

# Active Agents:

## An Active Inference Approach to Agent-Based Modeling in the Social Sciences

*Welcome – Please access Colab script + Slides at:*

<https://github.com/apashea/IC2S2-Active-Inference-Tutorial>

**Andrew Pashea**

Email: [andrewbpashea@gmail.com](mailto:andrewbpashea@gmail.com)

LinkedIn : <https://www.linkedin.com/in/andrewbpashea/>

*Active Inference Institute Intern, <https://www.activeinference.org/>*

*Data Analyst, K-8 Public Charter School Network*

*UChicago Social Sciences Division Alum*

*Harris School of Public Policy Applied Data Fellow Alum*

# Overview: Tutorial Context and Caveats

## **Conference themes:**

- Agent-based modeling: from “traditional” to “cognitive” with hands-on code walkthrough
- Cognitive science and theoretical discussions: applications of an established cognitive science framework, “Active Inference”, to CSS/ABM
  - No direct ‘ActInf for ABM’ framework nor multi-agent coding tutorial (currently) exists; so far most research is at an ‘advanced’ level
- Reproducibility: developing intuitions and fundamental knowledge for reading and writing in Active Inference, open-source programming resources, open-source code for learning and adaptation

## **Format:** Three Parts (with breaks!)

- Part 1: Agent-based modeling
  - Some motivations and guidelines, The Cognitive Turn, RL, ActInf, comparisons and research examples
- Part 2: Active Inference in-depth
  - Building intuition, cognitive mechanisms, ABC(DEFG)’s of Inference
- Part 3: Learning by Doing
  - Code walkthrough: recreation of a ‘traditional’ ABM of parallel-problem solving in networks, now with ActInf agents

# Overview: Tutorial Context and Caveats

## Preferred prerequisite knowledge:

- Probability theory and distributions, e.g., sum to 1, non-negative probability values, categorical distributions, Bayesian theory
- Optimization problems, e.g., predictive modeling, cost function minimization, gradient descent
- Basic-to-intermediate programming, e.g., conditional logic, loops/iterations and algorithmic thinking, functions, Python lists/dictionaries/DataFrames
- *That said, we will be building agents from scratch*

## Developed as resource for learners at the Active Inference Institute:

- ‘Living document’: Not expected to remain the same; various tools are being developed for optimizing code and computational run-time
- Some ‘slideshow karaoke’: slides should be comprehensible as a standalone resource for asynchronous learners
- Not a holistic learning resource: Active Inference is vast and rapidly expanding into various domains; ‘further reading’ references provided
- “As simple as possible, but not simpler”: ~3 hour conference tutorial is not enough to cover field holistically; we will cover various aspects (and maths involved) in a pragmatic fashion
  - intention: introduce ActInf to broader audience, providing intuition about the field as well as a more direct application example

# Agent-based Modeling: Key Terms & Overview

**Agent-based modeling**: ABMs are used in CSS to simulate and research social phenomena artificially. Artificial agents/entities are produced in an artificial environment, typically via computer programming, and interact with each other and their environment during simulations. This provides a safe, artificial environment for hypothesis-testing, theory-building, and counterfactual explorations in policy design and the study of emergent collective behavior prior to real-world implementation.

**Micro-to-macro**: By coding agents and their environment with the right sets of initial micro-level assumptions, ABMs can be used to study emergent macro-level phenomena resulting from collective behavior, e.g., social interaction, network dynamics, organizational productivity, spread of information/norms, housing segregation, impacts of financial crises, policy impact on behavior.

# Agent-based Modeling: Key Terms & Overview

**Agent**: Fundamental entities in an agent-based model, e.g., individual persons, organisms, institutions. Agents are typically endowed with the ability to act in different ways, e.g., interact with other agents, approach/avoid, explore/exploit, move (in physical space), buy/sell. Agents may be endowed with various behavioral and demographic characteristics, typically in accordance with assumptions drawn from empirical findings, e.g., survival, goal-directedness, social conformity, utility-maximization, occupation, belief adherence.

**Environment**: Agents are situated in an environment, e.g., networks, social organizations, physical spaces, markets, or other niches. Agents interact with one another and the environment itself. The environment both constricts and provides affordances to agents, e.g., agents may only interact with other agents in their own (sub)network, agents cannot move outside the boundaries of a physical environment, agents find rewards in their environment.

**Simulations**: Simulations are coded to create the agents and their environment and then executed over some duration, typically over discrete time steps ( $t=1, t=2, \dots, t=T$ ). Particular individual- and aggregate-results are recorded over time, e.g., individual agents' actions taken at each time step, aggregated purchases/sales of all agents, average distance between moving agents, other use-case specific variables.

# Agent-based Modeling: Some Traditional Guidelines

Lazer & Friedman (2007): “Agent-based modeling starts with a set of simple assumptions about agents’ behavior and inductively derives the emergent system-level behaviors that follow.” Four guideline criteria for ABMs (applicable to experimental research in general) are as follows:

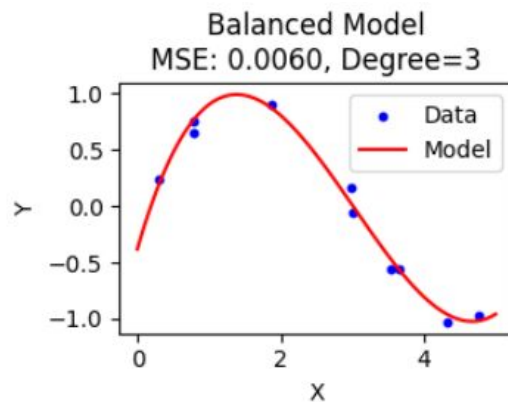
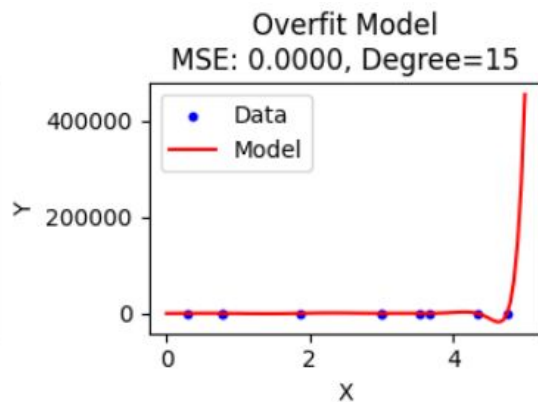
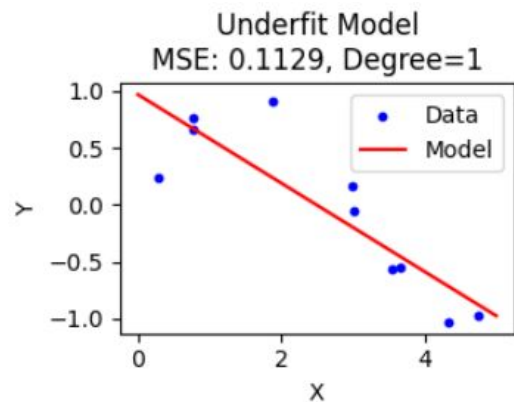
1. **Verisimilitude**: The *constrained assumptions made must be convincing to readers*, typically via being based on empirical findings in the literature. “For example, if one does not believe that the two-actor, two-choice set up of the prisoner’s dilemma captures important features of some human relations, then the rather plentiful (analytic and simulation) research on the prisoner’s dilemma has nothing to say about collective behavior.”
2. **Robustness**: Models should be *relatively simple enough* such that “results should withstand unimportant changes to the model.” Excessive complexity compromises the model’s generalizability to the phenomena of interest.

# Agent-based Modeling: Some Traditional Guidelines

Lazer & Friedman: “Agent-based modeling starts with a set of simple assumptions about agents’ behavior and inductively derives the emergent system-level behaviors that follow.” Four guideline criteria for ABMs (applicable to experimental research in general) are as follows:

3. **Reproducibility**: The model results must be fully reproducible by other researchers based on information in the paper, as well as from reasonably commented code for the simulation made available online.
4. **Non-triviality**: The results of the simulation should be non-obvious. “The objective of computational modeling should not be to reproduce what could be derived from verbal exposition; rather, it is to produce novel yet convincing insight.”

# Agent-based Modeling: Some Pragmatic Guidelines

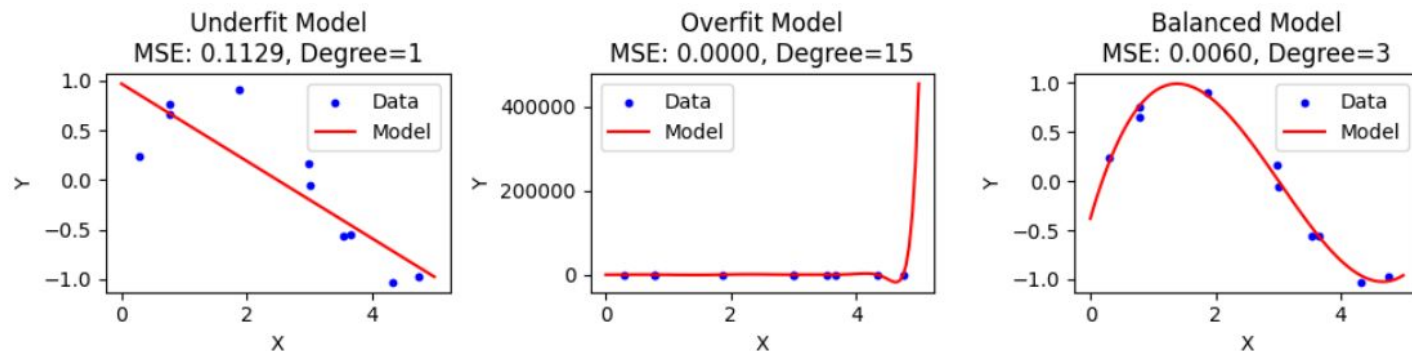


Analogous to the simplicity-complexity or underfitting-overfitting trade-offs in modelling in general, e.g., bias-variance or under-/over-fitting tradeoff in statistical modelling and machine/deep learning, theoretical models in social sciences, engineering design, biological systems modelling, ...

Example: fitting models with too few/many parameters



# Agent-based Modeling: Some Pragmatic Guidelines



“Everything must be made as simple as possible, but no simpler.”

- Albert Einstein (purportedly)

“All models are approximations. Assumptions, whether implied or clearly stated, are never exactly true. *All models are wrong, but some models are useful.* So the question you need to ask is not ‘Is the model true?’ (it never is) but ‘Is the model good enough for this particular application?’”

- statistician George E. P. Box

“A map is *not* the territory it represents, but, if correct, it has a *similar structure* to the territory, which accounts for its usefulness.”

- general semanticist Alfred Korzybski

# Agent-based Modeling: Too Simple?

“Simple as possible but no simpler” may be sufficient for approximating the bottom-up process of micro-level assumptions to macro-level emergent behavior, but how does this impact the agent?

“Traditional” ABMs tend to “hard-code” agent behavior with static rules-based action-selection as determined by the behavioral assumptions made on the part of the researcher, e.g., “if X then agents does Y” at each discrete time step, regardless of circumstance. It is “as if” agents adapt, but they do not functionally change or ‘learn’. *A thermostat is a thermostat anywhere...*

# Agent-based Modeling: Too Simple?

This presents limits of applying “traditional” ABM to simulations and/or policy design directed at individuals’ beliefs, e.g., fostering student confidence, influencing financial overconfidence, mitigating self-damaging behaviors.

Cognitive decision-making inferences and predictions are predominantly left to empirical data retrieval and analysis, a rich and growing field nonetheless divorced from the simplisiticity of traditional ABM model design.

“Through Google or Yahoo we may trace people's habits, moods, investment decisions, political views, risk propensity and attitudes toward culture, education and migration. Based on this information, we may drive production, capital movements, business strategies, political decisions, and international cooperation. But we will not be able to suggest effective plans for modifying such behaviors and the underlying mental states, unless we understand the mental dynamics and how this interacts with the social dynamics, and model the cognitive mechanisms that respond to external influence and rule behavioral change. In absence of such theory and model, we will not get to the core of hard problems” (Conte et. al, 2014).

# Agent-based Modeling: Shift to Cognitive Models

## **“Traditional” ABMs:**

Rules-based action-selection: “If this then agent does X, else agent does Y” without internal beliefs or mechanisms for autonomous action

Micro-level assumptions are kept as simple as possible, ‘hard-coded’ into simulations without real mechanisms for ‘learning’ / ‘training’

Macro-level behavior ideally mirrors empirical research, with insights yielded from modifying environment

NetLogo as predominant software with specific syntax and lacking external tool integration.

## **“Cognitive” ABMs:**

Autonomous action-selection: “Generative” agents choose own actions based on internal cognitive mechanisms, e.g., perception, action, learning, planning

Interdisciplinary work related to behavior in complex systems, e.g., cognitive science, psychology, physics and information theory, cybernetics machine-/deep-/reinforcement learning, biology, behavioral economics

Macro-level behavior ideally mirrors empirical research, with insights yielded from modifying environment *and* reviewing agents’ internal parameter changes

Various languages used, e.g., MATLAB, Python, julia.

# Agent-based Modeling: Cognitive Models

**Perception**: agents take environmental observations into account, in some representational form, e.g., probability distribution or table of values, which could be viewed as “beliefs,” “expectations,” or “predictions” with some memory mechanisms for storing this information and cognitive mechanisms and parameters for computing and learning representations.

- e.g., ML/DL models, train/test data for model optimization

**Action**: paired with perception, whereby given some observation(s) the agent chooses actions to achieve its goals; as with perception, such experiences can also be stored in memory and used to develop complex strategies which are not “hard-coded”, i.e. not necessarily expected by the modeler *a priori*

- e.g., RL agents, learning action selection via reward signals

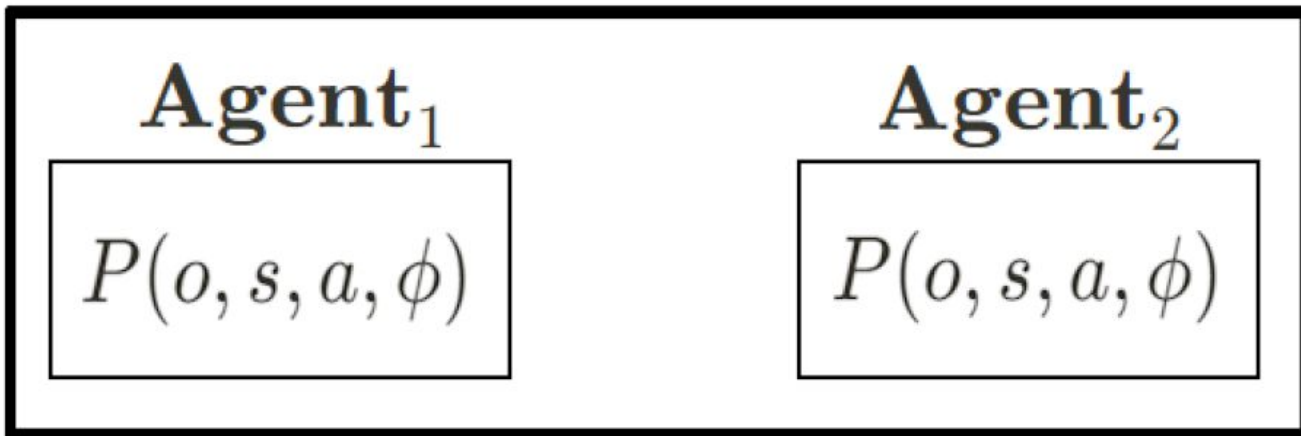
**Generative**: Agents are “generative” and tend to rely upon (often probabilistic) representations of observation-action, action-reward, and/or other variable mappings; they learn and update their own figurative “if X then do Y” rules adaptively (which may or may not include stochasticity) and *generate* actions given some input (state or observation)

# Agent-based Modeling: Cognitive Models

**Models in a model:** Cognitive/generative ABM agents thus require some model architecture for processing observations and algorithms for determining system behavior, e.g., optimization problem, reward/cost functions) – forming agents as (equipped with) *generative models*.

Our environment, with its own variables and parameters, now contains agents with *their own* variables and parameters.

