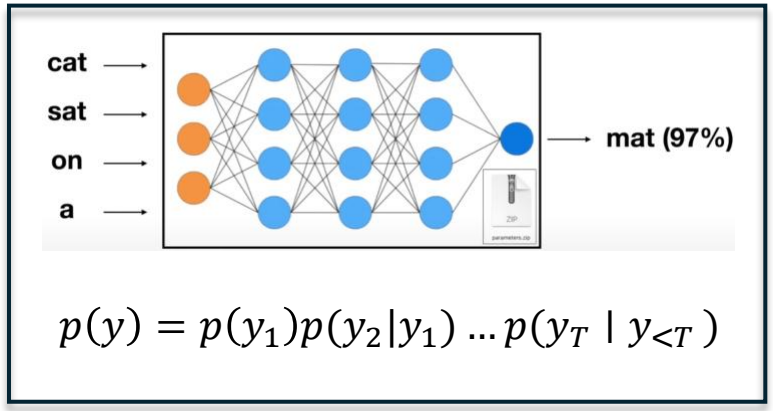
Success Conditioning as Policy Improvement: The Optimization Problem Solved by Imitating Success

Daniel Russo (Columbia University)

To appear at ICML 2026

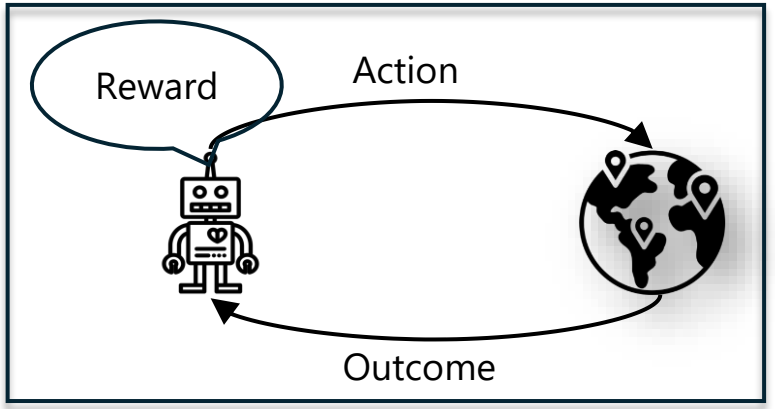
Question on my mind lately: Can sequence prediction be the engine of RL?

Offline sequence prediction



Challenge: learn (to sample from) the distribution of sequences of observed data.

Interactive decision-making

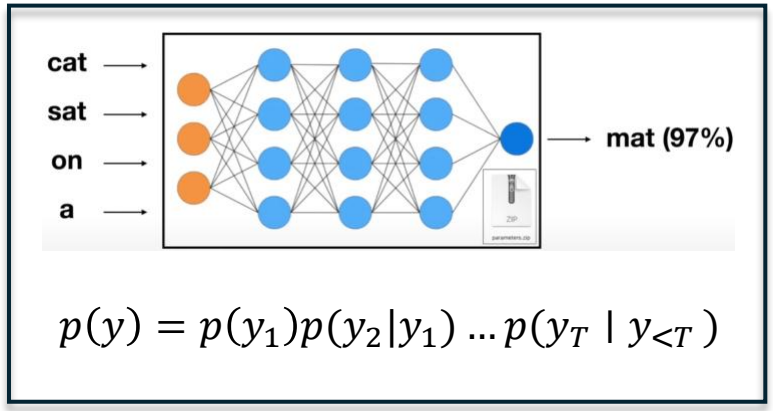


Challenges: learn via interaction to select sequences of actions toward a goal.

Which RL primitives can be recast as selective sequence prediction and autoregressive generation?

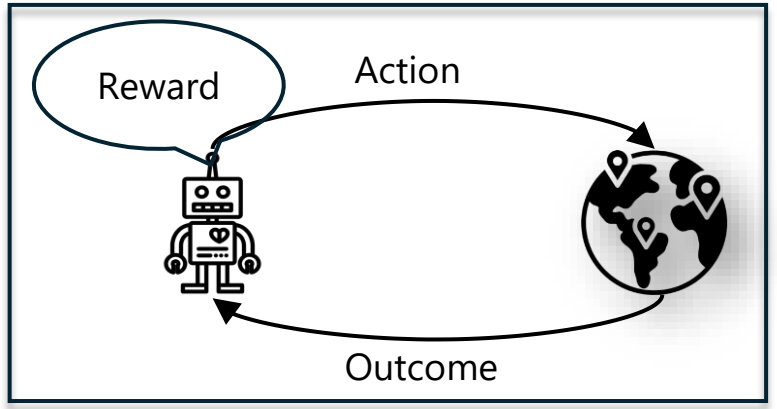
Question on my mind lately: Can sequence prediction be the engine of RL?

Offline sequence prediction



Challenge: learn (to sample from) the distribution of sequences of observed data.

