Paper : General agents need world models

Key claims :

1)Any agent that performs well in such tasks must implicitly or explicitly learn a model of the environment.

2) This internal model can be reconstructed from the agent's policy, showing that even seemingly model-free agents contain "latent predictive structure."

3) As agent performance or goal complexity increases, the fidelity of the learned world model must also increase.

Central Question : Do intelligent agents like humans or advanced AI systems need world models to generalize and act in complex environments, or can model-free methods suffice?

Humans excel at zero-shot and few-shot learning. we can solve new tasks with very little prior experience.
This ability is increasingly seen in large language models (e.g., GPT-3), prompting the community to explore the next frontier

" Can we build general-purpose agents that handle long-term goals in real-world environments?"

In cognitive science, it's well established that humans don't just react to stimuli.we maintain mental models of the world:

These models help us set abstract goals (not just based on current input).

They allow us to plan actions ahead of time (deliberative behavior).

So, world models are central to flexible, intelligent decision-making in humans.

There are two camps:

Model-based AI: Build agents that explicitly learn the environment's dynamics.
Benefits: safer, lower sample complexity, better planning and transfer.
Challenge: real-world environments are complex and hard to model accurately.

Model-free AI (Brooks' view): Don't learn a model at all.... just react via experience and reinforcement.
Pros: often works in practice and avoids hard modeling.
Examples: RL agents like DQN or even some LLM-based agents.

However, there's growing evidence that even model-free agents may learn implicit models or planning structures during training

Is it possible to reach human-level AI without explicitly learning a world model? Or is some form of modeling fundamentally required?

Key insights

Given just the policy of a well-performing agent (a goal-conditioned agent), we can extract an approximation of the environment's transition dynamics.

The better the agent (or the harder the tasks it can solve), the more accurate this extracted world model becomes.

The Formal Answer: Yes, World Models Are Necessary
The authors prove:

If an agent can solve many simple goal-directed tasks with bounded regret (i.e., performs reasonably well),
Then it must have learned a predictive model of the environment.
This is true even if:

The agent wasn't explicitly trained to predict.
The architecture is purely model-free.
There are no rationality or optimality assumptions.

Notation

Capital letters (e.g., X) represent random variables.

Lowercase letters (e.g., x) represent realized values of those variables.

Bold letters (e.g., X) represent sets of variables, such as {X1, X2, ..., Xm}.

Propositions in brackets (e.g., [X = x]) are true/false statements, often used for conditioning or logical clarity.

A Controlled Markov Process (cMP) is like a Markov Decision Process (MDP) but without rewards or a discount factor.

States: S Actions: A Transition function: Pss'(a) = Pr(S = s' | A = a, S = s)

a cMP describes how the world evolves in response to actions, but it does not define objectives (no reward yet).

Trajectories and Histories

A trajectory τ = (s0, a0, s1, a2, ...) is a sequence of states and actions over time.

A history h = (s , a , ..., st) is the trajectory up to time t representing what the agent knows so far.

Definition 1: Controlled Markov Process (cMP)

A cMP is defined as a tuple (S, A, P ss' (a))- just states, actions, and transitions.

No rewards or discounting -only dynamics.

Assumption 1: Standard Environment Conditions

To keep results clean and general, the authors assume:

Finite state and action spaces Irreducibility: any state is reachable from any other Stationarity: transitions don't change over time

|A| ≥ 2: at least two actions, allowing real choice

Goals Using Linear Temporal Logic (LTL)

Rather than just saying "reach state X," LTL lets us specify when something should happen using temporal operators.

Temporal Operators

Operator Meaning Example

⊤ (Now) Must be true now ([S = s]) → be in state s now

◯ (Next) Must be true in the next step ◯([S = s]) → reach s next step

<> (Eventually) Must be true at some point <>([S = s]) → eventually reach s

These allow time-sensitive goal specifications like:

<> ([S = s]) → Eventually reach state s

◯([A = alert, S = seng]) → Next, alert the engineer in state seng

Simple and Composite Goals

Goals can be combined to express richer tasks.

1. Sequential goals Perform sub-goals in order:

ψ = <φ1, φ2, φ3> → Do φ , then φ , then φ .