**Environment** :  $(a_t, o_t)$



# Agent-based Modeling: Cognitive Models

**Some criteria:** So, which model architecture and which algorithms?

Back to Lazer & Friedman:

- Simplicity-complexity trade-off?
- Empirical/theoretical support for the cognitive mechanisms?
- Reproducible? Computational costs?
- Model/parameter interpretability? Is it useful?

Explore-exploit trade-off: With perception, action, and other cognitive mechanisms in place, autonomous agents require using actions to both gain information about rewards *and* to actually achieve reward

- Too much exploration -> less accumulated reward, risk info optimum
  - *All day studying, not a page written...*
- Too much exploitation -> risk stuck at local reward optimum
  - *...or write a terrible book-length essay without references*

What is optimality? Task-performance and/or relative to the model itself?

- Is suboptimal/pathological behavior a sign of the agent not doing what the experimenter desires, or is this in fact “optimal” to the model? ...are we building superhumans?

# Which algorithms and architectures?

## Reinforcement learning:

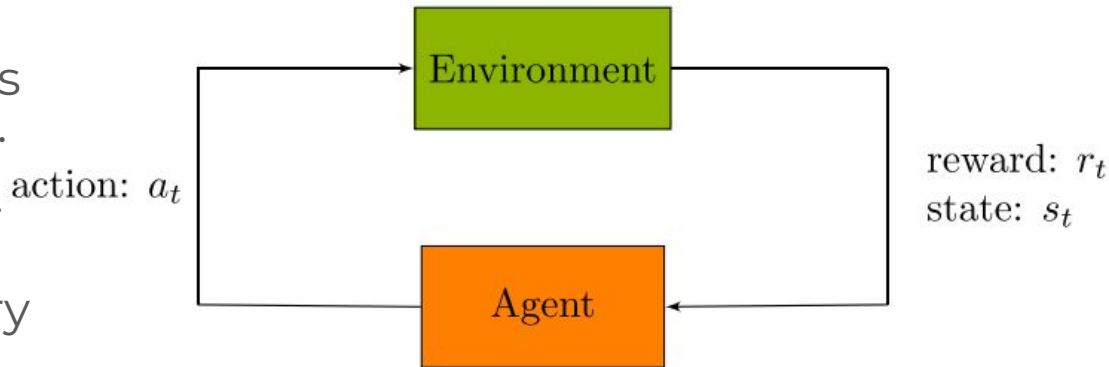
Class of algorithms which attempt to maximize a specific reward/cost function, the discounted sum of future expected rewards (Sutton et. al, 1998).

‘Model-free’ Q-learning:  
learn/update values of state-action pairs based on reward signal, without own model of transition probabilities or reward functions in advance.

Q-learning agents update their ‘Q-tables’, with valuations of state-action pairs as the primary interpretable parameters.

Computational costs are relatively low at this level of complexity.

state transition  $T : (s_t, a_t) \rightarrow s_{t+1}$ , reward function  $R : s_t \rightarrow r_t$



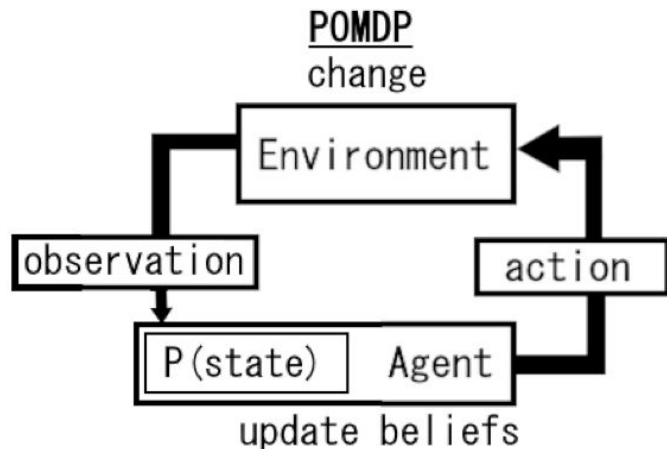
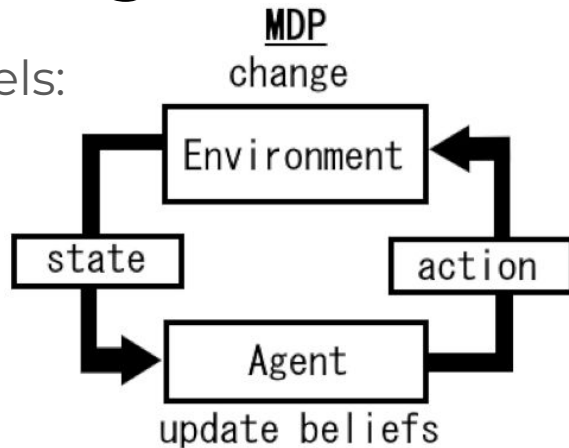
policy  $\pi : s_t \rightarrow a_t$ , expected return  $V : (s_t, a_t, \pi) \rightarrow \mathbb{R}^+$

(Emmert-Streib et. al, 2022).



# Which algorithms and architectures?

Typical base models:



**Markov Decision-Making Process (MDP)** : Environmental state is known to agent, i.e. uncertainty is only in the action-state reward valuations. Only uncertainty involved is in agent's reward valuations of action-state pairs.

**Partially-Observable Markov Decision-Making Process (POMDP)** : Environmental state is unknown, only inferred from (potentially noisy) observations, mirroring higher cognitive realism. Agents maintain and update/learn a probability distribution—its beliefs—over these “hidden” or “latent” states, based on observations (Meng et. al, 2021). This is frequently handled with more NN-related features, increasing computational complexity/cost.

# Which algorithms and architectures?

## Reinforcement learning:

Task-performance optimization: Research typically focuses on task performance optimization in other fields, e.g., robotics, AI tools and assistants.

Pavlovian learning: Some RL algorithms relatable to early Pavlovian conditioning models (better-than-expected outcomes strengthen associations between predictors and outcomes) (Collins, 2019).

...but modeling/incorporating *human* cognition is often non-priority or niche.

Deeper -> more opaque: Research trends towards 'deep' RL, using features and architectures from, for example, neural networks to approximate reward functions, but this risks 'black box' interpretability loss and rising computational costs.

Exploration/exploitation dilemma: Treated as separate problems, with additional rules for deciding, e.g., epsilon-greedy behavior, manually defining an exploration rate 'epsilon' to avoid behaviors which are repetitive or suboptimal (locally optimal) to the task. (Tijsma, 2016).

$$\text{---} \quad a_t = \begin{cases} \arg \max_a Q(s_t, a) & \text{with probability } 1 - \varepsilon \\ \text{random action} & \text{with probability } \varepsilon \end{cases}$$

# Which algorithms and architectures?

## Active Inference:

Normative Framework: A normative framework derived from first principles for studying the behavior of autonomous agents in situations requiring decision-making under uncertainty. Agents are equipped with generative models and take in observations from their environment which in turn are evaluated as (dis)confirmatory evidence for their current beliefs (Friston, textbook; and Sajid et. al, 2021). This begins with **free energy principle (FEP): “everything that changes in the brain must minimize free energy”** (Friston, 2022; Friston, 2009; and Sajid et. al, 2021).

Neural process theory: Foundations in computational neuroscience and study of neuronal dynamics and cognitive processes (including psychopathology, perception, decision-making) in animal and human behavior and in self-organizing systems more broadly. Abstract mathematical quantities – predictions, prediction errors, precision signals, etc. – manifest in neuronal dynamics, with physiological substrates (brain/organ activity and connectivity). FEP extends beyond minute continuous operations to homeostatic regulation in living organisms (Pezzulo et. al, 2015).

Generative models and no simpler: As a normative, principled framework, it has a top-down design philosophy. Rather than emulating the brain in its entirety, the challenge rather is “to find the *generative model* that describes the problem the brain is trying to solve.” (Friston et. al, 2022, pp. 104-105).

# Which algorithms and architectures?

## **Active Inference:**

Free Energy principle (FEP): Agents minimize a theoretical quantity “free energy,” equivalent to maximizing evidence for current beliefs (as opposed to expressly maximizing ‘reward’) in order to, e.g., infer states, evaluate actions, update own model of the world (observation and transition models). This leads to a (Bayesian)

**belief-based framework and naturally emergent adaptive behaviors.**

- (Bayesian) ‘surprise’ is the true quantity to be minimized but is typically intractable; free energy acts as an upper bound on surprise

Exploration-exploitation balance: ‘Epistemic’ value and ‘pragmatic’ value included in same optimization problem for agents with capacity for planning (optimizing expected *free energy*) rather than handled separately as in, say, epsilon-greedy RL agents. In fact action, perception, and other cognitive mechanisms are *all* linked together and play out in continuous free energy minimization.

“Self-evidencing” agents: Instead of external reward signals, agents seek to realize preferences conceptualized as prior beliefs over observations, relatable to “goals.” Agents can learn which observations are more rewarding, i.e. those which help free energy minimization or equivalently maximize model evidence (Sajid et. al, 2021).

Variational (Bayesian) inference schemes are used to mitigate heavy computational costs and/or potential intractability in optimization operations – not *ad hoc* but rather with conceptual foundations in metabolic constraints of biological organisms.

# Which algorithms and architectures?

## **Active Inference:**

Optimality defined as relative to the model, not the task: Like “traditional” ABM, emphasis on recreating empirical dynamics rather than superior task performance, evidential for example in the inclusion of ‘**false inference**’:

- a failure/aberration of belief updating: “linking together maladaptive behavior (e.g. compulsions or addictions) and psychological-level (e.g., false beliefs) and biological-level phenomena (e.g., abnormalities of neuromodulators).”
- in psychopathology, “the inferential mechanism operates normally, but based on a flawed generative model” (Friston, 2022, pp. 186-187).

**Competitive:** albeit when compared, Active Inference agents typically rival or outcompete (Q-learning and Deep Q-Learning) RL agents in task performance, even when disadvantaged vs. RL agents, e.g., less time spent training, initialized with imprecise beliefs about environmental state transitions (Paul et. al, 2022).

**Model architectures:** As with RL, typical baseline discrete-time models are the MDP and POMDP, built up from the HMM as a model of cognitive perception augmented with the capacity for action and planning...

...but – like RL agents – can be augmented with other architectures as well. Further, variational message passing schemes can hypothetically be applied to *any* model writeable-as a factor graph

# Active Inference and ABM: The Mindset

“Equipped with a mechanism to produce actions, an organism can engage in reciprocal exchanges with its environment.... Essentially, for each action-perception cycle, the **environment sends an observation** to the organism. The organism uses (an approximation to) Bayesian inference to infer its most likely hidden states. It then **generates an action and sends it to the environment in an attempt to make the environment less surprising**. The environment executes the action, generates a new observation, and sends it to the organism. Then, a new cycle starts.”  
*Though able to be described as a step-wise cyclical process in this way, this is considered to be a **continuous process** occurring in various timescales, where discrete-time approximates to continuous.*

“Active Inference goes beyond the recognition that perception and action have the same (inferential) nature. It also assumes that both *perception and action cooperate to realize a single objective*—or optimize just one function—rather than having two distinct objectives, as more commonly assumed. ...described in various (informal and formal) ways, including the minimization of surprise, entropy, uncertainty, prediction error, or (variational) free energy.” (Friston et. al, 2022).

# Active Inference and ABM: Scaling and Compatibility

**Agent in an agent:** As the free energy principle (FEP) plays out in broader/global systems – e.g., organisms, agents, collectives, including within each constituent subsystem – the overall free energy of the system will be continuously minimized. Here, (sub)systems are considered to be (modelled as) self-organizing entities who are statistically separable from one another by a Markov blanket (as we will see), where each subsystem is (equipped with) its own generative model. (Friston et. al, 2022).

Goal-directed, self-evidencing and self-organizing agents range from homeostasis-seeking behavior of cellular organisms up to collective human social and cultural practices (Friston et. al, 2022; and Bruineberg et al. 2018.)

*Ex. Neurons, brains and other organs, individuals, societies, ...*

---

# Active Inference and ABM: Scaling and Compatibility

## Further reading on Active Inference in relation to FEP/RL/DL/LLMs:

“Reward Maximization Through Discrete Active Inference” (Da Costa et. al, 2023).

- deeper comparison to RL and review of conditions under which Active Inference agents can be seen to be expressly maximizing reward (Bellman optimality)

“Canonical neural networks perform active inference” (Isomura et. al, 2022).

“Experimental validation of the free-energy principle with in vitro neural networks” (Isomura et. al, 2023).

“The Neural Correlates of Ambiguity and Risk In Human Decision-Making Under an Active Inference Framework” (Zhang et. al, 2024).

- Theoretical, in vitro, and neuroimaging experimental validations of neural networks exhibiting the free energy principle (maximizing model evidence; learning and plasticity)

“Generative agent-based modeling: an introduction and tutorial” (Ghaffarzadegan et. al, 2024).

“Generating meaning: active inference and the scope and limits of passive AI” (Friston et. al, 2023)

“Understanding, Explanation, and Active Inference” (Parr et. al, 2021).

“Predictive Minds: LLMs As Atypical Active Inference Agents” (Kulveit et. al, 2023).

- recent discussions of the relationship between LLMs and agents (including Active Inference agents) and augmenting the latter with the former
- LLMs-relatable to probabilistic, free energy minimizing Active Inference models but without ability to (directly) act; LLMs could augment action-capable ActInf agents with ability to verbally communicate and explain their own choices/beliefs



# Active Inference and ABM: Examples

## Theory-building examples:

“The theory of constructed emotion: an active inference account of interoception and categorization” (Barrett, 2017).

- Theory of constructed emotion as categorization of sensory inputs via process of perceptual, interoceptive active inference
- Emotions are not localizable/innate to specific parts of the brain, but are rather hypotheses/inferences the brain makes when integrating sensory information

“Active Inference and Social Actors: Towards a Neuro-Bio-Social Theory of Brains and Bodies in Their Worlds” (Cheadle et. al, 2024).

- Built sociological theory with view of brains as “enculturated” with emergent social dynamics, including a view towards a “probabilistic sociology” based in Active Inference

“Generative models, linguistic communication and active inference” (Friston et. al, 2021).

- Linguistic theory built from simulating communicating Active Inference agents (hierarchical POMDPs) who exhibit free energy minimization and theta-gamma coupling found in empirical neuroscience research on cognitive ‘learning’

# Active Inference and ABM: Examples

## Examples towards treatment interventions:

“Simulating the computational mechanisms of cognitive and behavioral psychotherapeutic interventions: insights from active inference” (Smith et. al, 2021).

- single-agent simulations of exposure therapy, analyzing relationships between explicit and implicit beliefs, and facilitation of belief-updating, in hypothetical patients with, e.g., stress/phobic disorders, fear-stimuli trauma associations
- find that agent’s implicit beliefs about stimulus as threat (and associated negative physiological outcomes, e.g., high arousal, negative affect) more likely to change in cases of ambiguity, suggesting therapy should emphasize/encourage ambiguity of negative stimulus to facilitate belief-updating rather than strictly safe

“Active learning impairments in substance use disorders when resolving the explore-exploit dilemma: A replication and extension of previous computational modeling results” (Taylor et. al, 2023).

- A more recent study from a series of studies involving fitting Active Inference-based Bayesian models to empirical data collected from individuals with and without substance use disorders engaging in a task
- Find those with substance use disorders exhibit slower learning in cases of negative task outcomes
- Suggest ‘learning rate’ model parameter may be a novel target for treatment intervention

# Active Inference and ABM: Examples

## Examples of adaptation to changing contexts::

“Active Inference for Autonomous Decision-Making with Contextual Multi-Armed Bandits” (Wakayama et. al, 2023).

- Simulation of agents in CMAB tasks where the ‘contexts’ (and corresponding rewards) change, leading to adaptation of actions/policies to different settings, with exploration-exploitation balancing via free energy minimization; comparison to other strategies shows fewer iterations and lower computational cost required for task-performative Active Inference agents

“Deeply Felt Affect: The Emergence of Valence in Deep Active Inference” (Hesp et. al, 2021).

