Understanding the Training and Inference of Reinforcement Learning

Shangtong Zhang
University of Virginia

What is RL?

What is RL?

- RL is PPO!

What is RL?

$$A_t \sim \pi(\cdot \mid S_t)$$

$$R_{t+1}, S_{t+1}$$

$$v_\pi(s) = E[R_{t+1} + \gamma R_{t+2} + \gamma^2 R_{t+3} + \ldots \mid S_t = s]$$

RL = SA + DP

- RL is to use Stochastic Approximation (SA) method to solve Dynamic Programming (DP) problem

- Bellman operator $Tv = r_\pi + \gamma P_\pi v$

RL = SA + DP

- $v_{k+1} = Tv_k = r_\pi + \gamma P_\pi v_k$
- Challenge: unknown P_π
- Solution: use a sample
  $$(P_\pi v_k)(s) = \sum_{s'} p(s' | s) v_k(s')$$
  $$(P_\pi v_k)(s) \approx v_k(s')$$

RL = SA + DP

- $v_{k+1} = Tv_k = r_\pi + \gamma P_\pi v_k$
- Challenge: full update is too aggressive
  $v_{k+1}(s) = r_\pi(s) + \gamma v_k(s')$
- Solution: incremental update
  $v_{k+1}(s) = v_k(s) + \alpha_k(r_\pi(s) + \gamma v_k(s') - v_k(s))$

RL = SA + DP

- $v_{k+1} = Tv_k = r_\pi + \gamma P_\pi v_k$

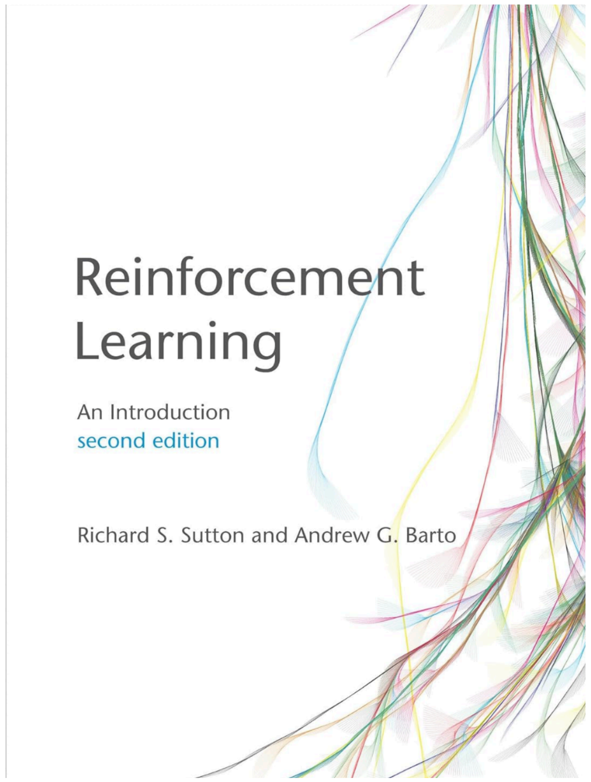
- Challenge: where to get s'?
  $\ldots, S_k, S_{k+1}, \ldots,$

- Solution: asynchronous update
  $v_{k+1}(S_k) = v_k(S_k) + \alpha_k(r_\pi(S_k) + \gamma v_k(S_{k+1}) - v_k(S_k))$

RL = SA + DP

- $v_{k+1} = v_k + \alpha_k h(v_k, Y_{k+1})$

- Different h realizes different RL algorithms, e.g., TD, Q-learning, linear TD, Gradient TD, Emphatic TD, average reward TD, differential TD, differential Q-learning