2. Alternative goals Satisfy any one of multiple goals:

ψ = ψ1 <LOGICAL DISJUNCTION> ψ2 → Succeed if either ψ or ψ is achieved.

Agent Definition

An agent is a policy π that takes:

A history h = (s0, a0, s1, ..., st),

A goal ψ, And outputs the next action at.

Formally: π(ht, ψ) → at

This is a goal-conditioned policy.

Optimal Agent

An optimal agent maximizes the chance of achieving any goal ψ from any state s :

π* = argmax π Pr(τ |= ψ | π, s)

Where:

π* is the optimal policy.

argmax π denotes the policy π that maximizes the expression.

Pr(τ |= ψ | π, s0) is the probability that trajectory τ satisfies the goal ψ, given policy π and starting state s0.

τ |= ψ means the trajectory τ satisfies the goal ψ.

Core Idea:

Real agents don't need to be perfect—just good enough.

In practice, AI agents (robots, assistants, RL policies):

Work in complex environments

Handle long, multi-step tasks

Sometimes fail

Instead of requiring optimality, the authors define bounded agents that are:

Effective on tasks with up to n sub-goals

Successful within a factor (1 − δ) of the optimal agent

Formal Definition:

For all goals with complexity ≤ n, the agent's success rate must be at least:

(1 − δ) × (optimal success rate)

This ensures the agent is reliably competent—not perfect, but close.

Example: Maintenance Robot

A bounded agent chooses the path with higher success probability.

Goal: Fix a machine (long sequence) or alert an engineer (shorter path)
We measure how close it comes to the optimal agent to compute δ.

No Strong Assumptions

The agent doesn't have to be fully rational or optimal.

This is key real-world agents (like LLMs or heuristics) often act imperfectly, yet we still want to measure their capabilities.

Why It Matters

If an agent consistently performs well on structured tasks despite no explicit world model, it likely has one implicitly.

Competence implies structure, and that structure is a world model.

world model is not just a map of what the environment looks like. it's a predictive model of how the environment changes in response to actions.

In formal terms: $\hat{P}_{ss'}(a) \approx P_{ss'}(a) = \Pr(S_{t+1} = s' \mid S_t = s, A_t = a)$

P^is the learned approximation, and the error is bounded by ε.

authors want to formally prove that even if an agent is not explicitly trained to model the world, if it performs well on a range of goal-directed tasks, then it must internally encode an approximate world model.

THEOREM 1

A goal-conditioned agentπ (Definition 5)

That performs reasonably well (with failure rate δ) on goals of depth up to n

There exists a predictive model, denoted as P, that can be extracted solely from the agent's behaviorthat is, from its policy. This model satisfies a provable error bound.

$$|\hat{P}_{ss'}(a) - P_{ss'}(a)| \leq \frac{(n-1)(1-\delta)}{2\sqrt{P_{ss'}(a)(1-P_{ss'}(a))}}$$

The error between the agent's inferred world model and the true transition model is bounded, and this bound depends on how good the agent is (small δ) and how complex the goals it can handle (large n).

And asymptotically, for small δ and large n:

$$|\hat{P}_{ss'}(a) - P_{ss'}(a)| = \mathcal{O}(\delta/\sqrt{n}) + \mathcal{O}(1/n)$$

As the agent becomes more competent (as δ approaches 0) and is capable of handling increasingly complex goals (as n approaches infinity), the approximation error decreases at the following rates:

O(δ / n): error due to imperfect performance

O(1 / n): error due to limited goal complexity

They describe an algorithm (Algorithm 1) that operates as follows:

It takes only the agent's policy as input.

It queries the agent with composite goals of the form ψ = ψ_a ψ_b, where ψ_a and ψ_b are incompatible goals.

It observes the action the agent chooses in response.

The chosen action indicates which goal the agent "believes" is more achievable.

This, in turn, reveals information about the agent's understanding of the environment-specifically, the transition probabilities.

The algorithm is unsupervised (it does not require labels or access to the environment's ground truth) and universal (it can be applied to any agent that meets the underlying assumptions).

Main Idea: The Better the Agent, the Better Its World Model

The accuracy of the learned world model improves when:

δ → 0: The agent performs closer to perfectly.

n → inf: The agent can handle longer, more complex goals.