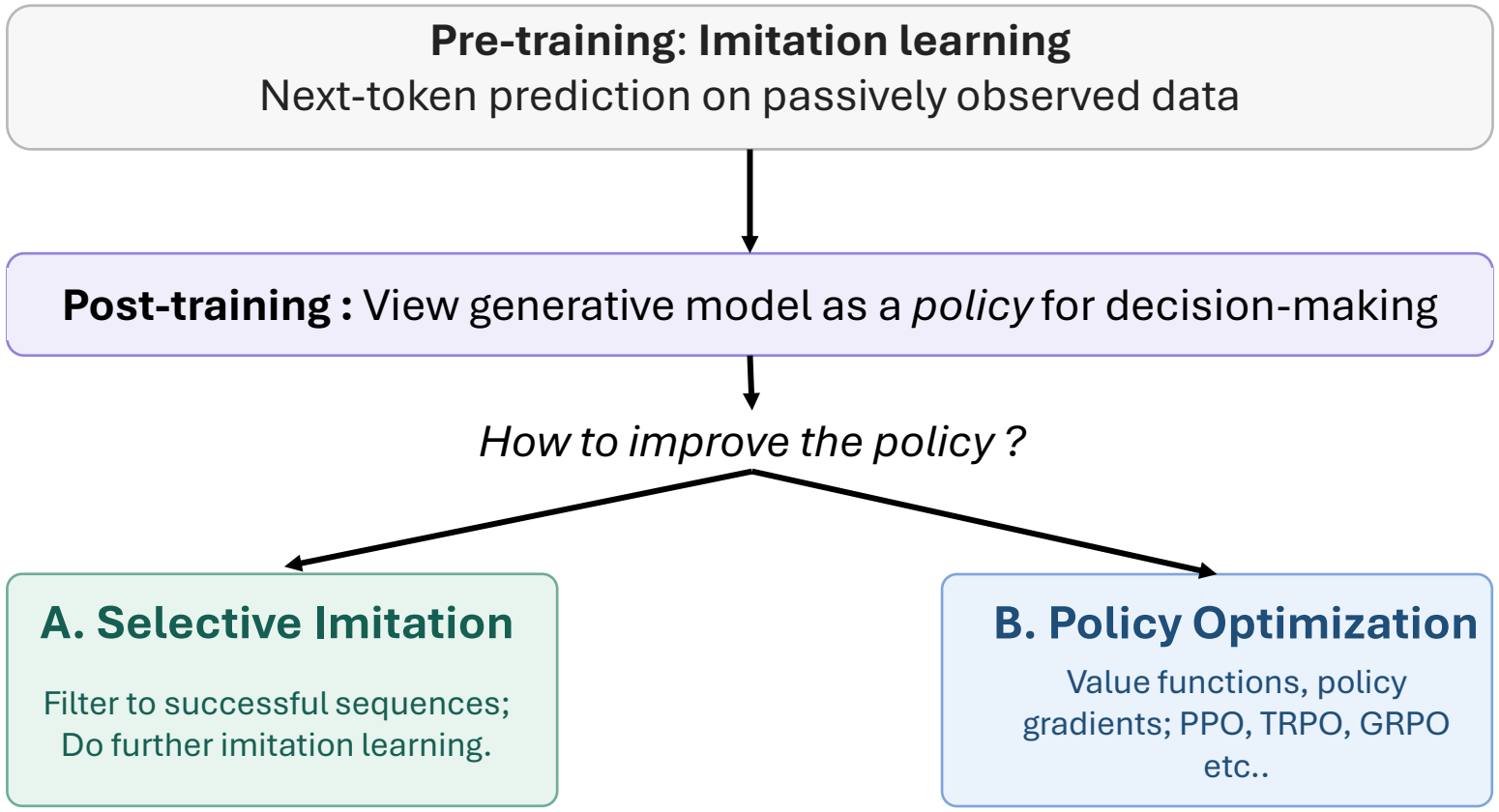
Interactive decision-making



Challenges: learn via interaction to select sequences of actions toward a goal.

- Work with KW Zhang, T Cai, H Namkoong: epistemic uncertainty quantificaiton and active exloration as autoregressive generation of missing data.
- Today‘s paper: policy improvement as sequence prediction on filtered data

Quick Teaser: Current pre-training -> post-training paradigm



Appears also in offline RL (e.g. decision transformers)

Quick Teaser: Current pre-training -> post-training paradigm

Pre-training: Imitation learning
Next-token prediction on passively observed data

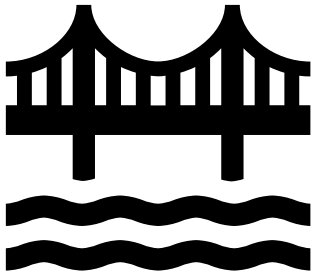
Post-training : View generative model as a *policy* for decision-making

How to improve the policy ?

Appears also in offline RL (e.g. decision transformers)

A. Selective Imitation

Filter to successful sequences; Do further imitation learning.



B. Policy Optimization

Value functions, policy gradients; PPO, TRPO, GRPO etc..

This paper: Interpreting selective imitation as policy optimization
Formalize A as "success conditioning." Show it is, implicitly, the exact solution to a trust region optimization problem (of type B).

Background and motivation

Post-training: two paths to improve the policy

A. Selective imitation

Rejection sampling + SFT

B. Explicit Policy Optimization

Reinforce, TRPO, PPO,GRPO,....