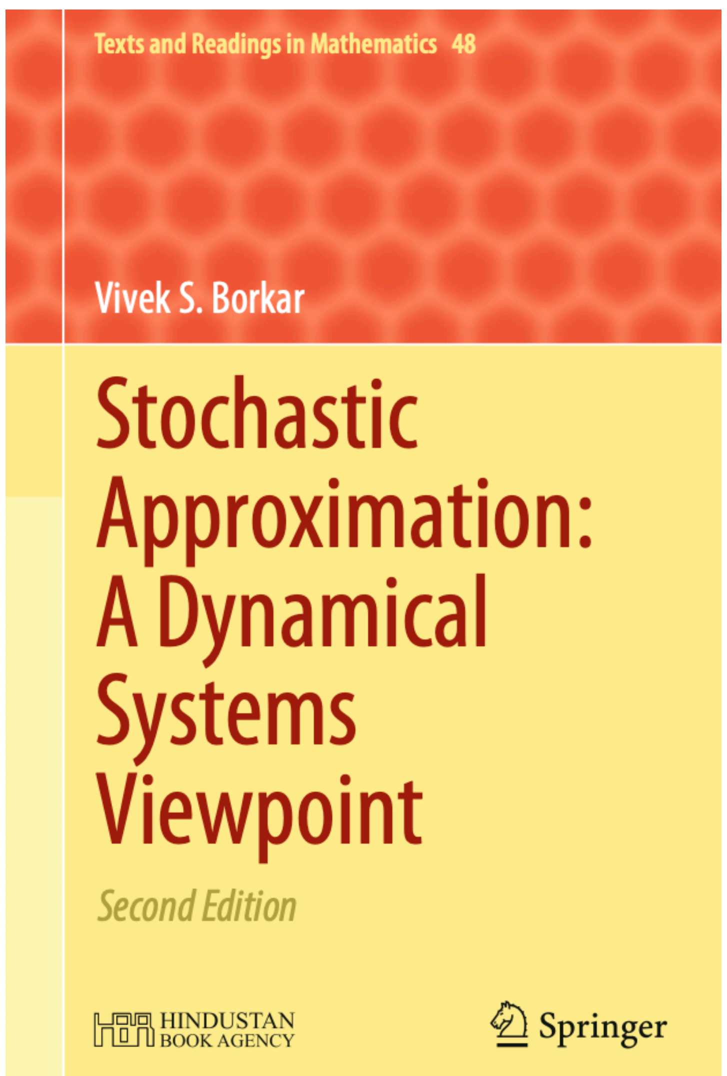


Does SA converge?

- $v_{k+1} = v_k + \alpha_k h(v_k, Y_{k+1})$
- $v_k \to v_*$ almost surely?
- Early RL pioneers borrow results from SA community
- Now RL theorists shift to fancier problems, e.g., offline RL, RLHF, etc.