- Use a POMDP with an additional hierarchical ‘affective’ (cortical) layer to analyze impact of affective responses to stimuli on agent’s action precision (‘subjective confidence’) to achieve its goals (T-maze experiment)

Interactive inference: a multi-agent model of cooperative joint actions” (Maisto et. al, 2022).

- two-agent simulations which recreate dynamics of empirical experiments of two-person tasks (‘leaderless’ and ‘leader-follower’ joint actions) where agents observe outcomes of own actions as well as partner’s actions

# Active Inference and ABM: Examples

## Simulating $N$ agents in networks:

“Active inferants: An active inference framework for ant colony behavior” (Friedman et. al, 2021).

- Simulate  $N$  agents (ants) collectively foraging, leaving observations (pheromones) in their environment for fellow agents to follow to reach goal (food source) in physical space
- Agents adapt to context changes (shifting food source location)

“Epistemic Communities under Active Inference” (Albarracin et. al, 2022).

- Simulate  $N$  agents (individuals using social media) sending and observing tweets with two opposing opinions in a social network (formally as connections between agents on random graphs with connection probability  $p$ )
- Run various parameter sweeps (changing Active Inference model parameters, e.g., beliefs about neighbors' opinions likelihood to change as well) to experiment and generate behavior/data
- Self-evidencing agents exhibit confirmation-bias dynamics as they attend more to the tweets of like-minded neighbors, reinforcing belief polarization and echo-chamber formations at the macro-level
- Find that after some threshold in their inferences, agents exhibit more fixity in their beliefs and behavior

# Active Inference: Learning Resources

[Active Inference: The Free Energy Principle in Mind, Brain, and Behavior](#)

(Parr et. al, 2022).

- **Canonical open-source textbook** for Active Inference, from theory to model construction and analysis and inter-field comparison

“A step-by-step tutorial on active inference and its application to empirical data” (Smith et. al, 2022).

- in-depth explanation of Active Inference, building POMDPs, fitting data to models, complemented with MATLAB code examples

*The Fundamentals of Active Inference* (Namjoshi, forthcoming).

- Forthcoming two-volume textbook on Active Inference, complemented with Python code examples

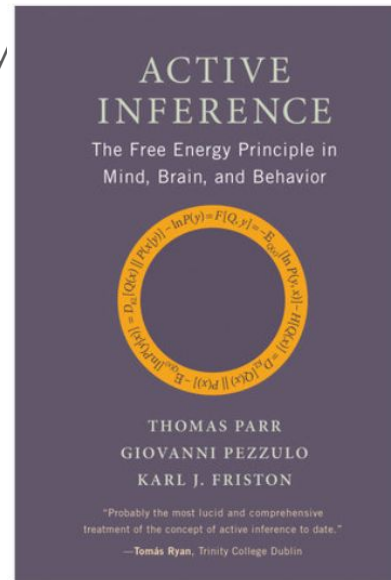
“**pymdp**: A Python library for active inference in discrete state spaces”

(Heins et. al, 2022) : <https://pymdp-rtd.readthedocs.io/en/latest/>

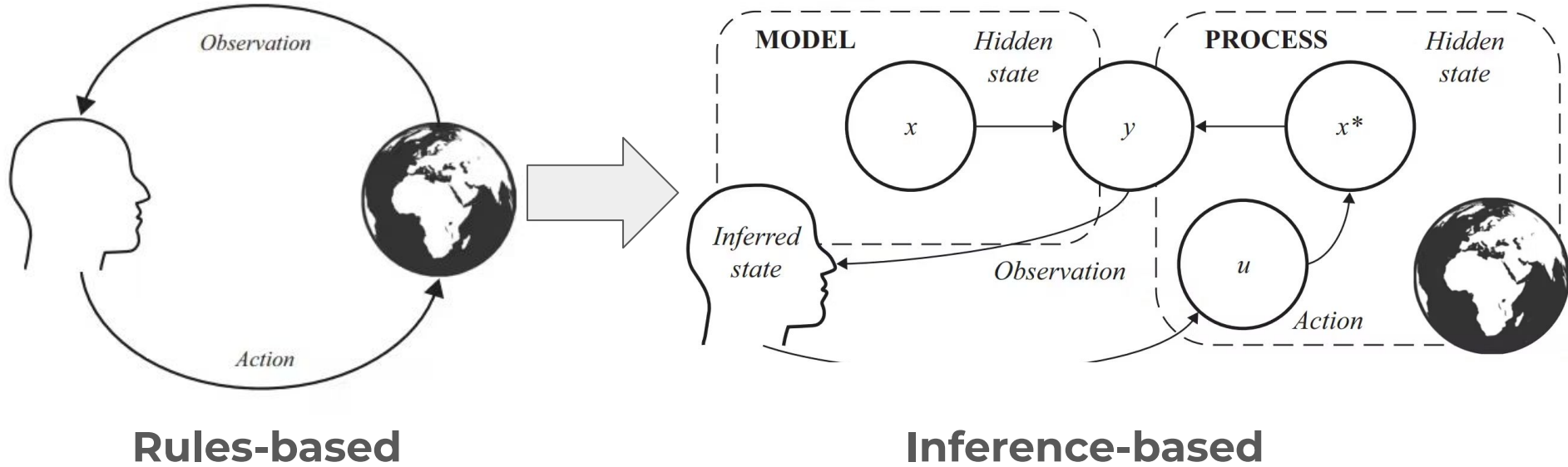
- Python package, incl. accessible Colab tutorial scripts, e.g., data structure fundamentals, free energy minimization, T-maze and multi-armed bandit examples

“Constructing cultural landscapes: Active Inference for the Social Sciences” (Active Inference Institute, 2023).

- Twelve-lecture series on applying Active Inference conceptual framework to, e.g., collective behavior, norm generation and diffusion, shared cultural reality



# From ABM to Active Inference: Inference-based action-selection



# Markov Blanket

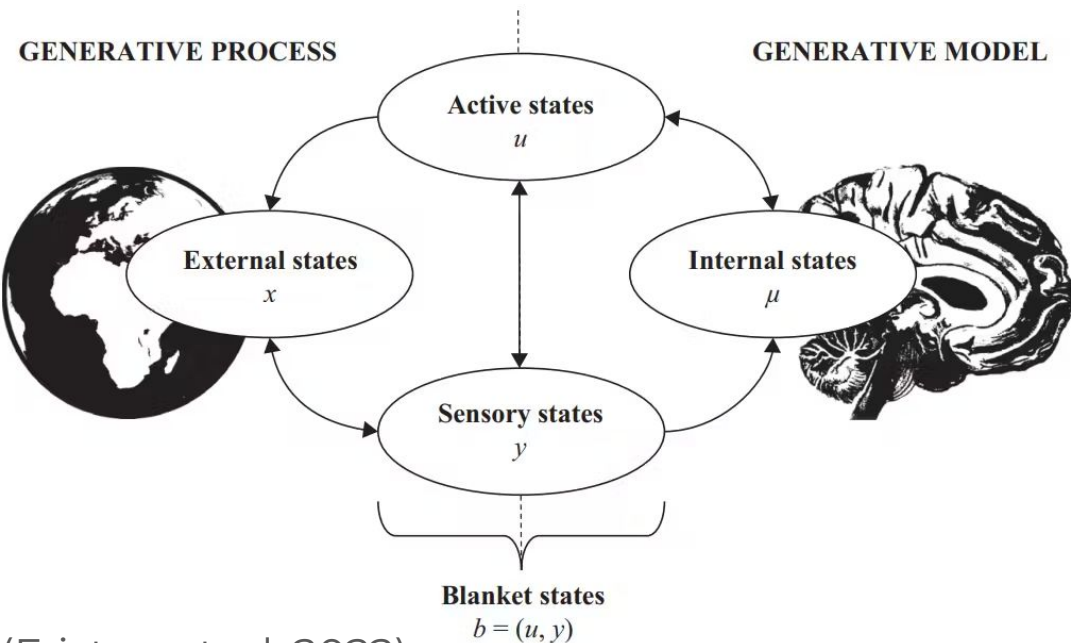
The agent-environment relationship can be seen as the relationship between a **generative model (agent)** and a **generative process (environ)**

These two entities come into contact via an action-perception loop as the agent makes inferences about its environment and acts based on FEP

Map is not the territory:  
An agent's internal states *might* come to mirror the environment's states, but there is no necessity of an identity  
"Knock on wood"...

Without separation between 'system' and 'not-system', there is no surprise: a system (agent) without homeostatic preferences (to maintain for survival) succumbs to its environment

Example: Markov chains with temporal dynamics do not require the past for the present to influence the future



(Friston et. al, 2022)

"An important precondition for any adaptive system is that it must enjoy some separation and autonomy from the environment—without which it would simply dissipate, dissolve, and thereby succumb to environmental dynamics."

# Inverse Problem:

Hidden states are the causes to be inferred

Observations/  
outcomes:

- **Interoception**  
(from within)
- **Exteroception**  
(from outside)
- *Proprioception*  
(position and movement in space)

Hidden states  
(prior beliefs), their  
relation to  
observations  
(likelihoods), and how  
to change them  
through action  
(transitions):

- Beliefs held/  
encoded in memory
- Planning ----

“Your brain receives sense data from the world, sights and sounds and smells and so on. And it receives sense data from the body. And so when there's a loud bang or a tug in your chest, your brain receives that information as the outcomes of some set of causes. But what caused that loud bang? ... What caused that tug in your chest? ... **Your brain actually doesn't know. It only receives the outcomes. It has to guess at what the causes are. This is called an inverse problem. You know what the outcome is, you don't know what the cause is.** And so your brain has one other source of information available to it. And that is past experience, memory. So it's not like you walk around consciously remembering things in order to perceive the world. Your brain is constantly remembering, meaning it is constantly reassembling bits and pieces of past experience in order to make sense of what's going on in the body, in relation to the world so that your brain knows what to do next.” (Feldman Barrett, 2021).



# Bayesian Inference and Biology

$$\text{Generative Model} = P(o, s)$$

$$\underbrace{P(s|o) = \frac{\overbrace{P(o|s)}^{\text{likelihood}} \overbrace{P(s)}^{\text{prior}}}{\underbrace{P(o)}_{\text{evidence}}}}_{\text{Bayes' Theorem}} \approx \underbrace{Q(s)}_{\text{approximate posterior}}$$

## Biological/metabolic constraints:

Approximate inference is needed to overcome the intractability of exact inference in most biologically plausible scenarios

## Bayesian Brain Hypothesis:

Perception is not a purely bottom-up passive process of inferring states from observations (sensory info).

Instead, **cognitive inference** involves combining **prior** knowledge about hidden states with incoming **observations** to compute a resultant **posterior** distribution over hidden states given the new observations. This posterior then can update the prior for the next step.

**Uncertainty and biases:** weak likelihoods can diminish impact of new evidence, strong likelihoods can overweight new evidence

# ABC(DEFG)'s of Inference

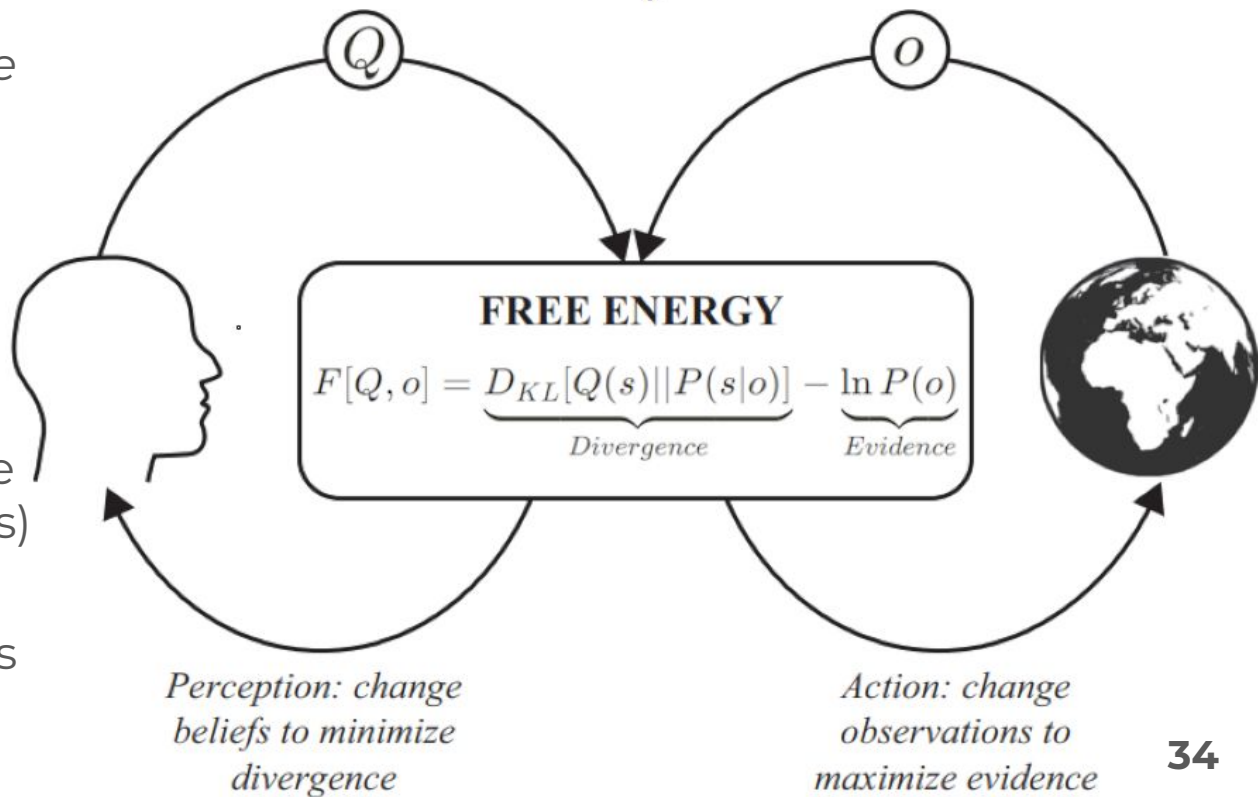
“F”: Variational free energy

$$F[Q, o] = \underbrace{D_{KL}[Q(s) || P(s|o)]}_{\text{Divergence}} - \underbrace{\ln P(o)}_{\text{Evidence}}$$

- recall that agent's are self-evidencing: they seek evidence for their generative model

- they seek to minimize surprise,  $-\ln P(o)$  (negative log model evidence), but this is often intractable (e.g., high number of states to be marginalized out; intractable integrals in continuous cases)

- Instead, free energy – an upper bound on surprisal – is minimized



# ABC(DEFG)'s of Inference

**"F": Variational free energy**  
- a functional of  $Q$  (agent's approximate beliefs) and observations (from the environment)

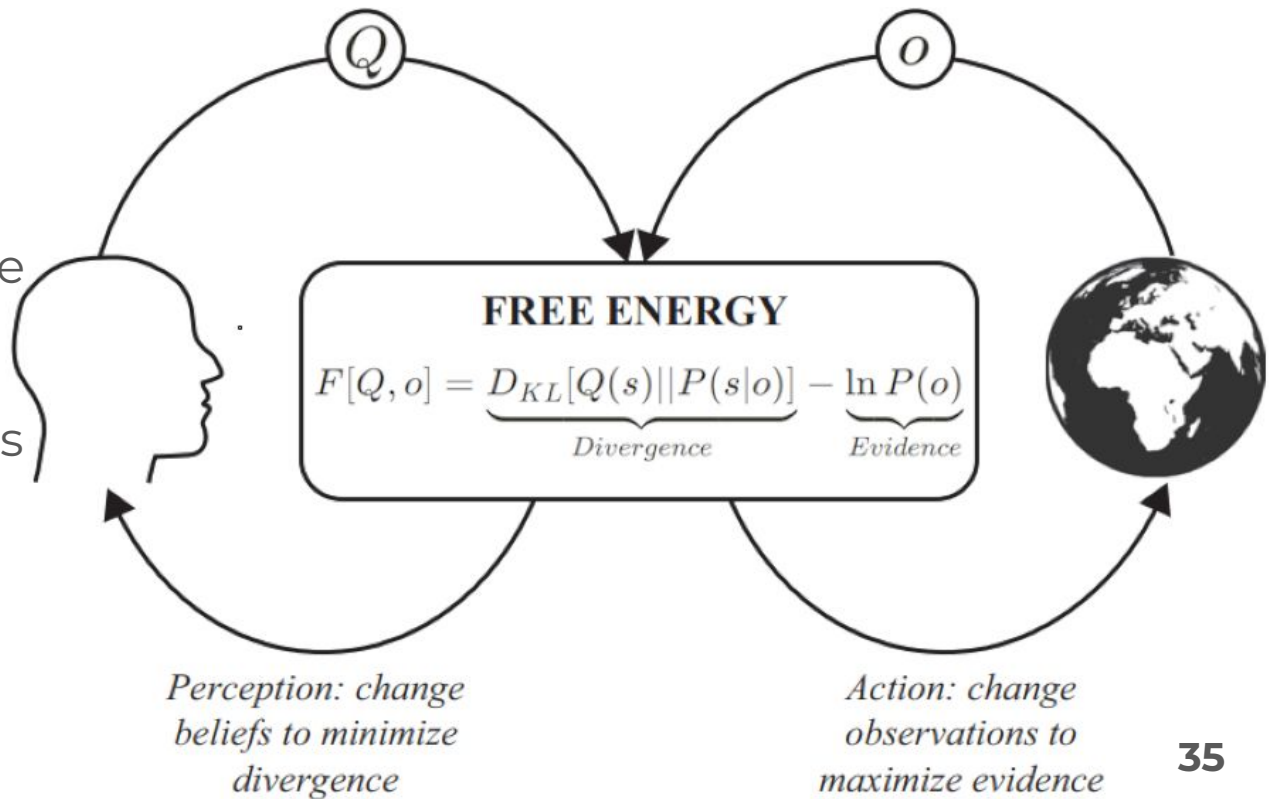
**- Divergence:**

Kullback-Leibler divergence between the inferred (approximate) distribution over states, and the model's posterior distribution over states

**- Evidence:**

Incoming observations ("ln" for easing mathematical operations involved)

$$F[Q, o] = \underbrace{D_{KL}[Q(s) || P(s|o)]}_{\text{Divergence}} - \underbrace{\ln P(o)}_{\text{Evidence}}$$



# Active Inference and the FEP, in brief

With Bayes' theorem, we can relate various variables (states, observations, actions), as probability distributions (agent beliefs)..