Key takeaway: Even if the agent isn't very good (say, δ is around 1), as long as it can solve long-term goals (large n), it still needs to have a fairly accurate model of the environment.

So you can trade off:

Skill (competence)

Goal depth (how complex or long the tasks are)

Improving either one helps the agent learn how the environment works.

Rare Events Are Hard to Learn

When the model's error is divided by how often a transition actually happens, you get the relative error. This error becomes huge when the transition is very unlikely.

$$\text{Relative Error} = \frac{|\hat{P}_{ss'}(a) - P_{ss'}(a)|}{P_{ss'}(a)} \to \infty \text{ as } P_{ss'}(a) \to 0$$

The prediction for such rare events can be very wrong, and that's okay under the theoretical bounds.

If something rarely happens in the environment, the agent doesn't need to learn it accurately.

Humans and AI agents usually ignore very rare events.

Only when goals get more complex or performance improves do agents need to understand more of the environment.

They focus on what happens often or what matters for achieving goals.

Limitation: Short-Sighted Agents (n = 1)

If the agent only solves one-step (immediate) tasks:

The main theorem gives a weak (trivial) result.

It's unclear if the agent truly avoids building a world model or if the math just doesn't capture it.

To fix this, the authors plan to create a new result focused on myopic agents those that:

Can only handle immediate tasks

Completely fail at longer tasks (δ = 1 when n > 1)

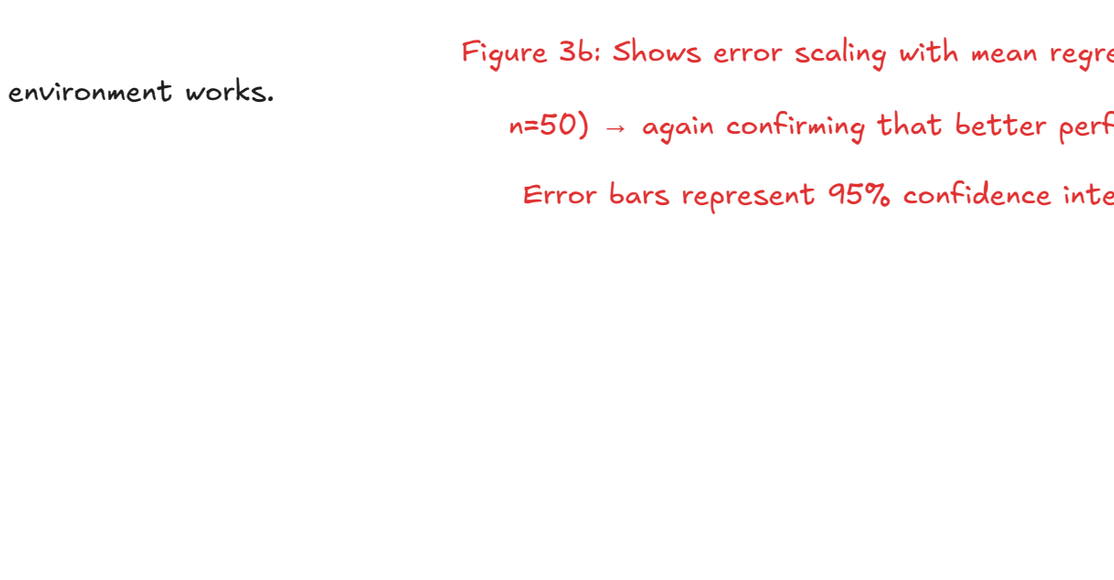

Figure 3a: Shows that the mean error in the recovered world model decreases as agents generalize to deeper goals → confirming Theorem 1.

Figure 3b: Shows error scaling with mean regret for high-depth goals (e.g.,

n=50) → again confirming that better performance = better implicit world model.

Error bars represent 95% confidence intervals over 10 independently trained agents.

Theorem 2: Myopic Agents Don't Need World Models

A myopic agent only aims to achieve immediate, depth-1 goals those that must be satisfied in the next time step. It doesn't plan ahead, just reacts.

In standard environments (finite, irreducible, stationary cMPs), an optimal myopic agent reveals no information about the environment's transition function.