Tools in SA community do not apply to RL well

- $v_{k+1} = v_k + \alpha_k h(v_k, Y_{k+1})$

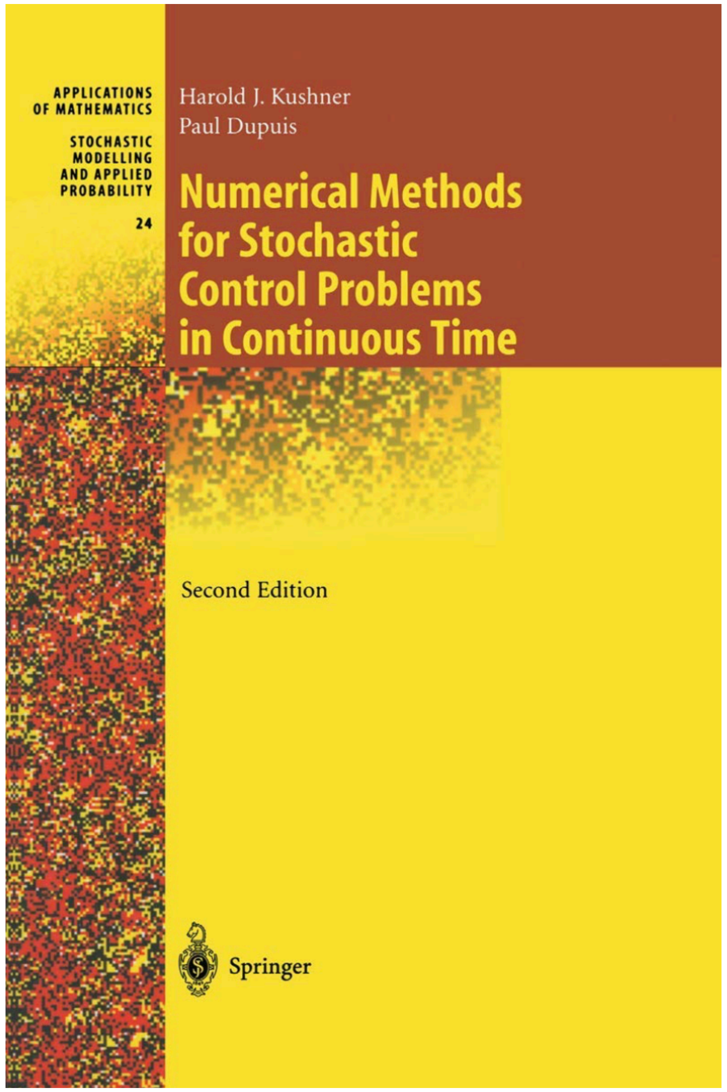


Assumption: $\{Y_k\}$ **are i.i.d.**

Reality in RL: $\{Y_k\}$ **are Markov chain**

Tools in SA community do not apply to RL well

- $v_{k+1} = v_k + \alpha_k h(v_k, Y_{k+1})$

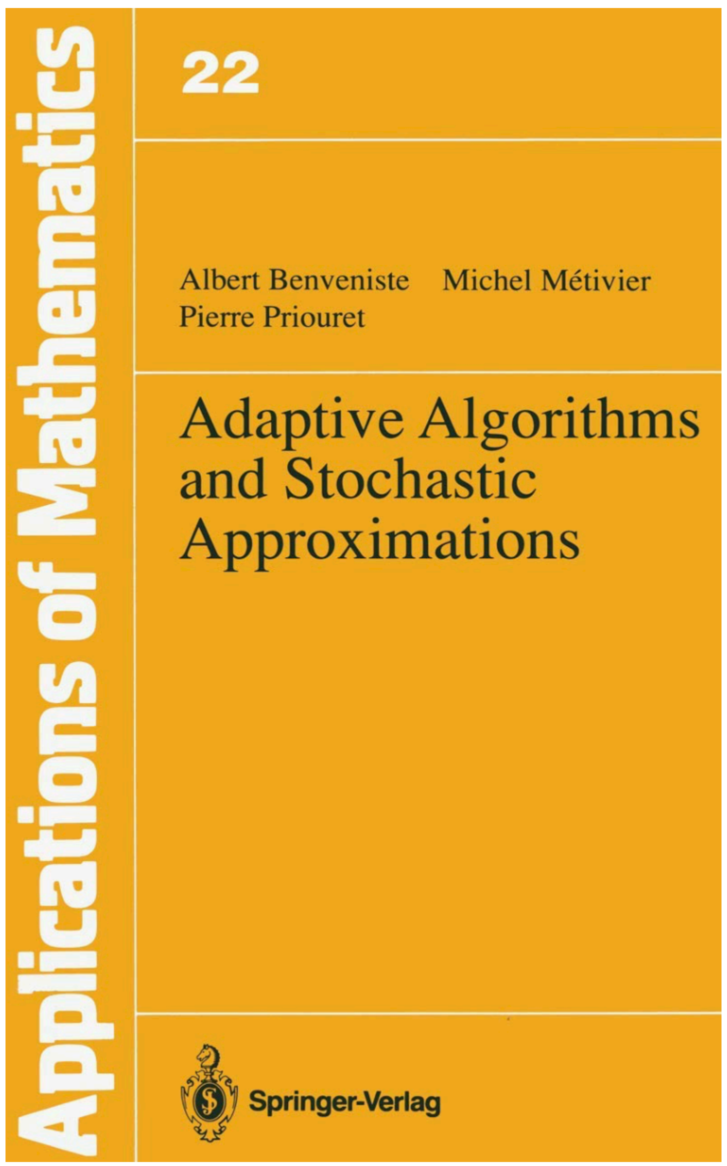


Assumption: $\sup_k ||v_k|| < \infty$ **a.s.**

Reality in RL: very hard to verify

Tools in SA community do not apply to RL well

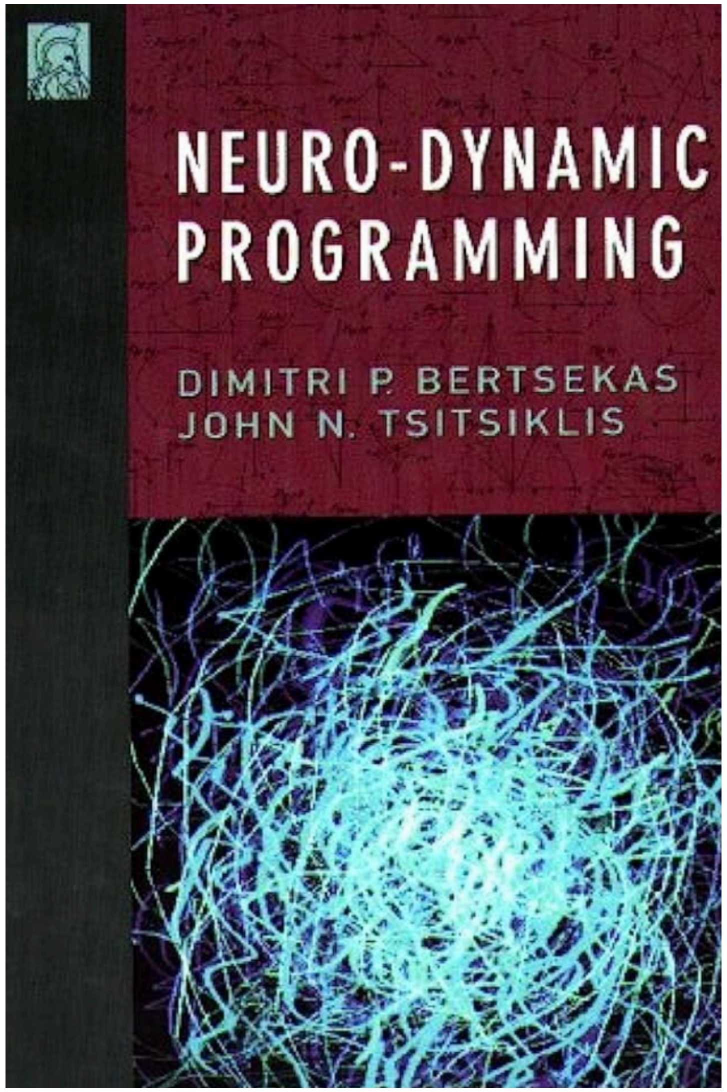
- $v_{k+1} = v_k + \alpha_k h(v_k, Y_{k+1})$



Assumption: Poisson's equation
Lyapunov function
Reality in RL: hard to prove existence

Tools in SA community do not apply to RL well

- $v_{k+1} = v_k + \alpha_k h(v_k, Y_{k+1})$



Assumption: linear h
with a negative definite matrix
Reality in RL: not true in many RL algorithms

Why can't the RL community have its own SA theory?

- $v_{k+1} = v_k + \alpha_k h(v_k, Y_{k+1})$

Cornell University

Computer Science > Machine Learning

arXiv:2401.07844 (cs)

[Submitted on 15 Jan 2024 (v1), last revised 29 Apr 2024 (this version, v3)]

The ODE Method for Stochastic Approximation and Reinforcement Learning with Markovian Noise

Shuze Liu, Shuhang Chen, Shangtong Zhang

View PDF

HTML (experimental)

Stochastic approximation is a class of algorithms that update a vector iteratively, incrementally, and stochastically, including, e.g., stochastic gradient descent and temporal difference learning. One fundamental challenge in analyzing a stochastic approximation algorithm is to establish its stability, i.e., to show that the stochastic vector iterates are bounded almost surely. In this paper, we extend the celebrated Borkar-Meyn theorem for stability from the Martingale difference noise setting to the Markovian noise setting, which greatly improves its applicability in reinforcement learning, especially in those off-policy reinforcement learning algorithms with linear function approximation and eligibility traces. Central to our analysis is the diminishing asymptotic rate of change of a few functions, which is implied by both a form of strong law of large numbers and a commonly used V4 Lyapunov drift condition and trivially holds if the Markov chain is finite and irreducible.