The recipe

1. Generate trajectories (*from base policy*)
2. Keep the ones that succeeded
3. Do supervised learning on them – just like pretraining

The recipe

1. Form an objective (maximize advantage)
2. Take gradient steps, or solve a local trust-region problem via many gradient steps

✓ **Same training technique as pretraining**

× No explicit decision-making objective

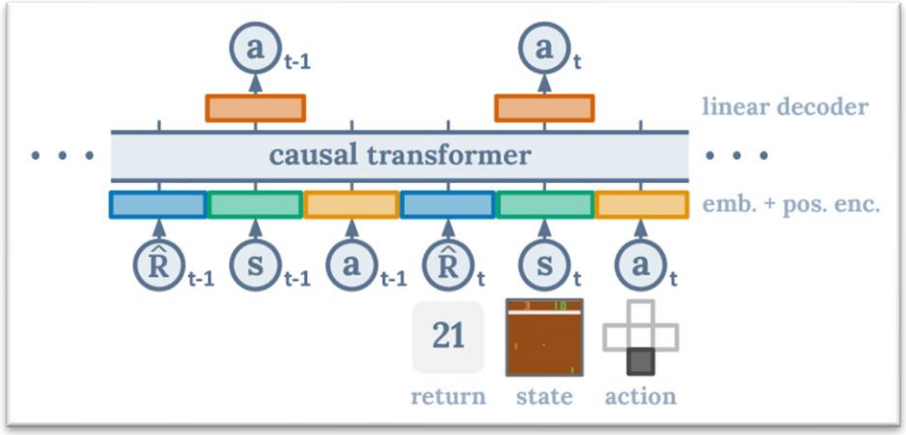
✓ **Explicitly optimize decision-making objective**

× Different, more complex, training stack. On-policy assumptions strain at scale.

Success conditioning – the same idea, two communities

Offline RL

Conditioned on the desired return, generate actions you'd have taken.



Return conditioned-RL, Upside-down RL; Decision-Transformers

LLM Post-training

Generate text or code; filter for success, do supervised fine-tuning on them.

- Alignment *(Llama 2/3)*
- Tool use *(Toolformer)*
- Reasoning *(STaR; DeepSeek-R1)*
- Robotics *(Ghasemipour et al)*

> These are the same operation: *learn to act as you would have, given you eventually succeeded*. I pull them together under the name **"success conditioning"**

Toward principled understanding

Informal comments in Chen et al's (2021) abstract:

> "Unlike prior approaches to RL that fit value functions or compute policy gradients, Decision Transformer simply outputs the optimal action by leveraging a causally masked transformer."

Follow ups: (Brandfonbrener et al. 2022) (Paster et al. 2022) (Yang et al, 2023)

> *When viewed as a route to optimal policy recovery, success conditioning has fundamental limitations, especially in stochastic environments.*

If success conditioning does not recover optimal policies, what is it optimizing for?

Formalizing success conditioning

Formulation: Episodic Markov Decision Process

- Trajectory: $\tau = (S_1, A_1, \ldots, S_{T-1}, A_{T-1}, S_T)$
- Terminal states: $\mathcal{T} = \{\text{successful states}\} \cup \{\text{failed states}\}$
- Reward: $R(\tau) \in \{0,1\}$ *(success/failure)*
- Stochastic "behavior" policy π_0 generates observed trajectories

Success Conditioned policy: mimics the distribution of actions along successful trajectories:

$$\pi_+(a \mid s) = P_{\pi_0}(A_t = a \mid S_t = s, R(\tau) = 1)$$

$$\pi_+ = \arg\min_{\hat{\pi}} \mathbb{E}\left[-\sum_{t=1}^{T} \log \hat{\pi}(A_t|S_t) \,\middle|\, R(\tau) = 1 \right]$$

Language modeling example

States & Actions:

- State S_t: sequence of tokens generated so far
- Action $A_t \in \mathcal{V}$: next token from vocabulary
- Transition: deterministic append $(S_t, A_t) \to S_{t+1}$

Behavior policy π_0: base model's next-token distribution

Success $R(\tau) = 1$: completed sequence meets criterion

- Correct answer
- Positive human rating
- Desired outcome

Success conditioning: Supervised fine-tuning on successful completions
$\to$ Imitates token-level choices from trajectories that ended in success

Failure mode of success conditioning

Example (Two-Armed Bandit)

- Single-step problem (no state)
- Arm 1: succeeds with probability 0.495
- Arm 2: succeeds with probability 0.505
- Behavior policy: uniform (pulls each arm 50%)
- Overall success rate: 0.5

Success-conditioned policy (by Bayes' rule):

$$\pi_+(a) = \frac{P(R = 1 \wedge A = a)}{P(R = 1)} \implies \begin{cases} \pi_+(1) = 0.495 \\ \pi_+(2) = 0.505 \end{cases}$$

Failure mode of success conditioning

Example (Two-Armed Bandit)

- Single-step problem (no state)
- Arm 1: succeeds with probability 0.495
- Arm 2: succeeds with probability 0.505
- Behavior policy: uniform (pulls each arm 50%)
- Overall success rate: 0.5

Success-conditioned policy (by Bayes' rule):

$$\pi_+(a) = \frac{P(R = 1 \wedge A = a)}{P(R = 1)} \implies \begin{cases} \pi_+(1) = 0.495 \\ \pi_+(2) = 0.505 \end{cases}$$

Success conditioning closes 1% of the optimality gap. It "fails" safely and observably by barely changing the behavior policy.