“Variational free energy” (VFE), and planning variant “expected free energy” (EFE), inherit from statistical physics, capturing distributional divergences and dissipation – **in Active Inference, this is applied to the belief states of agents.**

VFE has been used more generally in ML and study of open, nonequilibrium steady state systems, e.g., neuronal dynamics.

$$\underbrace{P(s|o) = \frac{\overbrace{P(o|s)}^{\text{likelihood}} \overbrace{P(s)}^{\text{prior}}}{\underbrace{P(o)}_{\text{evidence}}}}_{\text{Bayes' Theorem}} \approx \underbrace{Q(s)}_{\text{approximate posterior}}$$

(Friston et. al, 2022)

$$\begin{aligned} \underbrace{-\ln P(o|m)}_{\text{surprise}} &\leq \underbrace{F}_{\text{VFE}} \\ &= \underbrace{D_{KL}[Q(s)||P(s|o)]}_{\text{Divergence}} - \underbrace{\ln P(o)}_{\text{Evidence}} \\ &= \underbrace{-\mathbb{E}_{Q(s)}[\ln P(o, s)]}_{\text{Energy}} - \underbrace{H[Q(s)]}_{\text{Entropy}} \\ &= \underbrace{D_{KL}[Q(s)||P(s)]}_{\text{Complexity}} - \underbrace{\mathbb{E}_{Q(x)}[\ln P(o|s)]}_{\text{Accuracy}} \\ G(\pi) &= \underbrace{-\mathbb{E}_{Q(\tilde{s}, \tilde{o}|\pi)}[D_{KL}[Q(\tilde{s}|\tilde{o}, \pi)||Q(\tilde{s}|\pi)]]}_{\text{Information Gain}} - \underbrace{\mathbb{E}_{Q(\tilde{o}|\pi)}[\ln P(\tilde{o}|C)]}_{\text{Pragmatic value}} \\ &= \underbrace{\mathbb{E}_{Q(\tilde{s}|\pi)}[H[P(\tilde{o}|\tilde{s})]]}_{\text{Expected ambiguity}} + \underbrace{D_{KL}[Q(\tilde{o}|\pi)||P(\tilde{o}|C)]}_{\text{Risk (outcomes)}} \\ &\leq \underbrace{\mathbb{E}_{Q(\tilde{s}|\pi)}[H[P(\tilde{o}|\tilde{s})]]}_{\text{Expected ambiguity}} + \underbrace{D_{KL}[Q(\tilde{s}|\pi)||P(\tilde{s}|C)]}_{\text{Risk (states)}} \\ &= \underbrace{-\mathbb{E}_{Q(\tilde{s}, \tilde{o}|\pi)}[\ln P(\tilde{o}, \tilde{s}|C)]}_{\text{Expected energy}} - \underbrace{H[Q(\tilde{s}|\pi)]}_{\text{Entropy}} \end{aligned}$$

# Active Inference and the FEP, in brief

Agents continuously minimize VFE via perception/action in real-time, minimizing the divergence between their prior beliefs and their new posterior beliefs given new observations. VFE acts as a (tractable, variational) bound on (intractable) surprisal (negative log model evidence) via exploiting Jensen's Inequality. Expected free energy  $\ln(E[X]) \geq E[\ln(X)]$  capable of "planning," is for evaluating and selecting actions (pi) based on expected info gain and pragmatic value.

EFE uniquely involves agents' 'preferences' ('C' term) (*ibid*).

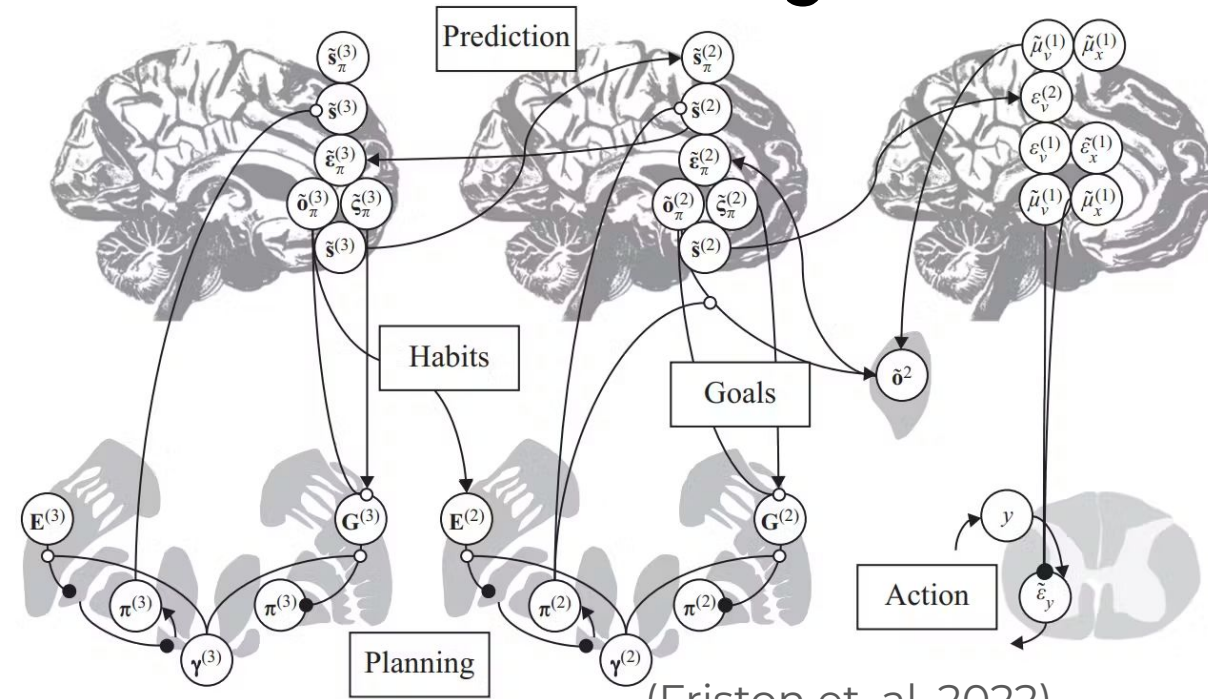
Computation is carried out via **gradient descent on free energy** as the variables/distributions are related and updated via **message passing schemes**, often visualized and scheduled on normal/Bayesian factor graphs.

(Friston et. al, 2022)

$$\begin{aligned}
 & \underbrace{-\ln P(o|m)}_{\text{surprise}} \leq \underbrace{F}_{\text{VFE}} \\
 & = \underbrace{D_{KL}[Q(s)||P(s|o)]}_{\text{Divergence}} - \underbrace{\ln P(o)}_{\text{Evidence}} \\
 & = \underbrace{-\mathbb{E}_{Q(s)}[\ln P(o, s)]}_{\text{Energy}} - \underbrace{H[Q(s)]}_{\text{Entropy}} \\
 & = \underbrace{D_{KL}[Q(s)||P(s)]}_{\text{Complexity}} - \underbrace{\mathbb{E}_{Q(x)}[\ln P(o|s)]}_{\text{Accuracy}} \\
 \\ 
 G(\pi) &= \underbrace{-\mathbb{E}_{Q(\tilde{s}, \tilde{o}|\pi)}[D_{KL}[Q(\tilde{s}|\tilde{o}, \pi)||Q(\tilde{s}|\pi)]]}_{\text{Information Gain}} - \underbrace{\mathbb{E}_{Q(\tilde{o}|\pi)}[\ln P(\tilde{o}|C)]}_{\text{Pragmatic value}} \\
 &= \underbrace{\mathbb{E}_{Q(\tilde{s}|\pi)}[H[P(\tilde{o}|\tilde{s})]]}_{\text{Expected ambiguity}} + \underbrace{D_{KL}[Q(\tilde{o}|\pi)||P(\tilde{o}|C)]}_{\text{Risk (outcomes)}} \\
 &\leq \underbrace{\mathbb{E}_{Q(\tilde{s}|\pi)}[H[P(\tilde{o}|\tilde{s})]]}_{\text{Expected ambiguity}} + \underbrace{D_{KL}[Q(\tilde{s}|\pi)||P(\tilde{s}|C)]}_{\text{Risk (states)}} \\
 &= \underbrace{-\mathbb{E}_{Q(\tilde{s}, \tilde{o}|\pi)}[\ln P(\tilde{o}, \tilde{s}|C)]}_{\text{Expected energy}} - \underbrace{H[Q(\tilde{s}|\pi)]}_{\text{Entropy}}
 \end{aligned}$$



# Anatomy of Inference



(Friston et. al, 2022)

Active Inference theorizes the relationships between neurobiological and computational architectures

Model variables/parameters map to various cognitive processes

Free energy / prediction error minimization is carried out via message passing schemes on factor graphs

This tutorial will focus on discrete-time POMDP models, but *message passing schemes for free energy minimization can in principle be applied to any probabilistic factor graph model*

# Modeling Inference: Variables in an Active Inference POMDP

**POMDP (partially-observable Markov decision-making processes)** are the exemplar

discrete-time models used in Active Inference, which entails the following key variables and characteristics:

- As before:

- observations - “o”
- states - “s”
- priors - “D”
- likelihoods - “A”

- Action

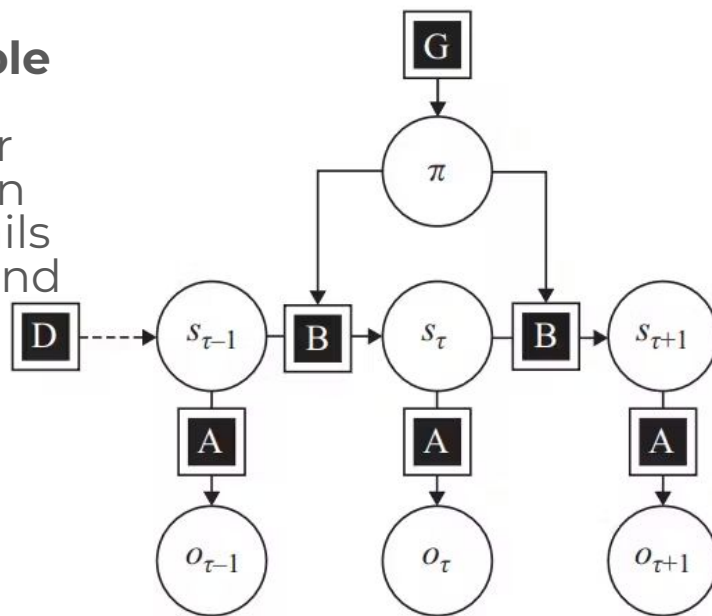
- habits - “E” (priors over policies)

- “policies” (sequences of actions; hypotheses about how to minimize free energy) - “pi”

- Preferences - “C” (priors over observations)

- Transitions of hidden states - “B”

- conditioned on actions/“policies” (pi), the *temporal dimension* of the model



**A**  $P(o_t^m | s_t^1, s_t^2, \dots, s_t^n, \dots)$

**B**  $P(s_{t+1}^n | s_t^n, \pi)$

**C**  $P(o_t^m)$

**D**  $P(s_1^n)$

**G**  $P(\pi | C, E)$

# From ABM to Active Inference: Inference-based action-selection

## Coming next:

### ABC(DEFG)'s of Inference

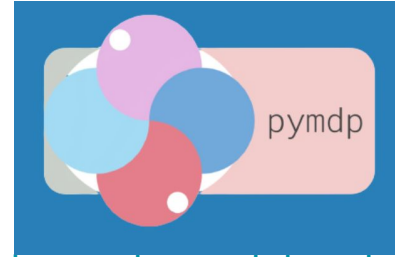
- defining components needed for constructing an ActInf POMDP agent
- minimizing free energy (F, G)
- policy inference
- “learning”

### Constructing a single agent

- plug our defined objects into the pymdp Agent() constructor
- these are the same agents used in the multi-agent ABM walkthrough in Part 3

### Single-agent inference example

- constructing a simple action-perception loop and storing/plotting results



<https://pymdp-rtd.readthedocs.io/en/latest/>

**Please access the Google Colab script in the tutorial github site if you have not done so!**

<https://github.com/apashea/IC2S2-Active-Inference-Tutorial>

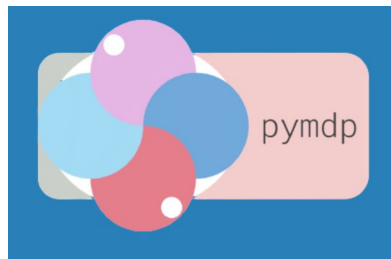


# From ABM to Active Inference: Inference-based action-selection

## Coming next:

### ABC(DEFG)'s of Inference

- defining components needed for constructing an ActInf POMDP agent
- minimizing free energy ( $F$ ,  $G$ )
- policy inference
- “learning”



<https://pymdp-rtd.readthedocs.io/en/latest/>

### Constructing a single agent

- plug our defined objects into the pymdp Agent() constructor
- these are the same agents used in the multi-agent ABM walkthrough in Part 3

### Single-agent inference example

- constructing a simple action-perception loop and storing/plotting results

```
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
import pymdp
from pymdp import utils
from pymdp.maths import softmax
from pymdp.agent import Agent
import copy
import math
import random
```

# Designing discrete-time Active Inference agents

POMDP (c.f. “Chapter 6” & “Chapter 7” in Friston et. al, 2022):

- **Defining states**: what are agents uncertain about?
- **Defining observations**: what do agents observe from their environment?
- **Defining controls/actions/policies**: what can agents do to interact with their environment?
  - **Actions**: discrete actions an agent can take at a given timestep
  - **Controls**: types of actions which are considered by the agent to actually change environmental states (e.g., rat can choose left/right arm in T-maze, but cannot impact reward location)
  - **Policies**: sequences of actions about which agent has beliefs, i.e. planning
- **Defining A, B, C, D, E**: probability distributions representing *agent-beliefs* relating states, observations, controls – i.e. *beliefs-based framework*
- **Defining our environment**: what takes in the agent’s actions and outputs observations?
- **Define the Active-Inference loop and run as simulation**: agent receives observations, infers states, learns, infers policies, commits an action
- **Comparison and revision** in relation to results and literature/domain knowledge (goal: reproduce empirical findings, e.g., of social dynamics, psychological or cognitive features, etc., then tune and explore counterfactuals)

# From ABM to Active Inference: Inference-based action-selection

## Agent we will construct:

Imagine an agent whose **task is to find a good solution to a complex task/problem**. They always 'prefer' (expect) to find better solutions to the problem than the solution they have currently. In the following single-agent simulation, the agent **is assumed to be in a network with one neighbor** (for demonstration, we only construct our agent).