Subjects: **Machine Learning (cs.LG)**; Artificial Intelligence (cs.AI)
Cite as: arXiv:2401.07844 **[cs.LG]**
(or arXiv:2401.07844v3 **[cs.LG]** for this version)
https://doi.org/10.48550/arXiv.2401.07844

Assumption: Law of large numbers (LLN)
Reality in RL: if LLN does not hold,
good luck with your RL alg.

LLN on Markov chains

- $$\frac{X_1 + X_2 + \ldots X_n}{n} \to E[X_1]$$

- $$\frac{1}{n}\sum_{i=1}^{n} h(Y_i) \to \sum_{y} d(y)h(y)$$
  time average → space average (ergodicity)

- automatically hold in finite chains
  especially powerful in off-policy RL algorithms with eligibility traces

Training is all before deep RL

- Tabular TD
  $v_{k+1}(S_k) = v_k(S_k) + \alpha_k(r_\pi(S_k) + \gamma v_k(S_{k+1}) - v_k(S_k))$
  $s \rightarrow v_*(s)$

- Linear TD
  $w_{k+1} = w_k + \alpha_k(r_\pi(S_k) + \gamma x(S_{k+1})^\top w_k - x(S_k)^\top w_k) x(S_k)$
  $s \rightarrow x(s)^\top w$

Inference matters after deep RL

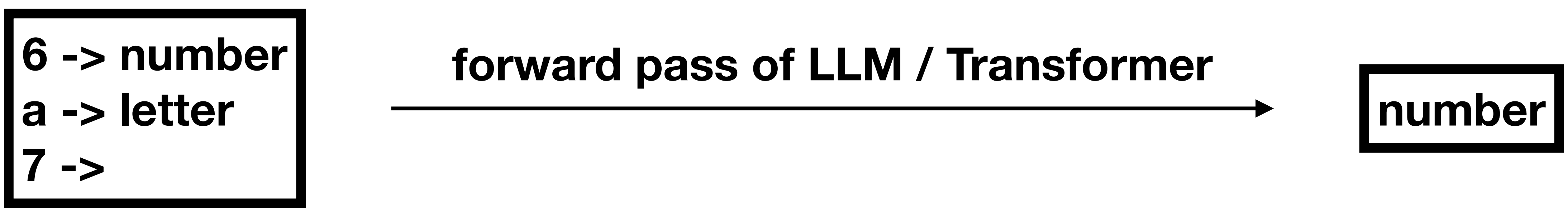
- Deep TD
  $\theta_{t+1} = \theta_t + \alpha_t(r_\pi(S_t) + \gamma v(S_{t+1}; \theta_t) - v(S_t; \theta_t)) \nabla v(S_t; \theta_t)$

- Deep-Q-Networks
  $\theta_{t+1} = \theta_t + \alpha_t(R_{t+1} + \gamma \max_a q(S_{t+1}, a; \bar{\theta}_t) - q(S_t, A_t; \theta_t)) \nabla q(S_t, A_t; \theta_t)$

- Training: $\theta_t \rightarrow \theta_{t+1} \rightarrow \ldots \rightarrow \theta_*$

- Inference: $(s, a) \rightarrow q(s, a; \theta_*)$

In-context learning is perhaps the most trending inference problem



In-context learning is perhaps the most trending inference problem

$$
\begin{aligned}
&S_1 \rightarrow r_\pi(S_1) \\
&S_2 \rightarrow r_\pi(S_2) \\
&\dots \\
&S_n \rightarrow r_\pi(S_n) \\
&s \rightarrow
\end{aligned}
$$

forward pass of LLM / Transformer →

In-context learning is perhaps the most trending inference problem

$$S_1 \to r_\pi(S_1)$$
$$S_2 \to r_\pi(S_2)$$
$$\dots$$
$$S_n \to r_\pi(S_n)$$
$$s \to$$

forward pass of LLM / Transformer →

$$r_\pi(s)$$

What about predicting value?