Formally:

|P_ss'(a) − P_ss'(a)| ≤ ε = 1

This trivial bound means the agent's policy is consistent with any transition function in [0,1], offering no insight into how the environment works.

Implication

You cannot recover even partial transition dynamics from a myopic agent's behavior.
The policy doesn't rely on or encode a world model. it simply maps observations to actions.

Key Consequences

Myopic agents don't learn or need world models.

World models are essential only for:

Multi-step planning

Sequential reasoning

Generalizing across time

This contrasts with Theorem 1, which shows that agents solving long-horizon tasks must encode a world model.

* Any agent with bounded regret on multi-step goals must learn a world model.
* Model-free approaches cannot avoid learning a world model.
* Explicit model-based architectures are better positioned to leverage this necessity.
* Implicit world models may explain emergent capabilities in foundation models.
* Learning a small set of structured tasks can imply enough knowledge to generalize broadly.
* World models support planning, uncertainty estimation, domain adaptation, and social reasoning.
* They also enable causal reasoning, counterfactual simulation, and intent attribution.
* Accurate world models are essential for AI safety, interpretability, and alignment.
* Extracting world models from agents provides a path toward safer, auditable AI.
* Strong AI is limited by the feasibility of learning accurate world models in complex, open systems.
* Generalization is fundamentally constrained by world model fidelity.
* Intractable environments prevent agents from satisfying regret bounds on long-horizon tasks.
* Online learning remains necessary in domains where models cannot be learned in advance.

Planning and inverse reinforcement learning (IRL) aim to recover goals or policies given the environment. This work completes the triangle by recovering environment dynamics from goals and policy.

Unlike IRL, which requires optimal policies across environments, this method only needs policies over multiple goals.

While mechanistic interpretability recovers internal representations via activations, this approach directly recovers world models from policies.

Algorithm 1 is unsupervised, architecture-agnostic, and works even when model internals are inaccessible. It's more general than probing or sparse autoencoders (SAEs), which are tied to specific models and environments.

This method recovers predictive transition functions P-hat_ss'(a), not just latent states. It targets the actual environment dynamics, though the agent's subjective model can also be recovered with modified assumptions.

Myopic agents don't need to learn world models, limiting the relevance of representation theorems focused on them.

Domain generalization requires more causal understanding than task generalization.

Future work could apply this method to LTL-based agents that generalize to formal temporal logic goals.

Classical representation theorems (e.g., Savage's) rely on strong rationality assumptions and don't explain how dynamics are learned.

By assuming only bounded regret, this work provides a more realistic and flexible framework than utility-based approaches.

The Good Regulator Theorem doesn't prove that agents learn predictive models only that they form control rules.

This work gives stronger evidence of learned world models than entropy-minimization arguments.

The results are consistent with psychological and neuroscience theories suggesting agents build models of their environments.

Rather than assuming world models exist, this work shows that behavioral competence implies their presence.

Algorithm: Estimate Transition Probability P-hat_ss'(a) from Policy

Inputs:

* Agent's policy: π(at | ht, ψ)
* Current state: s
* Action to evaluate: a
* Outcome state: s'
* Alternative action: b ≠ a
* Precision level: n (higher n means more accurate estimate)

Steps:

1. Initialize:
Set k* = n (this will track when the agent begins to prefer action a)

2. Loop for each k from 1 to n:

Define sub-goals:

* φ : "Take action a"
* φ : "Take action b"
* φ : "Next state must be s'"
* φ : "Next state must be anything other than s'"

Build short goal sequences:

* ψ = φ , φ → Take b, then reach s' (Fail path)
* ψ = φ , φ → Take b, then avoid s' (Success path)

Construct composite goals:

* ψa(k, n): Succeed if agent gets ≥ k successes out of n trials of ψ
* ψb(k, n): Succeed if agent gets ≥ k successes out of n trials of ψ

Combine goals:

* ψ = ψa(k, n) ψb(k, n)

Ask the agent: "What action would you take in state s, given the goal ψ?"

If the agent chooses action a:

* Set k* = k
* Break the loop

3. Estimate the transition probability:
P-hat_ss'(a) = (k* − 0.5) / n

Return:
Estimated transition probability P-hat_ss'(a)