Action-influence *(This is a new term and definition)*

Definition (Action-Influence): The average influence of the action-draw at state s is the squared coefficient of variation of the Q-value at s:

$$\mathcal{I}_{\pi_0}(s) := \left(\frac{\mathrm{Stdev}_{a\sim\pi_0(\cdot|s)}[Q_{\pi_0}(s,a)]}{\mathbb{E}_{a\sim\pi_0(\cdot|s)}[Q_{\pi_0}(s,a)]} \right)^2$$

Notation

- Success prob: $\rho(\pi) = P_\pi(R(\tau) = 1)$.
- Value function: $V_\pi(s) = P_\pi(R(\tau) = 1 \mid S_t = s)$
- Q-function: $Q_\pi(s,a) = P_\pi(R(\tau) = 1 \mid S_t = s, A_t = a)$
- Advantage: $A_\pi(s,a) = Q_\pi(s,a) - V_\pi(s)$
- Avg Advantage: $A_\pi(s,\pi') = \mathbb{E}_{a\sim\pi'(\cdot|s)}[A_\pi(s,a)]$

This is small if (i) the behavior policy doesn't explore or (ii) actions lead to similar success rates
In the bandit example, action-influence $\mathcal{I}_{\pi_0} = 0.0001$, capturing limited signal.

Main result 1:
Success conditioning as a policy improvement operator

Twitter level view of this result…

Setup: Unify popular methods in post-training (SFT + rejection sampling) and offline RL (e.g decision-transformers) under the name **success conditioning.**

Pervasive view: Success conditioning sidesteps "RL" because it doesn't estimate value functions / policy gradients or explicitly optimize any objective.

Our work: But success conditioning is "*secretly*" a local policy optimization method broadly in the same family as TRPO, PPO, GRPO, and your likely any other __PO.

Disambiguation: This equivalence holds 'on policy', when filtering to successful generation from the policy you're improving.

Trust region policy optimization – *a generic template*

Standard form:

$$\max_{\pi} L_{\pi_0}(\pi) \quad \text{s.t.} \quad \sum_{s} w(s) D(\pi(\cdot|s); \pi_0(\cdot|s)) \leq \Gamma$$

Key design choices:

1. **State weights** $w(s)$: which states matter most?
2. **Divergence** D: geometry of the trust region
3. **Radius** Γ: how far can we move?

- Captures many policy search methods: TRPO, PPO, MDPO, etc.
- *Next slide -> What is the radius controlling? What is L?*

Formal statement of Main Result 1: Success conditioning as a conservative policy improvement operator

Success conditioning π_+ is the *exact solution* to:

$$\begin{aligned} \max_{\pi} \quad & L_{\pi_0}(\pi) \\ \text{s.t.} \quad & \sum_s d^+_{\pi_0}(s)\chi^2(\pi(\cdot|s)\|\pi_0(\cdot|s)) \leq \sum_s d^+_{\pi_0}(s)\mathcal{I}_{\pi_0}(s) \end{aligned}$$

where:

- $L_{\pi_0}(\pi) = \sum_s d_{\pi_0}(s)A_{\pi_0}(s,\pi)$ is first-order improvement
- $d^+_{\pi_0}(s)$ is occupancy under successful π_0 trajectories
- χ^2 divergence: $\chi^2(P\|Q) = \mathbb{E}_{x\sim Q}\left[\left(\frac{P(x)}{Q(x)} - 1\right)^2\right]$
- Radius $\Gamma = \sum_s d^+_{\pi_0}(s)\mathcal{I}_{\pi_0}(s)$ determined by data

Deferred to later in the talk: comparison to the forward and reverse KL constraints that dominate the literature,

Comparing geometries: success conditioning uses an especially conservative trust region

$$\max_{\pi} L_{\pi_0}(\pi) \quad \text{s.t.} \quad \sum_{s} w(s) D(\pi(\cdot|s); \pi_0(\cdot|s)) \leq \Gamma$$

	Success conditioning	**Mirror Descent PO style**	**TRPO style**
Divergence	$\chi^2(\pi_{\text{new}}(\cdot\,|s)\,||\pi_{\text{old}}(\cdot\,|s))$	$\text{KL}(\pi_{\text{new}}(\cdot\,|s)\,||\pi_{\text{old}}(\cdot\,|s))$	$\text{KL}(\pi_{\text{old}}(\cdot\,|s)\,||\pi_{\text{new}}(\cdot\,|s))$
Formula	$\text{Var}_{a\sim\pi_{\text{old}}(\cdot|s)}\left(\frac{\pi_{\text{new}}(a|s)}{\pi_{\text{old}}(a|s)}\right)$	$\mathbb{E}_{a\sim\pi_{\text{new}}(\cdot|s)}\left[\log\left(\frac{\pi_{\text{new}}(a|s)}{\pi_{\text{old}}(a|s)}\right)\right]$	$\mathbb{E}_{a\sim\pi_{\text{old}}(\cdot|s)}\left[\log\left(\frac{\pi_{\text{old}}(a|s)}{\pi_{\text{new}}(a|s)}\right)\right]$
What it enforces	**Exploitation-friendly**: penalizes new policy for trying previously rare actions.	**Exploitation-friendly**: penalizes new policy for trying previously rare actions	***Exploration*-friendly**: penalizes new policy dropping old actions
Tolerance for exploring rare action $\pi_{\text{old}}(a) = \delta$	Restrictive: $\Theta\left(\sqrt{\delta}\right)$	Tolerant: $\Theta\left(\frac{1}{\log(1/\delta)}\right)$	Permissive: $\Theta(1)$

Recap: Success-conditioning as policy optimization

Success conditioning is a *conservative policy improvement operator.*
It maximizes policy improvement without venturing too far outside observed behavior.

