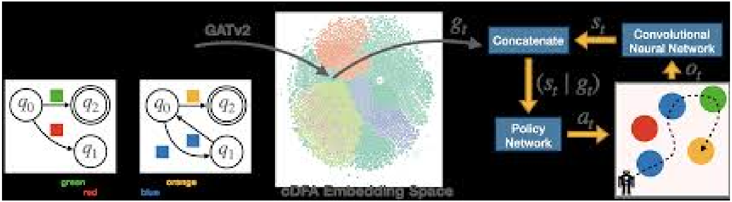
- Prior work has embedded DFAs into continuous spaces [6, 7]. We aim to leverage this to transfer safety guarantees and initialize SSMs for faster convergence



Graphic Showing Embedding of Automata for Deep Learning [7]

Projecting automata into Euclidean space enables SSMs to learn more complex reactive behaviors

- Prior work has embedded DFAs into continuous spaces [6, 7]. We aim to leverage this to transfer safety guarantees and initialize SSMs for faster convergence



Graphic Showing Embedding of Automata for Deep Learning [7]

- Recent SSM advances stem from effective initialization (e.g., HiPPO [8]). In this vein, we aim to leverage automata sample efficiency to better warm start SSMs

A. Gu and T. Dao, "Mamba: Linear-time sequence modeling with selective state spaces," *arXiv preprint arXiv:2312.00752*, 2023.

A. Gu, K. Goel, and C. Ré, "Efficiently modeling long sequences with structured state spaces," *arXiv preprint arXiv:2111.00396*, 2021.

E. Muškardin, B. K. Aichernig, I. Pill, A. Pferscher, and M. Tappler, "Aalpy: an active automata learning library," *Innovations in Systems and Software Engineering*, vol. 18, no. 3, pp. 417–426, 2022.

M. Hahn, "Theoretical limitations of self-attention in neural sequence models," *Transactions of the Association for Computational Linguistics*, vol. 8, pp. 156–171, 2020.

N. Nishikawa and T. Suzuki, "State space models are provably comparable to transformers in dynamic token selection," *arXiv preprint arXiv:2405.19036*, 2024.

C. W. Omlin and C. L. Giles, "Constructing deterministic finite-state automata in recurrent neural networks," *Journal of the ACM (JACM)*, vol. 43, no. 6, pp. 937–972, 1996.

B. Yalcinkaya, N. Lauffer, M. Vazquez-Chanlatte, and S. Seshia, "Compositional automata embeddings for goal-conditioned reinforcement learning," *Advances in Neural Information Processing Systems*, vol. 37, pp. 72 933–72 963, 2024.

A. Gu, I. Johnson, A. Timalsina, A. Rudra, and C. Ré, "How to train your hippo: State space models with generalized orthogonal basis projections," *arXiv preprint arXiv:2206.12037*, 2022.