$$S_1 \to r_\pi(S_1)$$
$$S_2 \to r_\pi(S_2)$$
$$\dots$$
$$S_n \to r_\pi(S_n)$$
$$s \to$$

forward pass of LLM / Transformer →

$$v_\pi(s)$$

Humans predict value via TD

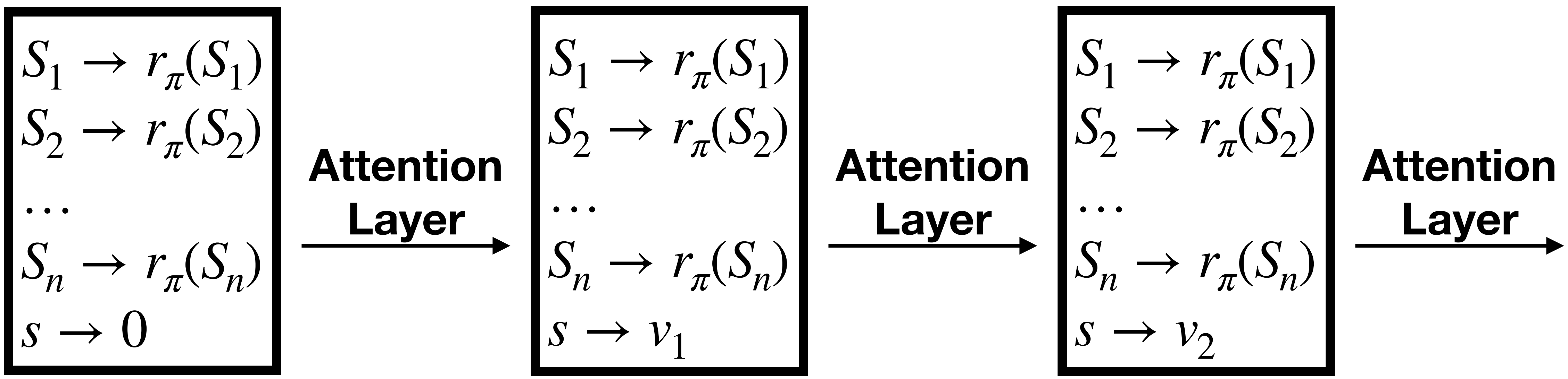
$$
\begin{array}{l}
S_1 \to r_\pi(S_1) \\
S_2 \to r_\pi(S_2) \\
\dots \\
S_n \to r_\pi(S_n) \\
s \to
\end{array}
$$

$$w_{k+1} = w_k + \alpha_k(r_\pi(S_k) + \gamma x(S_{k+1})^\top w_k - x(S_k)^\top w_k)x(S_k)$$

$$w_0 \to w_1 \to w_2 \to w_3 \to \dots$$

$$w_0^\top x(s) \to w_1^\top x(s) \to w_2^\top x(s) \to w_3^\top x(s) \to \dots$$

Transformers **CAN** mimic what humans do!



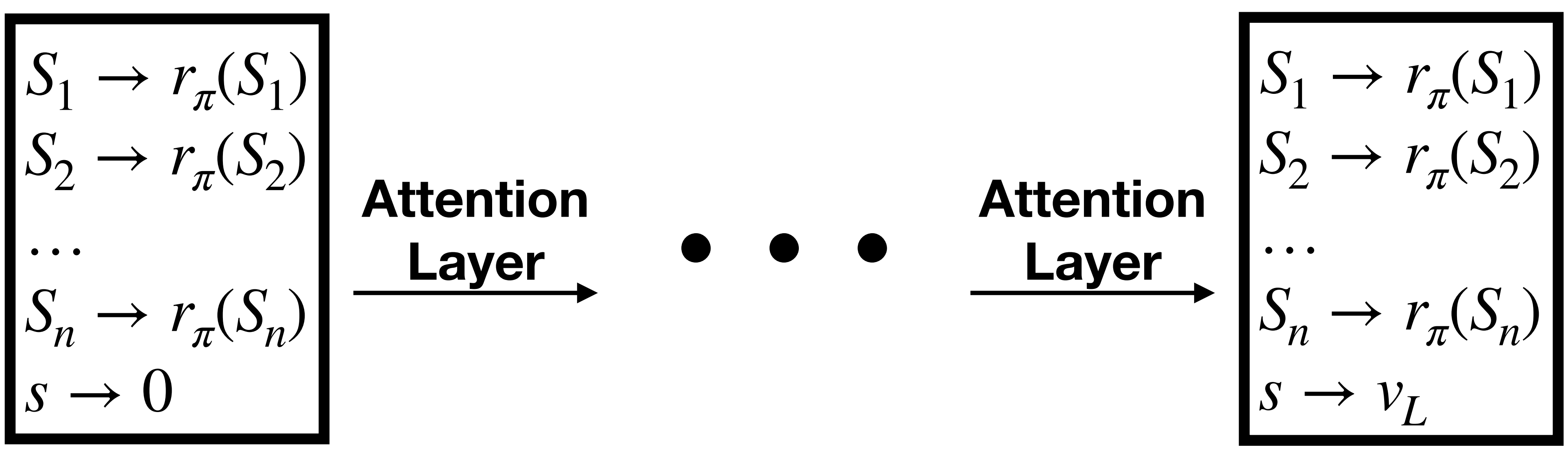
$$v_1 = w_1^\top x(s) \qquad v_2 = w_2^\top x(s)$$