> ***Theorem (Informal):*** *The success conditioned policy is the exact solution to a trust region policy optimization problem with unique geometry (χ^2 instead of KL) and a radius determined automatically by the action-influence under the behavior policy.*

✓ **Same training technique as pretraining**

× No explicit decision-making objective

This result →

✓ Actually, it secretly optimizes an objective

Main result 2:
An identity for the improvement magnitude

Main Result (2): Magnitude of improvement

Theorem: (Improvement = Movement = Action-influence)

For any state s:

$$\underbrace{\frac{A_{\pi_0}(s,\pi_+)}{V_{\pi_0}(s)}}_{\text{Relative advantage}} = \underbrace{\chi^2(\pi_+(\cdot|s)\|\pi_0(\cdot|s))}_{\text{Policy change}} = \underbrace{\mathcal{I}_{\pi_0}(s)}_{\text{Action-influence}}$$

An exact relation between

1. *The policy improvement: what you care about, but see after deployment.*
2. *The policy change:* something you directly observe, a **useful diagnostic**
3. *The action-influence:* the property of the MDP and π_0 driving everything

A conservative operator: moves exactly how much it improves.

When it fails, it does so observably, by hardly changing the policy at all.

Consequence 1: Failure mode

Theorem: (Improvement = Movement = Action-influence)

For any state s:

$$\underbrace{\frac{A_{\pi_0}(s,\pi_+)}{V_{\pi_0}(s)}}_{\text{Relative advantage}} = \underbrace{\chi^2(\pi_+(\cdot|s)\|\pi_0(\cdot|s))}_{\text{Policy change}} = \underbrace{\mathcal{I}_{\pi_0}(s)}_{\text{Action-influence}}$$

Failure mode of exact success conditioning: if action-influence is small:

- The policy barely changes
- Policy improvement is correspondingly small

This low action-influence regime is exactly the failure mode seen in the bandit example.

When does the failure mode occur?

Definition (Action-Influence): The average influence of the action-draw at state s is the squared coefficient of variation of the Q-value at s:

$$\mathcal{I}_{\pi_0}(s) := \left(\frac{\mathrm{Stdev}_{a \sim \pi_0(\cdot|s)}[Q_{\pi_0}(s,a)]}{\mathbb{E}_{a \sim \pi_0(\cdot|s)}[Q_{\pi_0}(s,a)]} \right)^2$$

Q. When is action influence (and policy improvement) small?

A. If stochasticity in the behavior policy doesn't greatly impact final outcomes.

- *No two actions result in substantially different success probabilities*
- *Or the policy is nearly deterministic, so random draws don't matter.*

Consequence 2: A potential diagnostic

If I perform success conditioning offline, and deploy online, how do I anticipate the extent of policy improvement before deployment?

For any state s:

$$\underbrace{\frac{A_{\pi_0}(s,\pi_+)}{V_{\pi_0}(s)}}_{\text{Relative advantage}} = \underbrace{\chi^2(\pi_+(\cdot|s)\|\pi_0(\cdot|s))}_{\text{Policy change}} = \underbrace{\mathcal{I}_{\pi_0}(s)}_{\text{Action-influence}}$$

Quantity	Directly observable?
Relative policy improvement	After deployment, not from offline data
Action-influence	Requires Q-values that success conditioning purposefully avoids estimating
Policy change in χ^2	Directly observable from the policy we train.

Policy movement measures policy improvement *before deployment.*

Extensions

Corollary: monotonic improvement (skipped)

Success conditioning always improves (or maintains) performance:

$$\rho(\pi_+) \geq \rho(\pi_0)$$

At any state s:

$$\frac{V_{\pi_+}(s) - V_{\pi_0}(s)}{V_{\pi_0}(s)} \geq \mathcal{I}_{\pi_0}(s) \geq 0$$

Extension: impact of estimation error (skipped)

If estimated policy $\hat{\pi}$ achieves excess cross-entropy loss $\Delta = \ell(\hat{\pi}) - \ell(\pi_+)$, then:

$$|\rho(\hat{\pi}) - \rho(\pi_+)| \leq \sqrt{M \cdot \Delta/2}$$

where $M = \sup_s \frac{d^+_{\pi_0}(s)}{d_{\pi_+}(s)}$ measures distribution shift.