**At each discrete timestep during an action-perception loop**, this agent...

**...can choose from two actions/controls/policies\***: to 'explore' new solutions alone, or to 'exploit' their neighbor (by stealing their answer)

**...infers an 'attention' hidden state**: who they infer who they are attending

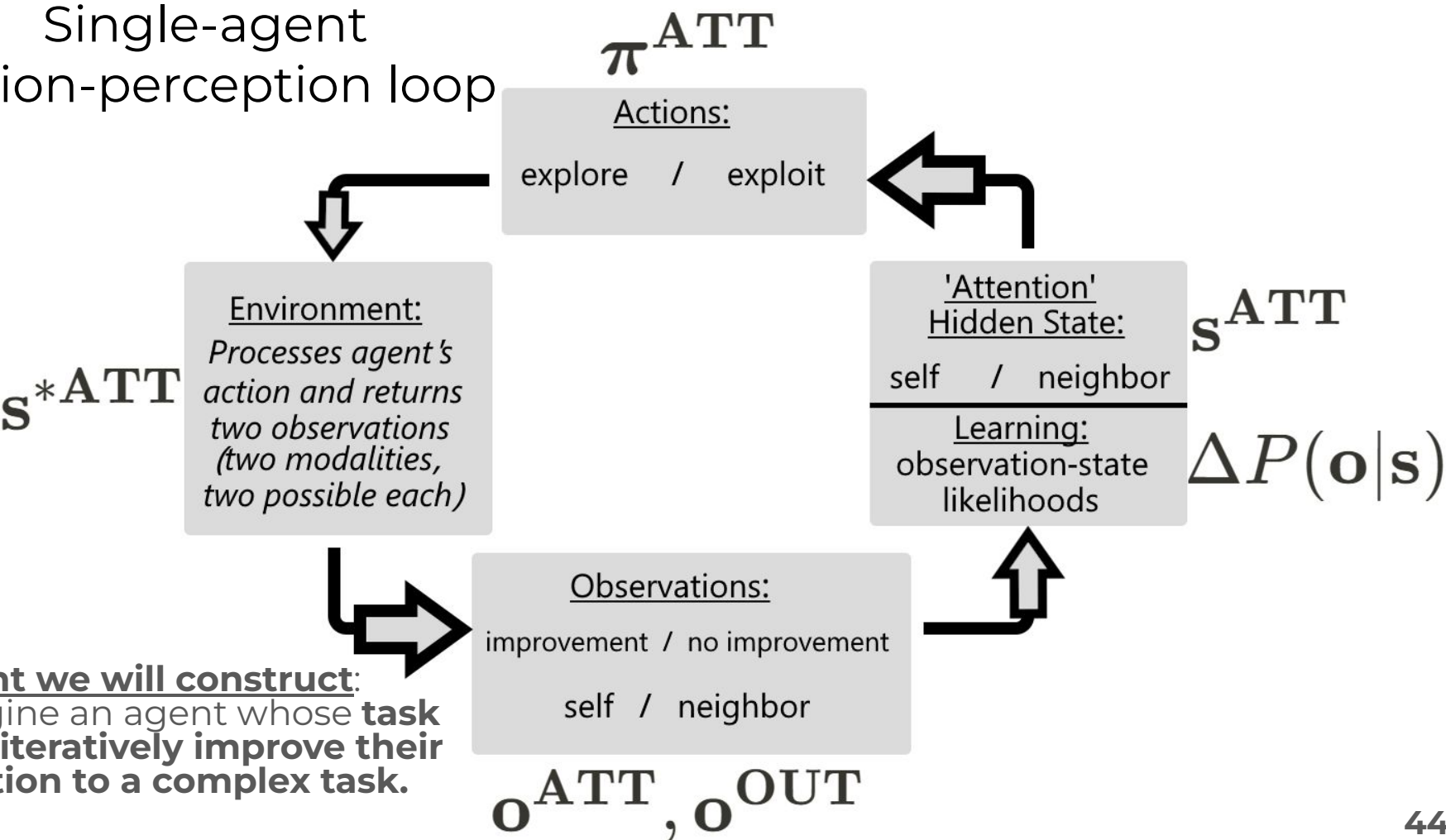
- 'explore' links to attending to 'self' with full certainty (probability 1.0)
- 'exploit' links to attending to 'neighbor' with full certainty (probability 1.0)
- Thus the agent will always know who it is attending to (i.e. fully informative transition model) based on its chosen action

**...receives two observations**: who they are attending, and whether or not they achieve an 'improvement' in their current solution or 'no improvement' in their current solution

**...learns** the relationship between observed improvement outcomes and the attention state of who are attending – i.e. they *learn* who they should attend to in order to see an 'improvement' in their solution.

*\*note: explore/exploit here are actions – they are not necessarily equivalent to the 'explore/exploit' dilemma. For goal-drive expected free energy minimization, each action can have its own exploratory or pragmatic value. The nomenclature used simply aligns with the names of the actions in the Lazer-Friedman model.*

# Single-agent Action-perception loop



**Agent we will construct:**  
Imagine an agent whose **task** is to iteratively improve their solution to a complex task.

# ABC(DEFG)'s of Inference

First define what we will be relating in the model:



Link as categorical distributions:

## (Hidden) States

- separated into N hidden state factors, each with their own discrete levels (discrete states)

$$\mathbf{s} = \{\mathbf{s}^1, \mathbf{s}^2, \dots, \mathbf{s}^N\}$$

## Observations/outcomes

- separated into M outcome modalities, each with their own discrete levels (discrete observations)

$$\mathbf{o} = \{\mathbf{o}^1, \mathbf{o}^2, \dots, \mathbf{o}^M\}$$

## Controls/actions/policies

- separated into F control factors, each with their own discrete levels (discrete actions)

$$\boldsymbol{\pi} = \{\boldsymbol{\pi}^1, \boldsymbol{\pi}^2, \dots, \boldsymbol{\pi}^F\}$$

"A matrix" : likelihood model

- one matrix per outcome modality

$$\mathbf{A}^m = P(\mathbf{o}_\tau^m | \mathbf{s}_\tau^1, \mathbf{s}_\tau^2, \dots, \mathbf{s}_\tau^N)$$

"B matrix" : transition model

- one matrix per hidden state factor

$$\mathbf{B}^n = P(\mathbf{s}_{\tau+1}^n | \mathbf{s}_\tau^n, \boldsymbol{\pi})$$

"C matrix" : preferences

- one vector per outcome modality

$$\mathbf{C}^m = P(\mathbf{o}_\tau^m)$$

"D matrix" : priors over initial states

- one vector per hidden state factor

$$\mathbf{D}^n = P(\mathbf{s}_{\tau=1}^n)$$

"E matrix" : habits

- one vector per control factor

$$\mathbf{E}^f = P(\boldsymbol{\pi}_{\tau=1}^f)$$

# ABC(DEFG)'s of Inference

## Example: An agent in a two-agent network (self and one neighbor)

- “Attention” (ATT) hidden state factor with 2 discrete states
  - “Outcome” (OUT) observation modality with 3 discrete observations
  - “Attention” (ATT) observation modality with 2 discrete observations
  - “Attention” (ATT) control factor with 2 discrete controls (actions)
- Our agent infers if attending to self or neighbor, observes improvement or not + whom they are attending, and can ‘explore’ or ‘exploit\_neighbor1’

***First we set discrete names for each and store their lengths***

$$s^{n_1} = s^{\text{ATT}}$$

```
attention_state_names = ['self', 'neighbor1']  
num_states = [len(attention_state_names)] # [2]  
num_factors = len(num_states) # 1
```

$$o^{m_1} = o^{\text{OUT}}$$

```
outcome_obs_names = ['improvement', 'no improvement', 'unobserved']
```

$$o^{m_2} = o^{\text{ATT}}$$

```
attention_obs_names = ['self', 'neighbor1']  
num_obs = [len(outcome_obs_names), len(attention_obs_names)] # [3,2]  
num_modalities = len(num_obs) # 2
```

$$\pi^{f_1} = \pi^{\text{ATT}}$$

```
action_names = ['explore', 'exploit_neighbor1']  
num_controls = len(action_names) # 2
```



# ABC(DEFG)'s of Inference

**“A matrix”**: likelihood model describing agent's beliefs regarding the conditional probability of an **outcome given the hidden state(s)**

We need **one subarray A[i] per outcome modality**  $A^m = P(o_\tau^m | s_\tau^1, s_\tau^2, \dots, s_\tau^N)$

|                                    |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                       |
|------------------------------------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| $A^{\text{OUT}}$                   | <pre># Initialize A matrix, with array per modality A = utils.obj_array(num_modalities) # Initialize A matrix for 'outcome' modality as zeros A[0] = np.zeros( ( len(outcome_obs_names), len(attention_state_names) ) ) # For each attention state level ('self', 'neighbor1') for attention_state in range(len(attention_state_names)):     # set as uniform distribution and normalize     A[0][:,attention_state] = utils.norm_dist(np.ones(shape=len(outcome_obs_names)))</pre>   |
| $P(o^{\text{OUT}} s^{\text{ATT}})$ |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                       |
| $A^{\text{ATT}}$                   | <pre># Initialize A matrix for 'attention' modality as zeros A[1] = np.zeros( ( len(attention_obs_names) , len(attention_state_names) ) ) # For each attention state level ('self','neighbor1') set identity matrix # i.e. no ambiguity about to whom agent is attending, then ensure normalized for attention_state in range(len(attention_state_names)):     A[1][attention_state,attention_state] = 1     A[1][:,attention_state] = utils.norm_dist(A[1][:,attention_state])</pre> |
| $P(o^{\text{ATT}} s^{\text{ATT}})$ |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                       |

**Array structure:** B[i][ 'observation' , 'state factor 1' , 'state factor 2' , ... ]

# ABC(DEFG)'s of Inference

**“A matrix”**: likelihood model describing agent’s beliefs regarding the conditional probability of an **outcome given the hidden state(s)**

$$\mathbf{A}^m = P(\mathbf{o}_\tau^m | \mathbf{s}_\tau^1, \mathbf{s}_\tau^2, \dots, \mathbf{s}_\tau^N)$$

```
print(utils.is_normalized(A[0]))
print(A[0])
```

```
True
[[0.33333333 0.33333333]
 [0.33333333 0.33333333]
 [0.33333333 0.33333333]]
```



$\mathbf{A}^{\text{OUT}}$

$\mathbf{o}^{\text{OUT}} = \text{improvement}$   
 $\mathbf{o}^{\text{OUT}} = \text{no improvement}$   
 $\mathbf{o}^{\text{OUT}} = \text{unobserved}$

$P(\mathbf{o}^{\text{OUT}} | \mathbf{s}^{\text{ATT}})$

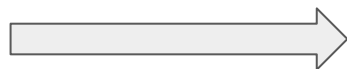
$\mathbf{s}^{\text{ATT}} = \text{self}$

$\mathbf{s}^{\text{ATT}} = \text{neighbor1}$

$$\begin{bmatrix} 0.33\bar{3} & 0.33\bar{3} \\ 0.33\bar{3} & 0.33\bar{3} \\ 0.33\bar{3} & 0.33\bar{3} \end{bmatrix}$$

```
print(utils.is_normalized(A[1]))
print(A[1])
```

```
True
[[1. 0.]
 [0. 1.]]
```



$\mathbf{A}^{\text{ATT}}$

$\mathbf{o}^{\text{ATT}} = \text{self}$   
 $\mathbf{o}^{\text{ATT}} = \text{neighbor1}$

$P(\mathbf{o}^{\text{ATT}} | \mathbf{s}^{\text{ATT}})$

$\mathbf{s}^{\text{ATT}} = \text{self}$

$\mathbf{s}^{\text{ATT}} = \text{neighbor1}$

$$\begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}$$



# ABC(DEFG)'s of Inference

**“B matrix”**: transition model describing agent's beliefs regarding the conditional probability of the hidden state at the next time step, given the current hidden state and an action performed

$$\mathbf{B}^n = P(\mathbf{s}_{\tau+1}^n | \mathbf{s}_{\tau}^n, \boldsymbol{\pi})$$

Ex. “If I explore, the state will transition from self or neighbor to self”

**We need one subarray B[i] per state factor, here just one**

$\mathbf{B}^{\text{ATT}}$

$$P(\mathbf{s}_{\tau+1}^{\text{ATT}} | \mathbf{s}_{\tau}^{\text{ATT}}, \boldsymbol{\pi}_{\tau}^{\text{ATT}})$$

```
# Initialize B matrix, with array per state factor
B = utils.obj_array(num_factors)
# Initialize B matrix for 'attention' state factor as zeros
B[0] = np.zeros( ( len(attention_state_names) ,
                  len(attention_state_names),len(action_names) ) )
# For each action ('explore', 'exploit_neighbor1')...
# ...set precise beliefs about attention state change
for action_i in range(len(action_names)):
    B[0][:,:,action_i][action_i] = 1
```

**Array structure:** B[i][ 'state at t+1' , 'state at t' , 'action' ]

# ABC(DEFG)'s of Inference

**“B matrix”**: transition model describing agent’s beliefs regarding the conditional probability of the hidden state at the next time step, given the current hidden state and an action

$$\mathbf{B}^n = P(\mathbf{s}_{\tau+1}^n | \mathbf{s}_{\tau}^n, \boldsymbol{\pi})$$

```
print(utils.is_normalized(B[0]))
print(B[0])
```

True

```
[[[1. 0.]
  [1. 0.]]
```

$$\mathbf{s}_{\tau+1}^{\text{ATT}=\text{self}} \begin{cases} \mathbf{s}_{\tau}^{\text{ATT}=\text{self}} \\ \mathbf{s}_{\tau}^{\text{ATT}=\text{neighbor1}} \end{cases}$$

```
[[[0. 1.]
  [0. 1.]]]
```

$$\mathbf{s}_{\tau+1}^{\text{ATT}=\text{neighbor1}} \begin{cases} \mathbf{s}_{\tau}^{\text{ATT}=\text{self}} \\ \mathbf{s}_{\tau}^{\text{ATT}=\text{neighbor1}} \end{cases}$$

$$\mathbf{B}^{\text{ATT}} \quad P(\mathbf{s}_{\tau+1}^{\text{ATT}} | \mathbf{s}_{\tau}^{\text{ATT}}, \boldsymbol{\pi}_{\tau}^{\text{ATT}})$$

$\boldsymbol{\pi}^{\text{ATT}=\text{explore}}$

$\boldsymbol{\pi}^{\text{ATT}=\text{exploit neighbor1}}$

$$\begin{bmatrix} 1 \\ 1 \end{bmatrix} \quad \begin{bmatrix} 0 \\ 0 \end{bmatrix}$$

$\boldsymbol{\pi}^{\text{ATT}=\text{explore}}$

$\boldsymbol{\pi}^{\text{ATT}=\text{exploit neighbor1}}$

$$\begin{bmatrix} 0 \\ 0 \end{bmatrix} \quad \begin{bmatrix} 1 \\ 1 \end{bmatrix}$$

50

# ABC(DEFG)'s of Inference

**“C matrix”**: preferences (priors over observations) denoting the outcomes the agent prefers (or not) and will thus seek out (or avoid) in a goal-directed manner. Since these are probability distributions, agents are seeking the outcomes they already expect most, i.e. “self-evidencing”.  $\mathbf{C}^m = P(\mathbf{o}_\tau^m)$

- Here for the ‘outcome’ modality, the agent strongly prefers ‘improvement’, disprefers ‘no improvement’, and is indifferent to ‘unobserved’. We then softmax the distribution to sum to 1 with non-negative values.