(If the linear attention layer has special weights)

Transformers **DO** mimic what humans do!

- Deep TD
  $$\theta_{t+1} = \theta_t + \alpha_t(r_\pi(S_t) + \gamma v(S_{t+1}; \theta_t) - v(S_t; \theta_t)) \nabla v(S_t; \theta_t)$$

- Parameterize $v(\text{context}, s; \theta)$ as an L-layer transformer

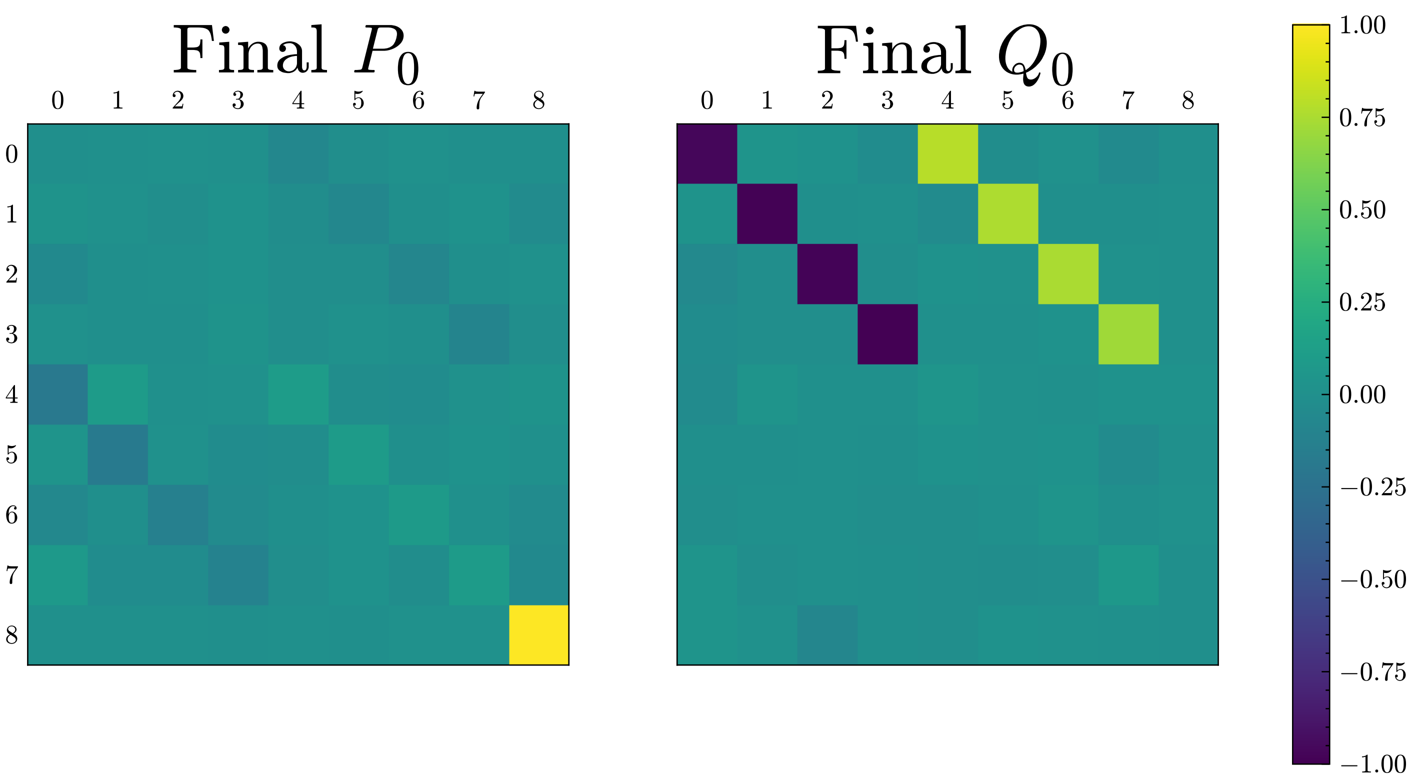


Transformers DO mimic what humans do!

- Run deep TD to train the L-layer transformer $v(c, s; \theta)$
$$\theta_{t+1} = \theta_t + \alpha_t(r_\pi(S_t) + \gamma v(c, S_{t+1}; \theta_t) - v(c, S_t; \theta_t)) \nabla v(c, S_t; \theta_t)$$

- The weights that implement in-context TD emerge after training!

$$\theta_\infty = (P_0, Q_0)$$



WHY do transformers mimic what humans do?

- Run deep TD to train the L-layer transformer $v(c, s; \theta)$
  $\theta_{t+1} = \theta_t + \alpha_t(r_\pi(S_t) + \gamma v(c, S_{t+1}; \theta_t) - v(c, S_t; \theta_t)) \nabla v(c, S_t; \theta_t)$

- The weights that implement in-context TD form an invariant set of the deep TD update