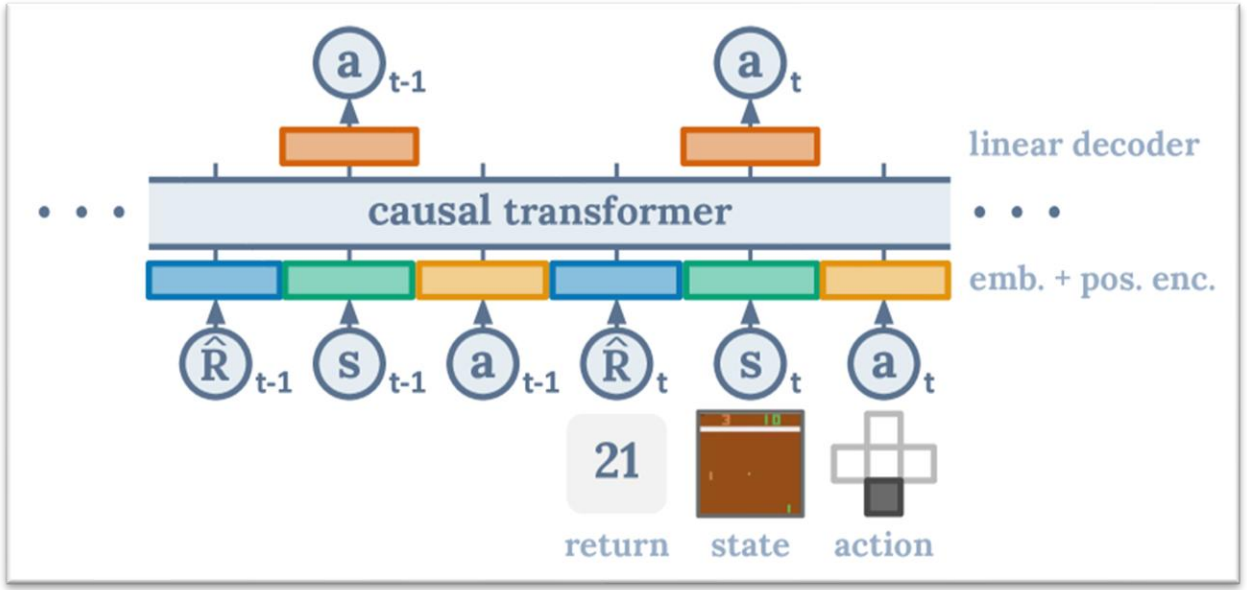
Excess sequence prediction loss controls online decision-making performance, upto a distribution shift factor.

- Note, M=1 in typical RLHF + RLFVR.

Dense rewards and return thresholding

Return conditioned-RL, Upside-down RL; Decision-Transformers

Conditioned on the desired return (i.e. cumulative reward) of a realized trajectory, generate actions you'd have taken.

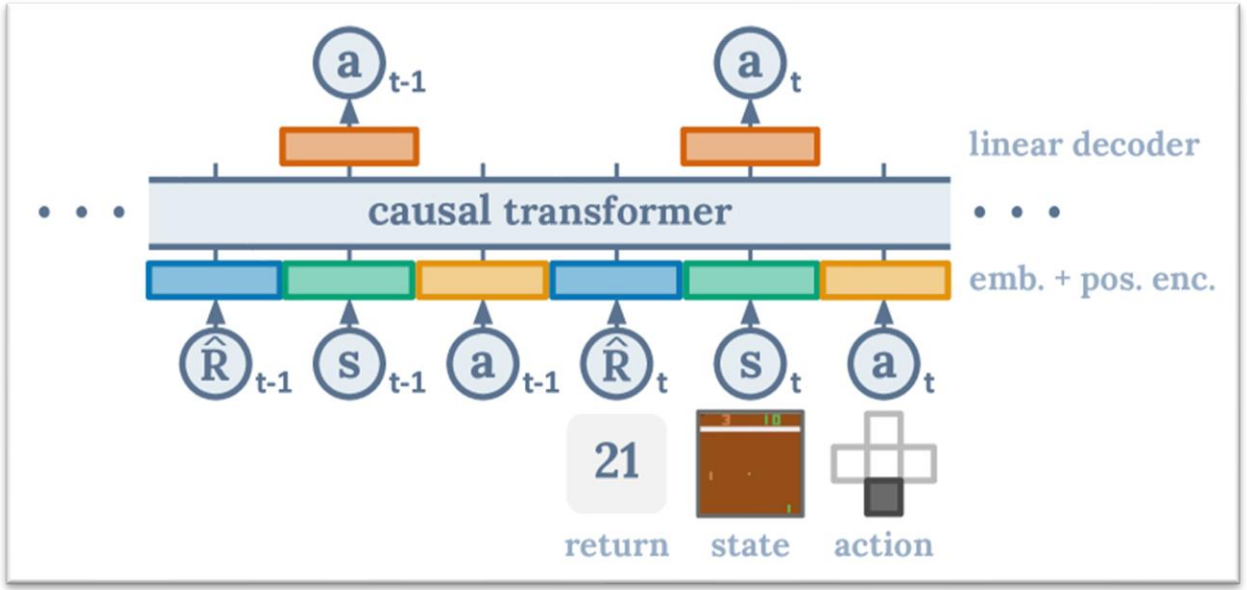


Vague goal: How should we understand this practice?

Dense rewards and return thresholding

Conditioned on the desired return (i.e. cumulative reward) of a realized trajectory, generate actions you'd have taken.

Return conditioned-RL, Upside-down RL; Decision-Transformers



- **Training reward**: return thresholding is conditioning on a binary success measure
- **Evaluation reward**: papers still evaluate on average (dense) reward earned.

Refined goal: systematically understand the implications of this mismatch

Toward precisely studying return-thresholding

Step 1: if rewards are not binary, what would it mean to faithfully apply success-conditioning?

True dense reward: $\mathrm{Y}(\tau) \in (0,1)$
Randomized rounding: Set binary reward $\mathrm{R}(\tau)|\mathrm{Y}(\tau) \sim \mathrm{Bernoulli}(\mathrm{Y}(\tau))$
Equivalence of objectives noise: $\mathbb{E}[\mathrm{Y}(\tau)]$ and $\mathbb{E}[\mathrm{R}(\tau)]$ agree for every policy.