**We need one subvector  $\mathbf{C}[i]$  per outcome modality**

```
# Initialize C matrix
C = utils.obj_array_zeros(num_obs)

COUT
P(oτOUT) | # Initialize C matrix for 'outcome' modality
           | C[0] = utils.norm_dist(softmax(np.array([5, -1, 0])))

CATT
P(oτATT) | # Initialize C matrix for 'attention' modality
           | C[1] = utils.norm_dist(np.ones(shape=len(attention_obs_names)))
```

# ABC(DEFG)'s of Inference

**“C matrix”**: preferences (priors over observations) denoting the outcomes the agent prefers (or not) and will thus seek out (or avoid) in a goal-directed manner. Since these are probability distributions, agents are seeking the outcomes they already expect most, i.e. “self-evidencing”.  $\mathbf{C}^m = P(\mathbf{o}_\tau^m)$

```
print(utils.is_normalized(C[0]))
print(C[0])
```

$\mathbf{C}^{\text{OUT}}$

$P(\mathbf{o}_\tau^{\text{OUT}})$

True

```
[0.99086747 0.00245611 0.00667641]*
```



$\mathbf{o}^{\text{OUT}} = \text{improvement}$

$\mathbf{o}^{\text{OUT}} = \text{no improvement}$

$\mathbf{o}^{\text{OUT}} = \text{unobserved}$

*\*Note the softmax amplification*

$\sigma^*[5, -1, 0]$

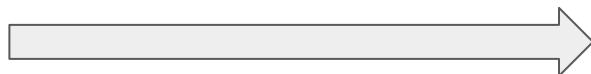
```
print(utils.is_normalized(C[1]))
print(C[1])
```

$\mathbf{C}^{\text{ATT}}$

$P(\mathbf{o}_\tau^{\text{ATT}})$

True

```
[0.5 0.5]
```



$\mathbf{o}^{\text{ATT}} = \text{self}$

$\mathbf{o}^{\text{ATT}} = \text{neighbor1}$

$\sigma[0.5, 0.5]$

52

# ABC(DEFG)'s of Inference

**“D matrix”**: priors over initial states, the agent’s beliefs about the current hidden state at the start of the simulation ( $t=1$ )

- We will use a discrete uniform distribution, i.e. the agent is uncertain as to who it is initially attending to

$$\mathbf{D}^n = P(\mathbf{s}_{\tau=1}^n)$$

```
DATT # Initialize D matrix  
D = utils.obj_array(num_factors)  
# Initialize D matrix for 'attention' state factor  
 $P(\mathbf{s}_{\tau=1}^{\text{ATT}})$  D[0] = utils.norm_dist(np.ones(shape=num_states))
```

```
print(utils.is_normalized(D[0]))  
print(D[0])
```

**D**<sup>ATT</sup>

$$P(\mathbf{s}_{\tau=1}^{\text{ATT}})$$

True  
[0.5 0.5]



$\mathbf{s}^{\text{ATT}}=\text{self}$      $\mathbf{s}^{\text{ATT}}=\text{neighbor1}$   
[0.5,                      0.5]



# ABC(DEFG)'s of Inference

**“E matrix”**: ‘habits’, i.e. priors over initial actions/policies

- learned over time when repeatedly relied upon to minimize free energy, resulting in heuristics-like, context-free (‘belief-free’) actions to be used in high ambiguity scenarios. This risks keeping agent in local optima and/or committing suboptimal actions (Friston, 2016).  $\mathbf{E}^f = P(\boldsymbol{\pi}_{\tau=1}^f)$
- note: *pymdp* can automatically construct the E matrix based on how certain argument are defined; we define them explicitly here for clarity

$\mathbf{E}^{\text{ATT}}$

# Initialize E matrix

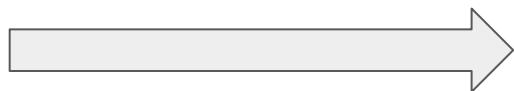
$P(\boldsymbol{\pi}_{\tau=1}^{\text{ATT}})$  | `E = utils.norm_dist(np.ones(shape=num_controls))`

```
print(utils.is_normalized(E))  
print(E)
```

$\mathbf{E}^{\text{ATT}}$

$P(\boldsymbol{\pi}_{\tau=1}^{\text{ATT}})$

True  
[0.5 0.5]



$\boldsymbol{\pi}^{\text{ATT}=\text{explore}}$

[0.5,

$\boldsymbol{\pi}^{\text{ATT}=\text{exploit neighbor1}}$

0.5]

# Learning

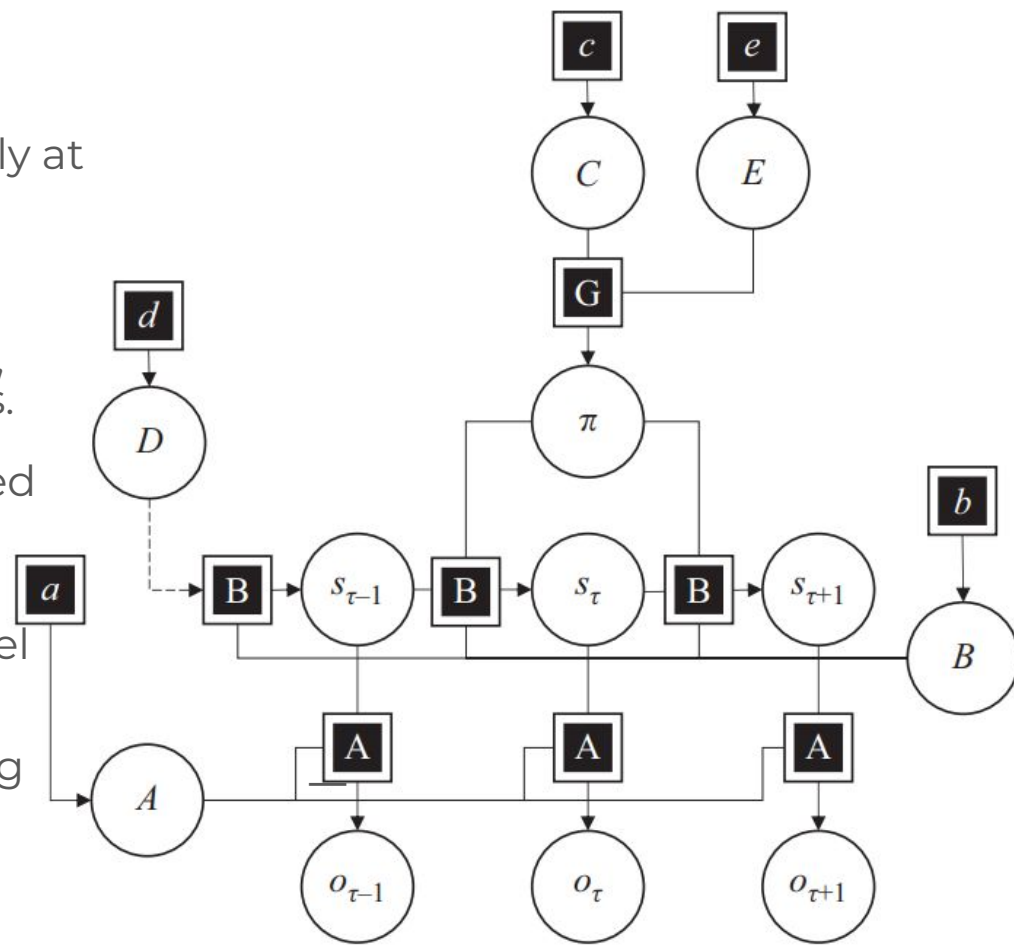
Agents learn– **update/modulate their internal variables/parameters** – typically at a **slower timescale** than real-time perceptual inference.

This may be coded to occur after every  $X$  timesteps, or at the end of an entire trial, depending on the use case/assumptions.

This timescale differentiation is supported by literature on learning, e.g., memory consolidation, LTM/STM, fast vs. slow neurotransmitters. This mediates the impact of new observations on the model (Friston et. al, 2016; Smith et. al, 2020).

Computationally, this occurs by assigning Dirichlet distributions (conjugate to categorical distributions) as **hyperparameters to our variables**.

These hyperparameters are thus **priors (beliefs about) the variables themselves**.



# Learning

Here our agent will **‘learn’ their likelihood model**, i.e. the probabilistic associations they have made between states and observations contained in the **A matrix**.

We will assign ‘pA’ (prior beliefs about A) to our agent, a Dirichlet distribution – ‘pseudo-counts’ – conjugate to our original A matrix of categorical distributions (notice same structure in code). This lets the agent compare posteriors about A to a prior about A for Bayesian belief updating.

The ‘scale’ argument will scale the priors before we assign them to the agent. This will up- or down-weight the impact of experience (how much learning will impact the agent’s beliefs) and relates to precision modulation and synaptic plasticity (*ibid*).

This can be related to the agent’s “confidence” in its beliefs about how outcomes map to states.

```
# priors over A for learning
```

```
pA = utils.dirichlet_like(A, scale = 1.0)
print(pA)
```

```
[array([[0.33333333, 0.33333333],
        [0.33333333, 0.33333333],
        [0.33333333, 0.33333333]]) array([[1., 0.],
        [0., 1.]])]
```

```
# priors over A for learning
```

```
pA = utils.dirichlet_like(A, scale = 0.5)
print(pA)
```

```
[array([[0.16666667, 0.16666667],
        [0.16666667, 0.16666667],
        [0.16666667, 0.16666667]]) array([[0.5, 0. ],
        [0. , 0.5]])]
```

$$\frac{\partial \mathbf{F}}{\partial \ln \mathbf{A}} = \mathbf{a} - a - \sum_{\tau=1}^T o_{\tau} \otimes \mathbf{s}_{\tau}$$

$$Q(\mathbf{A}^m) = P(\mathbf{A}^m) + \alpha_{\mathbf{A}} \left( \left( o_{\tau}^m \otimes Q(\mathbf{s}) \right) * \mathbf{A}^m \right)$$



# ABC(DEFG)'s of Inference

## Constructing our agent using the Agent constructor:

We simply plug in our defined variables into the call to Agent() which will house the agent's beliefs, parameters, memory, etc. as class attributes

- inference\_algo= 'MMP' : specifies that we will use 'marginal message passing' as the particular message passing scheme for belief updating and free energy minimization. This algorithm strikes a balance between variational message passing (found to produce overconfident posteriors) and belief propagation (computationally expensive) (Parr et. al, 2019).

- In a more recent (non-pip) pymdp update, among other various new features (jax back-end, model fitting developments), the 'sophisticated' argument has been added for allowing sophisticated inference, optimizing policy tree search (Friston et. al, 2021; and Hodson et. al, 2023). This would be useful in cases of more actions available and/or longer policy lengths, e.g., 4 actions, 5-step policies =  $4^5 = 1024$  policies to evaluate (!).

```
agent = Agent(A = A, B = B, C = C, D = D, E = E, pA = pA,  
             inference_algo = 'MMP', policy_len=1,  
             inference_horizon=2, sampling_mode = 'full',  
             action_selection = 'stochastic')
```

# ABC(DEFG)'s of Inference

## Constructing our agent using the Agent constructor:

- We simply plug in our defined variables into the call to Agent()
- Other arguments:
  - policy\_len=1: agents will only plan their actions one step into the future, as they have relatively simple one-step actions (explore or exploit)
  - inference\_horizon=2: the agent will try to infer states for both  $t$  and  $t+1$  (into the future), figuring into *expected free energy* for policy inference and evaluation
  - sampling\_mode = 'full': agent will select actions to take based on the full posterior over policies computed during policy inference
  - action\_selection = 'stochastic': agent will sample actions from their posterior beliefs over policies probabilistically (rather than 'deterministic' where action with highest posterior probability is necessarily chosen)

```
agent = Agent(A = A, B = B, C = C, D = D, E = E, pA = pA,  
              inference_algo = 'MMP', policy_len=1,  
              inference_horizon=2, sampling_mode = 'full',  
              action_selection = 'stochastic')
```

# ABC(DEFG)'s of Inference

With these variables in place and the agent now constructed, the agent can now iteratively:

- **infer states** based on incoming observations  
`agent.infer_states(obs)`
- **learn/update its model parameters** based on incoming observations  
`agent.update_A(obs)` ←----- *used in this tutorial*  
`agent.update_B(qs_prev)`  
`agent.update_D()`
- **infer actions/policies** to commit next  
`agent.infer_policies()`  
`agent.sample_action()`

```
agent = Agent(A = A, B = B, C = C, D = D, E = E, pA = pA,  
             inference_algo = 'MMP', policy_len=1,  
             inference_horizon=2, sampling_mode = 'full',  
             action_selection = 'stochastic')
```


# ABC(DEFG)'s of Inference

**Hidden State inference:** in pymdp, using Marginal Message Passing, variational free energy (VFE) is computed *per policy*.


- see also: [pymdp manual free energy calculation](#)

$$F(\pi, \tau) = \underbrace{D_{KL}[Q(s_\tau|\pi) || P(s_\tau|s_{\tau-1}, \pi)]}_{Complexity} - \underbrace{\mathbb{E}_Q[\ln P(o_\tau|s_\tau)]}_{Accuracy}$$

- **Receiving observations and inferring states:** in pymdp, observations appear as vectors (Python: lists) whose indices are observation modalities and values are observation levels


$$\mathbf{o}_\tau = [o_\tau^{\text{OUT=improvement}}, o_\tau^{\text{ATT=self}}]$$

```
obs = [0,0] # observes 'improvement' when attending 'self'  
qs = agent.infer_states(obs) # returns approx posterior Q(s)  
print(agent.F) # VFE over policies
```


$$\mathbf{F}_\pi = \begin{matrix} F_{\pi^{\text{ATT=explore}}} & F_{\pi^{\text{ATT=exploit}}} \\ \begin{bmatrix} -0.89\bar{6} & -0.89\bar{6} \end{bmatrix} \end{matrix}$$

# ABC(DEFG)'s of Inference

Policy inference/selection:  $\underbrace{Q(\pi)}_{\text{posterior beliefs over policies}} = \sigma\left(-\underbrace{\mathbf{G}_\pi \gamma}_{\text{EFE}} - \underbrace{\mathbf{F}_\pi}_{\text{VFE}} + \ln \underbrace{\mathbf{E}_\pi}_{\text{Habits}}\right)$

---


$$F(\pi, \tau) = \underbrace{D_{KL}[Q(s_\tau|\pi) || P(s_\tau|s_{\tau-1}, \pi)]}_{\text{Complexity}} - \underbrace{\mathbb{E}_Q[\ln P(o_\tau|s_\tau)]}_{\text{Accuracy}}$$

VFE is computed per timestep for each policy during state inference:

Complexity: the VFE value for a given policy will be greater if the agent's (approximate) belief about the current hidden state given the policy is far from the transition model's prediction of what the state should be given the policy  
 - our agent has perfect awareness of transitions of the relationship between policies and states (the B matrix is an identity matrix), thus complexity is zero

Accuracy: the VFE value for a given policy will be greater if the agent's likelihood model does not predict the current observation very well  
 - with zero complexity, and accuracy as a negative term, *our agent's VFE values per policy will always be equivalent to each other, negative, and oscillate* based on observations received versus the A matrix's ability to predict them

# ABC(DEFG)'s of Inference


## State inference & Learning:

- in the step using `agent.infer_states(obs)` we stored the ``qs`` it returned
- our example's agent will always have perfect awareness of the state it's in, so *in this particular case* the results are not particularly interesting...
- ...but being able to extract this information can be useful in use-cases involving uncertain states... **and in our case, for learning**

- **Q(s):** the resultant approximate posterior over states **using the 'MMP' algorithm** has the following complex structure:

- **qs[policy index][timestep index][hidden state factor]**

```
print(qs)  $P(s_{\tau}^{\text{ATT}=self} | \pi_{\tau}^{\text{ATT}=explore})$   $P(s_{\tau}^{\text{ATT}=neighbor1} | \pi_{\tau}^{\text{ATT}=explore})$   
[array([array([array([1.00000000e+00, 2.29826198e-15]), dtype=object), ←  $\pi_{\tau}^{\text{ATT}=explore}$   
      array([array([1.00000000e+00, 8.47203606e-16]), dtype=object]), ←  $\pi_{\tau+1}^{\text{ATT}=explore}$   
      dtype=object)  
array([array([array([1.00000000e+00, 2.29826198e-15]), dtype=object), ←  $\pi_{\tau}^{\text{ATT}=exploit}$   
      array([array([8.47203606e-16, 1.00000000e+00]), dtype=object]), ←  $\pi_{\tau+1}^{\text{ATT}=exploit}$   
      dtype=object)  
      ]
```

 = current beliefs (at t)



# ABC(DEFG)'s of Inference

## Policy inference/selection:

$$Q(\pi) = \sigma(-\mathbf{G}_\pi \gamma - \mathbf{F}_\pi + \ln \mathbf{E}_\pi)$$

- How agents update their beliefs about actions (and which to take)
- Some nomenclature:
  - “Action”: a particular action an agent can take
  - “Control”: an action an agent can take to actually impact (i.e. ‘control’) states, as represented in the B matrix
  - actions/controls often denoted “a” or “u”
  - “Policy”: a sequence of actions whose length depends on the agent’s policy length
  - policies often denoted by  $\pi$
  - Example: “explore at t, explore at t+1” is a two-step policy to explore twice in a row:
$$\pi^1_\tau = (a^\text{explore}_\tau, a^\text{explore}_{\tau+1})$$

Our example agents will only have policies of length one since the task is the same at each timestep (argument: “policy\_len = 1”)

# ABC(DEFG)'s of Inference

Policy inference/selection: 
$$\underbrace{Q(\pi)}_{\text{posterior beliefs over policies}} = \sigma\left(-\underbrace{\mathbf{G}_\pi}_{\text{EFE}}\gamma - \underbrace{\mathbf{F}_\pi}_{\text{VFE}} + \ln \underbrace{\mathbf{E}_\pi}_{\text{Habits}}\right)$$

- **“posterior beliefs over policies”**: categorical distribution over expected policies, determining policy selection
  - in our example, with action\_selection= “stochastic”, agent will probabilistically sample from this distribution
  - ex. if  $Q(\pi) = [0.7, 0.3]$ , agent will choose ‘explore’ with prob 0.7 and ‘exploit’ with prob 0.3
- **“VFE”**: Variational free energy (here, computed per policy)
- **“EFE”**: Expected free energy (here, computed per policy)
  - minimized in *planning*, taking into account *epistemic and pragmatic value of actions to be taken*
- **Gamma**: ‘action precision’ in range [0,1], relating to agent’s confidence in its action model; up-/down-weights influence of agent’s planning
- **“Habits”**: as seen previously ( $\pi$  subscript for uniformity of notation and reminder of similar structure to  $VFE_\pi$  and  $EFE_\pi$ )



# ABC(DEFG)'s of Inference

Policy inference/selection:  $\underbrace{Q(\pi)}_{\text{posterior beliefs over policies}} = \sigma\left(-\underbrace{\mathbf{G}_\pi \gamma}_{\text{EFE}} - \underbrace{\mathbf{F}_\pi}_{\text{VFE}} + \ln \underbrace{\mathbf{E}_\pi}_{\text{Habits}}\right)$

$$G(\pi) = \underbrace{-\mathbb{E}_{Q(\tilde{s}, \tilde{o}|\pi)}[D_{KL}[Q(\tilde{s}|\tilde{o}, \pi) || Q(\tilde{s}|\pi)]]}_{\text{Information Gain (Epistemic Value)}} - \underbrace{\mathbb{E}_{Q(\tilde{o}|\pi)}[\ln P(\tilde{o}|C)]}_{\text{Pragmatic value}}$$

EFE is computed per timestep for each policy during policy inference:

- Information gain: also referred to as epistemic value – the value derived from learning new information – the expected difference between agent’s posterior beliefs given new observations given a policy and agent’s prior beliefs given a policy. Thus unlike VFE’s emphasis on *minimizing divergence*, EFE tries to *maximize* this difference.

- Pragmatic value: the probability of new observations given their assumed prior probabilities (‘preferences’), weighted by their probability given the policy.

As with various other optimization problems, the logarithm is used for easing computation (sums rather than products).

$$\mathbb{E}_{Q(\tilde{o}|\pi)}[\ln P(\tilde{o}|C)] = \sum_{\tilde{o}} Q(\tilde{o}|\pi) \ln P(\tilde{o}|C) \quad \mathbf{65}$$

# ABC(DEFG)'s of Inference

Policy inference/selection: 
$$\underbrace{Q(\pi)}_{\text{posterior beliefs over policies}} = \sigma\left(-\underbrace{\mathbf{G}_\pi \gamma}_{\text{EFE}} - \underbrace{\mathbf{F}_\pi}_{\text{VFE}} + \ln \underbrace{\mathbf{E}_\pi}_{\text{Habits}}\right)$$

---


$$G(\pi) = \underbrace{\mathbb{E}_{Q(\tilde{s}|\pi)}[H[P(\tilde{o}|\tilde{s})]]}_{\text{Expected Ambiguity}} - \underbrace{D_{KL}[Q(\tilde{o}|\pi)||P(\tilde{o}|C)]}_{\text{Risk}}$$

EFE is computed per timestep for each policy during policy inference:

- Expected Ambiguity: the expected entropy, or ‘inaccuracy,’ of state-observation mappings, ex.  $P(o=\text{Heads}/s=\text{raining})$  will be maximally ambiguous as  $P(o=\text{Heads}/s=\text{sunny})$  would be equally likely (one-to-many observation-state mappings).

- Risk: as the agent accumulates observations and actions histories, this is the divergence between the probability of new data given the policy and given preferences, respectively. Figuratively, the difference between what the agent expects to happen given policies and what the agent prefers. Relatable to ‘expected complexity’, higher when there are one-to-many mappings between policies and outcomes, ex. stochastic one-armed bandits (‘slot machines’).

Relatable to ‘risk’ in economics or control theory

# ABC(DEFG)'s of Inference

Policy inference/selection:  $\underbrace{Q(\pi)}_{\text{posterior beliefs over policies}} = \sigma\left(-\underbrace{\mathbf{G}_\pi \gamma}_{\text{EFE}} - \underbrace{\mathbf{F}_\pi}_{\text{VFE}} + \ln \underbrace{\mathbf{E}_\pi}_{\text{Habits}}\right)$

$$G(\pi) = \underbrace{-\mathbb{E}_{Q(\tilde{s}, \tilde{o}|\pi)}[D_{KL}[Q(\tilde{s}|\tilde{o}, \pi) || Q(\tilde{s}|\pi)]]}_{\text{Information Gain (Epistemic Value)}} - \underbrace{\mathbb{E}_{Q(\tilde{o}|\pi)}[\ln P(\tilde{o}|C)]}_{\text{Pragmatic value}}$$

EFE is computed per timestep for each policy during policy inference:

- as the agent learned (updated its A matrix), it has evidence for 'explore'

```
q_pi, neg_efe = agent.infer_policies() # returns Q(pi) and negative G
print(agent.G*-1)
# stochastically sample actions
action_sampled = agent.sample_action() # returns vector of action(s) selected
action_sampled_id = int(action_sampled[0]) # extract value as integer
print(action_sampled_id)
```

$[3.15837839 \ 3.48714546]$   $\xrightarrow{\theta}$   $\pi_{\text{ATT=explore}}^{\text{ATT=exploit}}$   $\mathbf{G}_\pi = \begin{bmatrix} G_{\pi \text{ATT=explore}} & G_{\pi \text{ATT=exploit}} \\ 3.15\bar{8} & 3.48\bar{7} \end{bmatrix}$

# ABC(DEFG)'s of Inference

## Learning the A matrix:

- For **learning** we extract the posterior over states only for the current timestep given the policy chosen, **qs[policy chosen][current timestep]**
- The agent's original A matrix for the 'outcome' modality was perfectly uncertain. After one observation of 'improvement' + 'self' and running learning, the agent has up-weighted the probability of 'improvement' given state 'self' –

**it now associates the 'self' state with 'improvement' more**

```
action_sampled_id = 0 # (demo) hypothetical action chosen
# temp set agent's qs only to expected states at next timestep
# given action chosen for next timestep
agent.qs = copy.deepcopy(qs[action_sampled_id][0])
agent.update_A(obs) # run learning
agent.qs = copy.deepcopy(qs) # set qs back to previous
print(agent.A) # print the new likelihood model
[array([[0.66666667, 0.33333333],
        [0.16666667, 0.33333333],
        [0.16666667, 0.33333333]]) array([[1., 0.],
        [0., 1.]])]
```

*\*\*We will see shortly how the agent actually chooses actions*

**This agent needs learning to adapt**

$$\mathbf{A}^{\text{OUT}} = P(\mathbf{o}^{\text{OUT}} | \mathbf{s}^{\text{ATT}})$$

$$Q(\mathbf{A}^m) = P(\mathbf{A}^m) + \alpha_A \left( \left( o_\tau^m \otimes Q(\mathbf{s}) \right) * \mathbf{A}^m \right)$$



## Two-stage trial:

'Neighbor' (t=0 to t=14):

- initialize **obs** = **[0,1]**
- 'explore' returns 'no improvement' **[1,0]**
- 'exploit' returns 'improvement' **[0,1]**

'Self' (t=15 to t=29):

- reversal: **[0,0]** & **[1,1]**

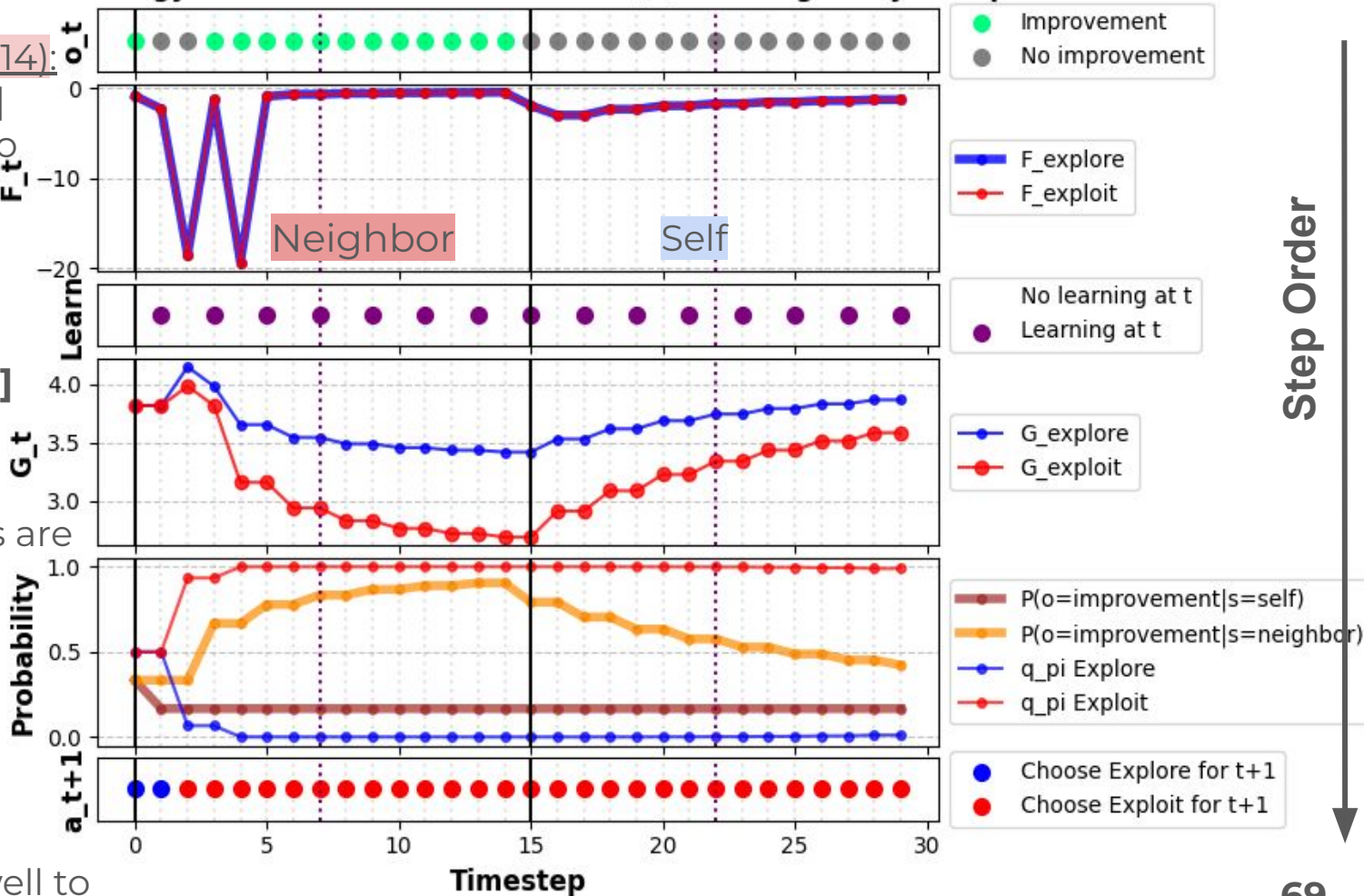
-  $F_{\pi}$  always equal:  
negative values  
denote both actions  
are informative

-  $G$  for policies  
differs based on  
expected  
epistemic vs.  
pragmatic value

- the agent adapts well to

Stage 1 in a few trials... **but must unlearn during Stage 2**

## Free Energy Minimization with Likelihood (A) Learning every 2 steps



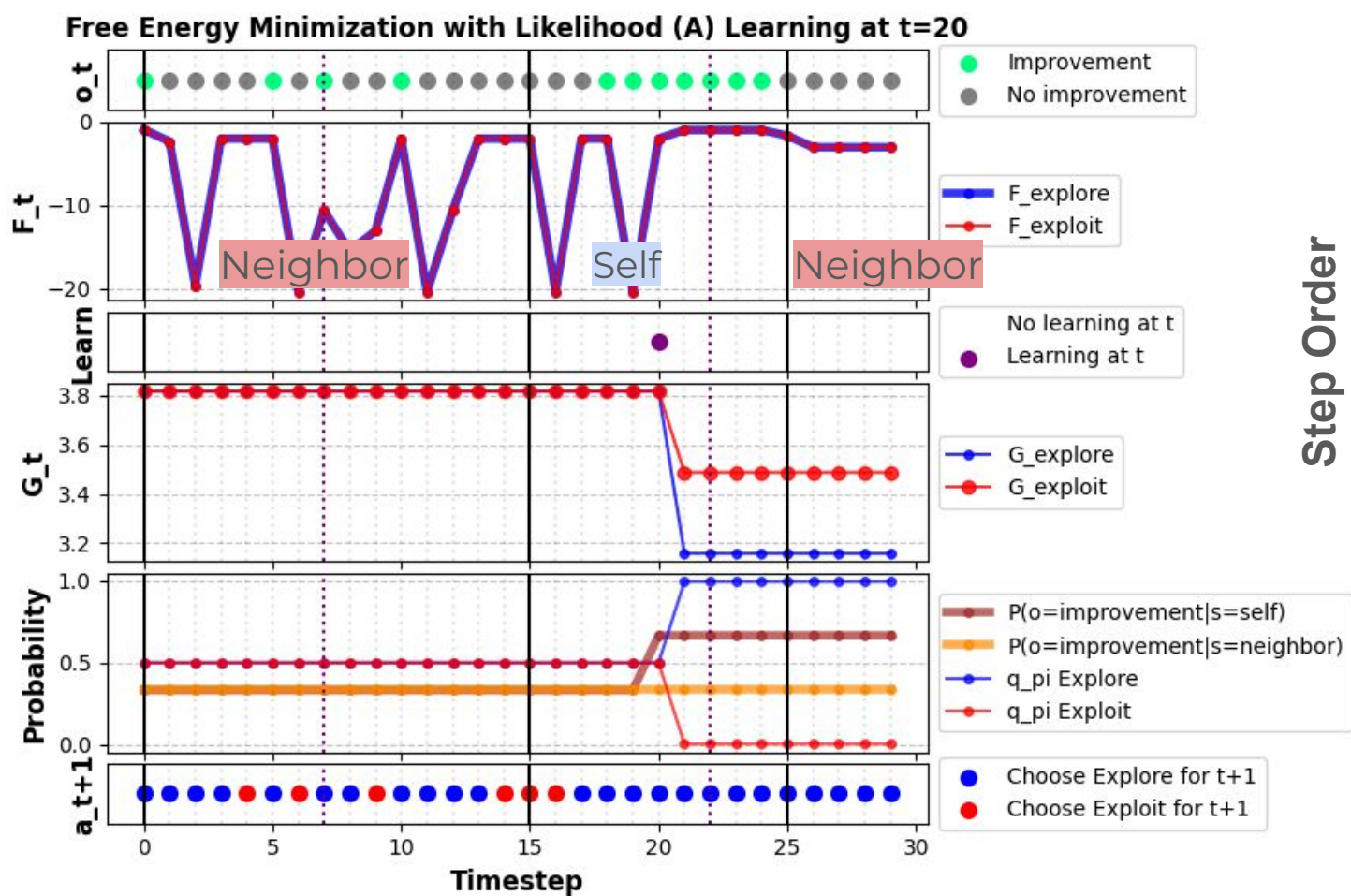
# Three-stage trial, one learning step:

- as before, except  
change back to  
'Neighbor' at t=25

- **Without learning:**  
there are no updates  
to the model's  
internal structure –  
only stochastic  
choices from current  
 $Q(\pi)$

- Agent only learns  
*once*, at t=20, which  
moves agent in  
'optimal' direction  
but no further  
changes

- **With only uniform and perfect distributions and no learning, nothing tips decision**  
**making away from stochastically choosing explore/exploit with 0.5 probability**



# Multi-Agent Application: Parallel Problem-Solving on an NK Landscape

## Coming next:

- We will plug these agents into a modified two-stage version of a paradigmatic ABM simulation for studying parallel problem-solving in networks

ABM: Agents in a network all attempt to find (explore) the best solution individually, but take (exploit) the solution of a network neighbor when one is available. This defines the deterministic behavior of agents per timestep.