Transformers can implement more RL algorithms

- Residual gradient
- TD(λ)
- Average-reward TD

In-context regression as gradient descent (Ahn et.al. 2023)

- RL algorithm is **NOT** gradient descent
  The algorithm in inference is **NOT** gradient descent
  The training algorithm is **NOT** gradient descent

- RL prediction is inhomogeneous

In-context regression as gradient descent (Ahn et.al.)

- To implement average-reward TD
  - multiple head linear attention
  - overparameterized prompt

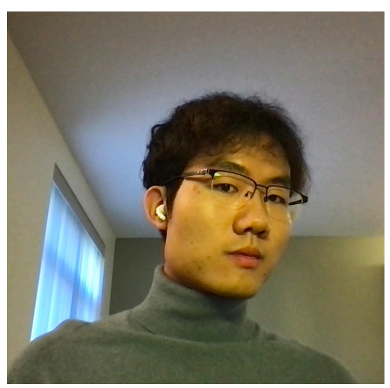
$$
\begin{array}{l}
S_1 \to r_\pi(S_1) \\
S_2 \to r_\pi(S_2) \\
\dots \\
S_n \to r_\pi(S_n) \\
s \to
\end{array}
\qquad\qquad
\begin{array}{l}
S_1 \to r_\pi(S_1)X \\
S_2 \to r_\pi(S_2)X \\
\dots \\
S_n \to r_\pi(S_n)X \\
s \to
\end{array}
$$

Shuze Liu
(UVA)

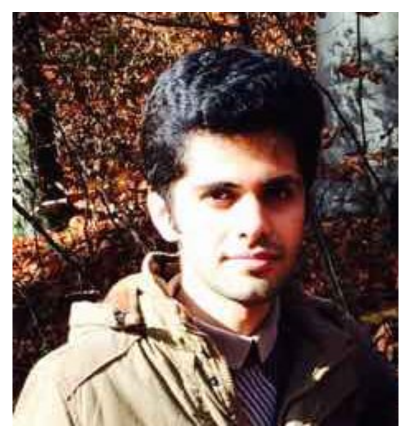
Shuhang Cheng
(Scaled Foundation)

The ODE paper:
https://arxiv.org/abs/2401.07844

Ethan Blaser
(UVA)



Jiuqi Wang
(UVA)



Hadi Daneshmand
(MIT)

The in-context TD paper:
https://arxiv.org/abs/2405.13861

Thanks!