Success conditioning on binary $R(\tau)$ is an improvement operator for $\mathbb{E}[Y(\tau)]$

Return thresholding is success conditioning on a mis-aligned binary reward:

- Train on: $\widetilde{\mathrm{R}}(\tau) = 1(\mathrm{Y}(\tau) > \theta)$; Eval on $\mathbb{E}[\mathrm{Y}(\tau)]$

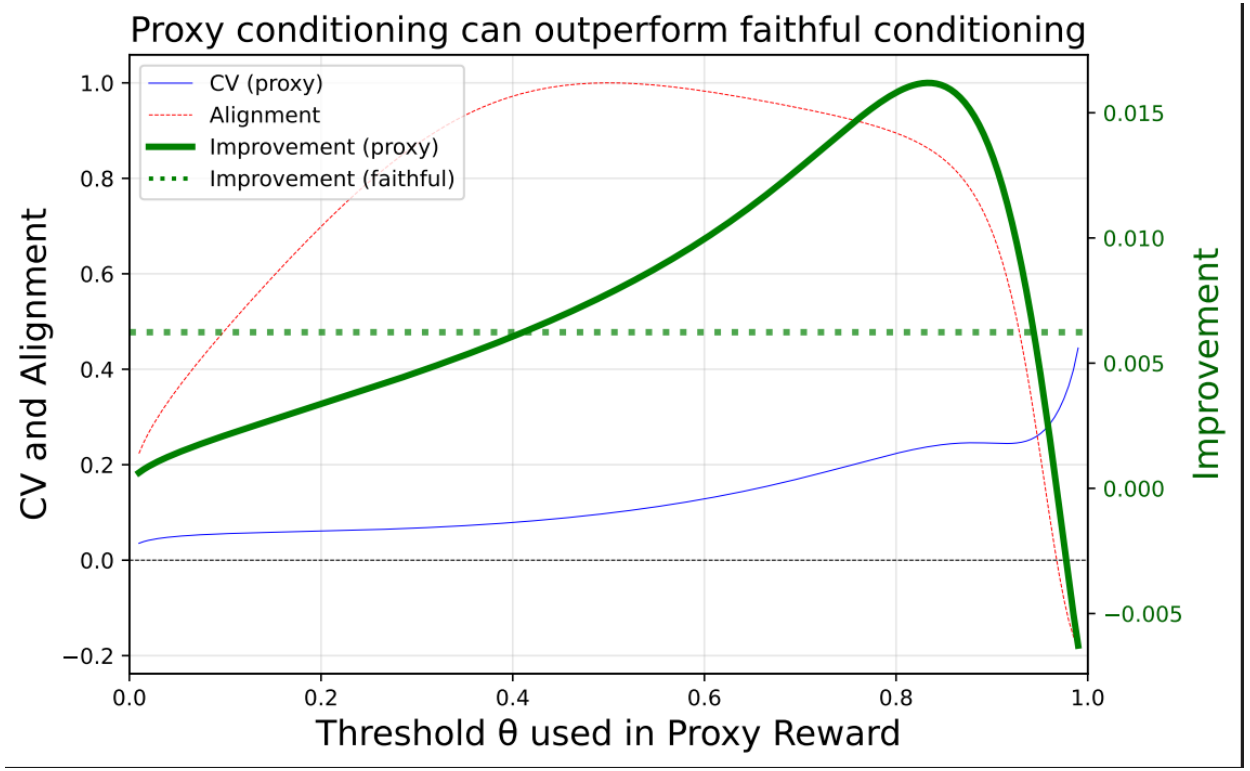
When and why is this a good idea?

Theory of return conditioning

$$\frac{\text{Improvement of SC with proxy reward } \widetilde{\mathrm{R}}}{\text{Improvement of SC with faithful binary R}} = \frac{A_{\pi_0}(s, \tilde{\pi}_+)}{A_{\pi_0}(s, \pi_+)} = \sqrt{\frac{\tilde{\mathcal{I}}_{\pi_0}(s)}{\mathcal{I}_{\pi_0}(s)}} \cdot \mathrm{Corr}(A_{\pi_0}, \tilde{A}_{\pi_0})$$

Thresholding on a high return:

- Can amplify action influence and improvement.
- Risks severe misalignment with true objective.



Conclusion and open questions

- We've formalized disparate methods as 'success conditioning.'
- Interpreted success conditioning as policy improvement.
- Action-influence quantifies gains from success conditioning

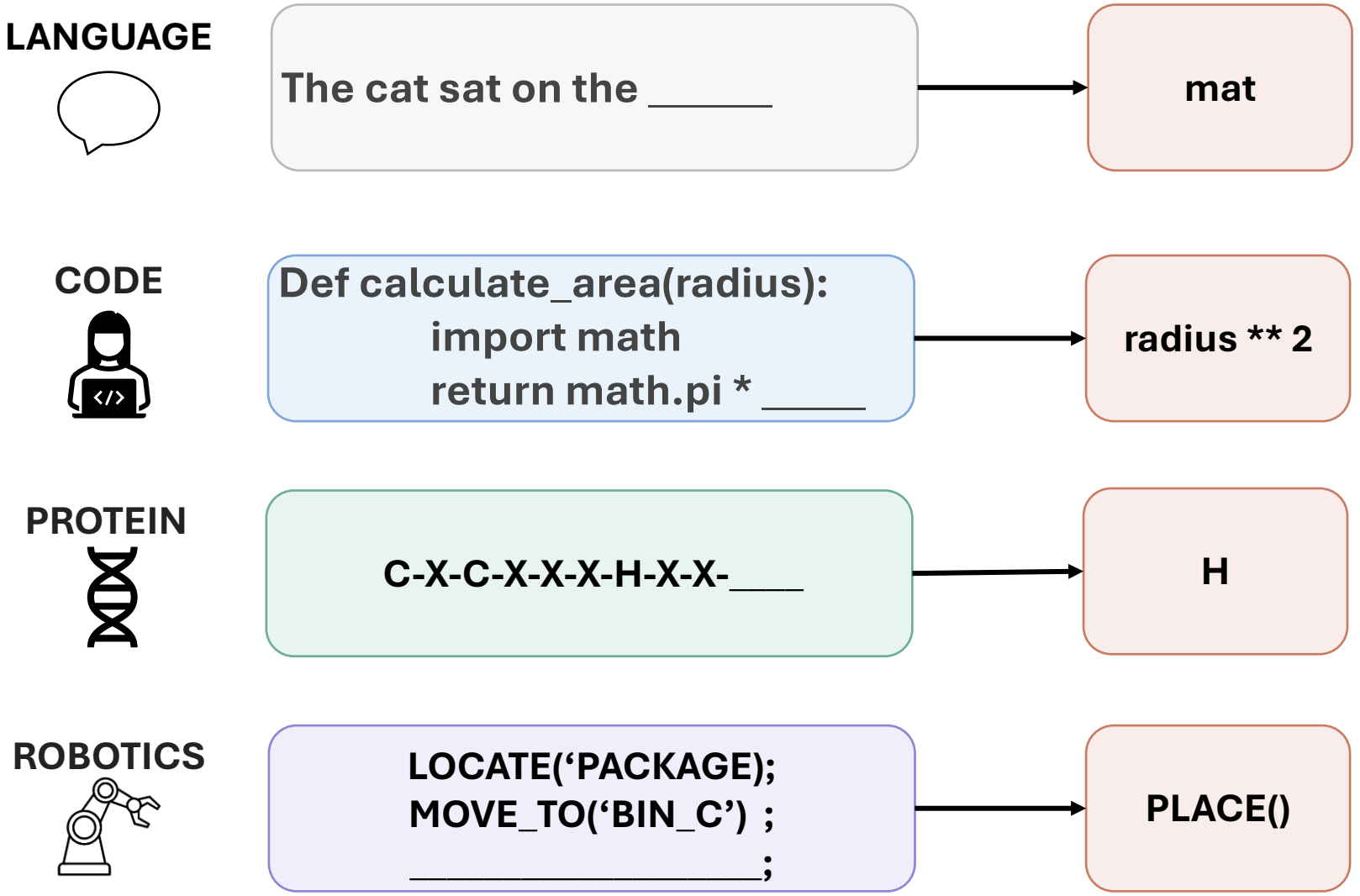
<u>Some next directions (talk to me offline)</u>

1. **Off-policy** trajectories and exploration: how far *can success conditioning be pushed*.
   - *Intuitively, SC is already more off-policy friendly. + It has the gradient direction of reinforce, but with the stale data-generating distribution of the behavior policy.*
2. When is policy improvement **a useful diagnostic** for current practice.
3. Convergence of **iterated success conditioning** (*exploration collapse is the subtelty)*

Thanks! Questions?

Backup slides

Pre-training: imitate data at massive scale



Often, predicting what comes next teaches the model to *imitate* human actions...

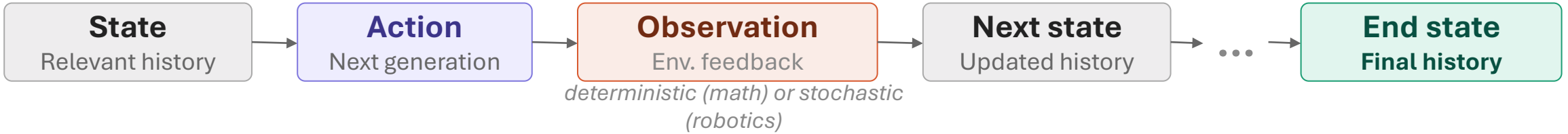
But there is no reward signal or clear decision-making objective.

Toward post-training: generative models as decision-making policies

Math example trajectory:

Robotics example trajectory:

General structure (episodic MDP):



Shift: view generative model as a decision-making policy.

Next, Given a base policy from pretraining, how do we improve it?

Quick intro to policy improvement theory

Taylor expansion of the success probability:

$$\rho(\pi) = \rho(\pi_0) + \underbrace{\sum_s d_{\pi_0}(s) A_{\pi_0}(s,\pi)}_{:=L_{\pi_0}(\pi)} + \underbrace{\sum_s \left(d_\pi(s) - d_{\pi_0}(s)\right) A_{\pi_0}(s,\pi)}_{:=\mathrm{Rem}_{\pi_0}(\pi)}$$

where d_π is the occupancy measure

$$d_\pi(s) = \mathbb{E}_\pi \left[\sum_{t=1}^{T-1} 1(S_t = s) \right]$$

- **Trust regions methods:** optimize L while controlling the Dist. Shift term Rem.
- **Interpretation:** Relates the value of small, persistent policy to changes π_0 (via ρ) to the advantage of *one-period* deviations from π_0 (via L)