- Review the paradigm ([Lazer et. al, 2007](#)) ([Netlogo code](#)), including its limitations for some policy-related use-cases
- In the tutorial Google Colab script:
  - We will look at creating X number of agents in networks and run simulations – random graphs with connectivity rate  $p$ , as well as identical subnetworks, ex. ‘5 groups of 4 agents’
  - Review the summary results for potential insights and value added from using Active Inference agents
    - the metrics collected in the original ABM
    - the cognitive metric “Personal Efficacy” from learning A matrix

# Multi-Agent Application: Parallel Problem-Solving on an NK Landscape

ABM: Agents in a network all attempt to find (explore) the best solution individually, but take (exploit) the solution of a network neighbor when one is available. This defines the deterministic behavior of agents per timestep.

This paradigm can be used to study polarization and consensus emergence involved in complex problem-solving as an exploration-exploitation scheme.

Research areas of interest:

- team performance
- social networks
- student group projects
- network dynamics and emergent behavior
- diversity of ideas

Potential parameters to modify:

- network connectivity  $0 \leq p \leq 1$
- complexity of problem to be solved (NK Landscape)
- errors in copying (mistakes when exploiting)



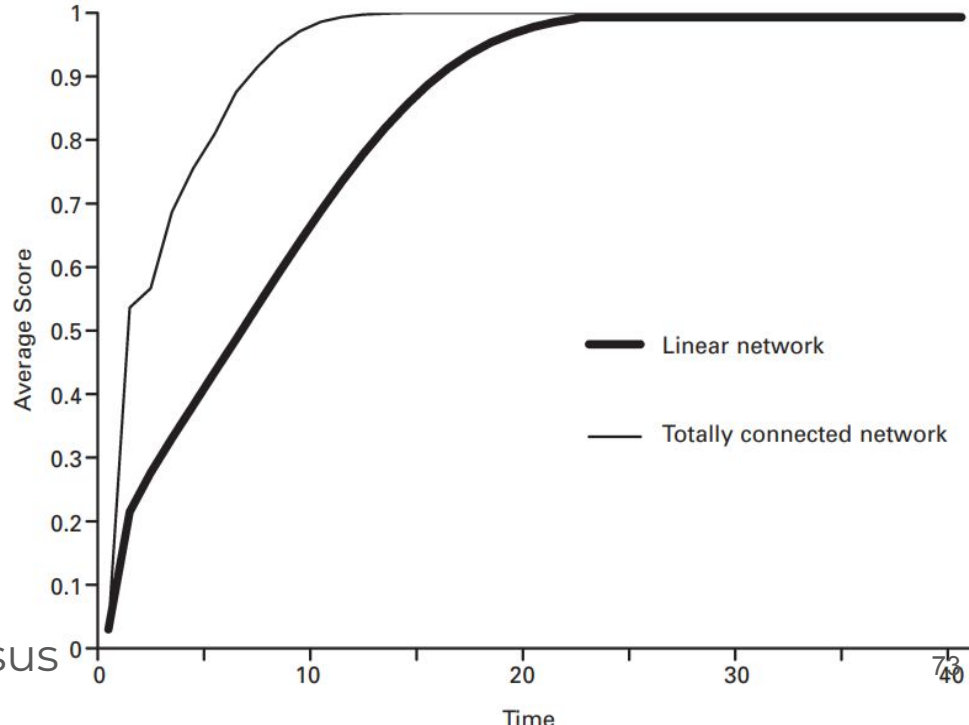
# Multi-Agent Application: Parallel Problem-Solving on an NK Landscape

ABM: Agents in a network all attempt to find (explore) the best solution individually, but take (exploit) the solution of a network neighbor when one is available.

Figure 8. Relative average performance for linear and totally connected networks solving a simple problem.

Lazer & Friedman (2007) found that:

- Densely connected networks produce fast consensus of agents around same 'best' solution – which diminishes need for further problem-solving
- Sparsely connected networks lead to polarization as clusters converge on distinct solutions – local consensus with global difference (potential local optima)
- Error-in-copying increases idea diversity and mitigates fast consensus

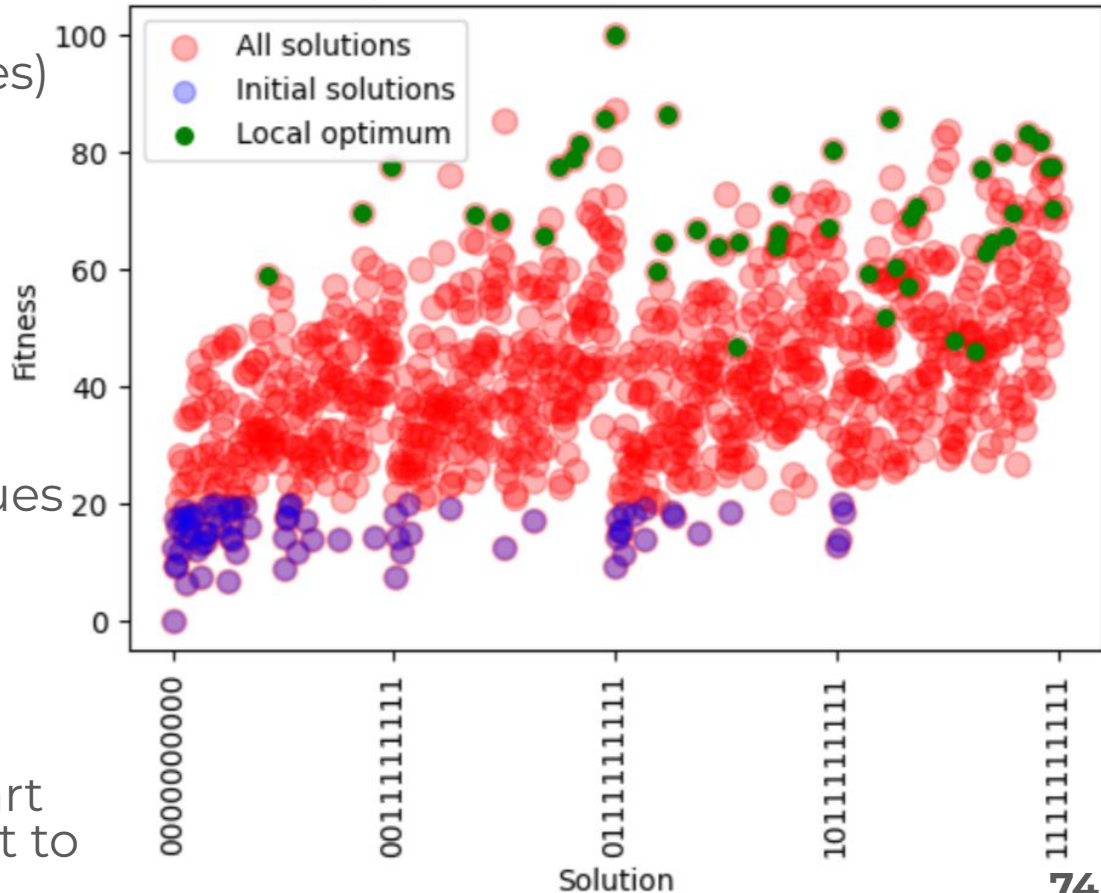


# NK Landscape (Levinthal, 1997)

NK Landscape with N=10, K=5: 1024 solutions total

Agents work towards the best solution (ranked by fitness values) in a fitness landscape:

- **N components**, each component taking one of B values (here, binary: B=2)  
ex.  $B^N = 2^{10} = 1024$  solutions
- **K interdependencies** between components
  - determines how fitness values are assigned to solutions
  - higher K -> more complexity (risk of local optimum)
  - ex. 'designing a car'
- Agents can be initialized to start with poor solutions and attempt to improve from there



# Lazer & Friedman Model (NetLogo)

Agents know the fitness levels of all of their neighbors' solutions.

Deterministically at each timestep they do the following:

- Exploit the best solution of their neighbors if it surpasses the agent's own solution  
*ex. if neighbor i's solution has the best fitness in the network, the agent will take neighbor i's solution*

- Explore new solutions if no neighbor has a better solution by 'flipping one bit':

example ( $N=10$ ,  $K>1$ ):

current solution at  $t$ :

- 0000000001 with fitness '62'

explored solution for  $t+1$ :

- 0000100000 with fitness '48'

NetLogo code:

```
to run-step
  ask turtles [
    ;; step 1: ask around. do any of my neighbors have better solutions?
    let best-solution solution
    let better-one? false
    ask link-neighbors [
      if evaluate-fitness solution > evaluate-fitness best-solution
      [
        set best-solution solution
        set better-one? true
      ]
    ]

    ;; step 2: if nobody had a better one, explore
    if not better-one?
    [
      let found-solution explore
      if evaluate-fitness found-solution > evaluate-fitness solution
      [
        set best-solution found-solution
        set better-one? true
      ]
    ]

    set solution best-solution
  ]
end
```

# Lazer & Friedman Model with ActInf Agents

Agents know the fitness levels of all of their neighbors' solutions.

Deterministically at each timestep they do the following:

- Exploit the best solution of their neighbors if it surpasses the agent's own solution  
*ex. if neighbor i's solution has the best fitness in the network, the agent will take neighbor i's solution*
- Explore new solutions if no neighbor has a better solution by 'flipping one bit':  
example ( $N=10, K>1$ ):  
current solution at  $t$ :
  - 00000000001 with fitness '62'explored solution for  $t+1$ :
  - 0000100000 with fitness '48'

Agents only knows the fitness level of a neighbor's solution if they 'attend' to that neighbor. Agent chooses, based on FEP, at each timestep to do the following:

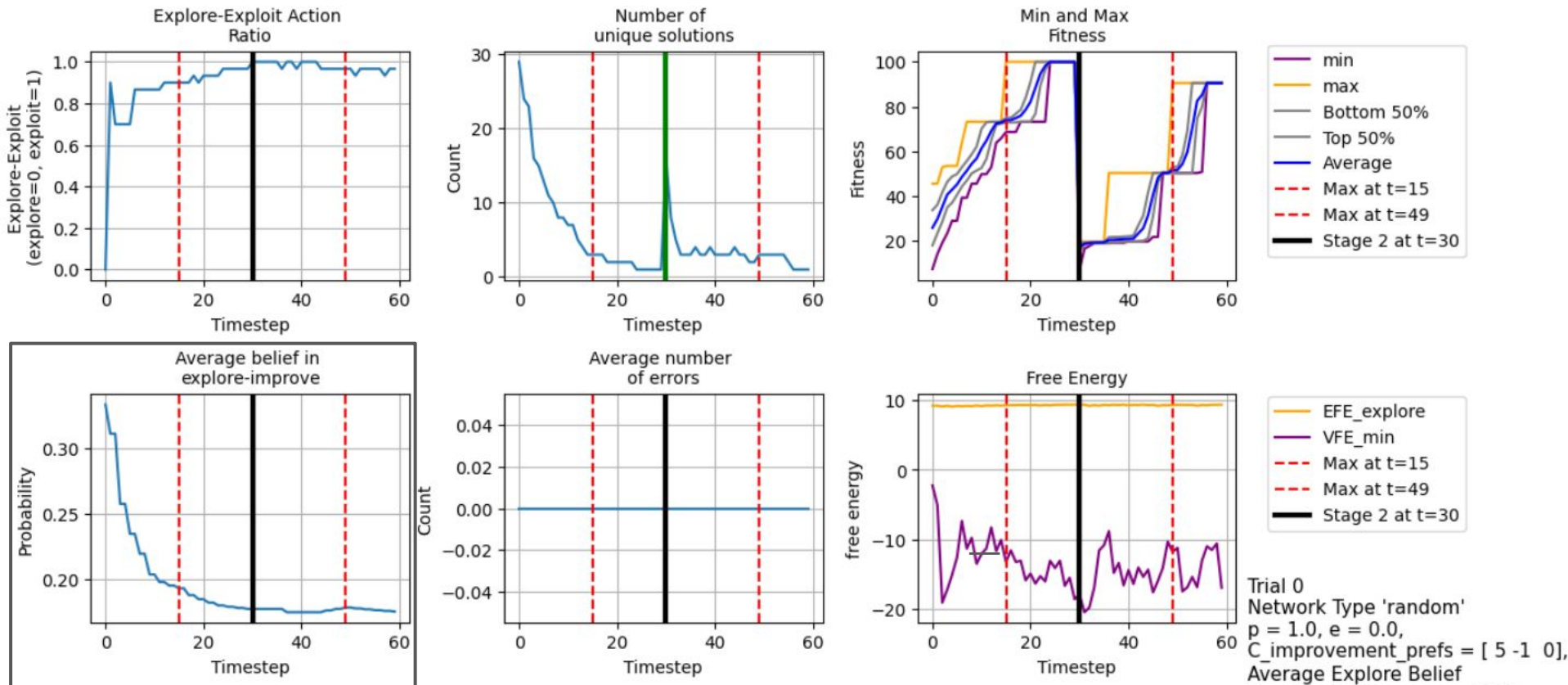
- Exploit one of their neighbors, revealing that neighbor's fitness  
*ex. if chosen neighbor's solution's fitness is higher, agent changes to it else keeps current fitness*
- Explore new solutions if agent so chooses (based on FEP):  
example ( $N=10, K>1$ ):  
current solution at  $t$ :
  - 00000000001 with fitness '62'explored solution for  $t+1$ :
  - 0000100000 with fitness '48'

# Lazer & Friedman Model $\Rightarrow$ with ActInf Agents

## Additional modifications/questions:

- **Consider the long-term impact of the task:** If agents actually *learn* from these experiences, what is the longevity of this performance? Shouldn't agents perform *better* on a second trial?
  - Change: **Two-stage trial** with a unique NK landscape for each and plot results
- What is the impact of this task on agent's beliefs, for example, their sense of 'Personal Efficacy'? Are agents better prepared for handling tasks individually and/or collectively going forward?
  - Change: **track  $P(o=improvement/s=self)$**  over time
- Say we have multiple groups – for example a *Problems-Based Learning* assignment for students – split into N groups of M agents?
  - Change: **additional 'subgroups' option** for making network into N separate subnetworks of M agents

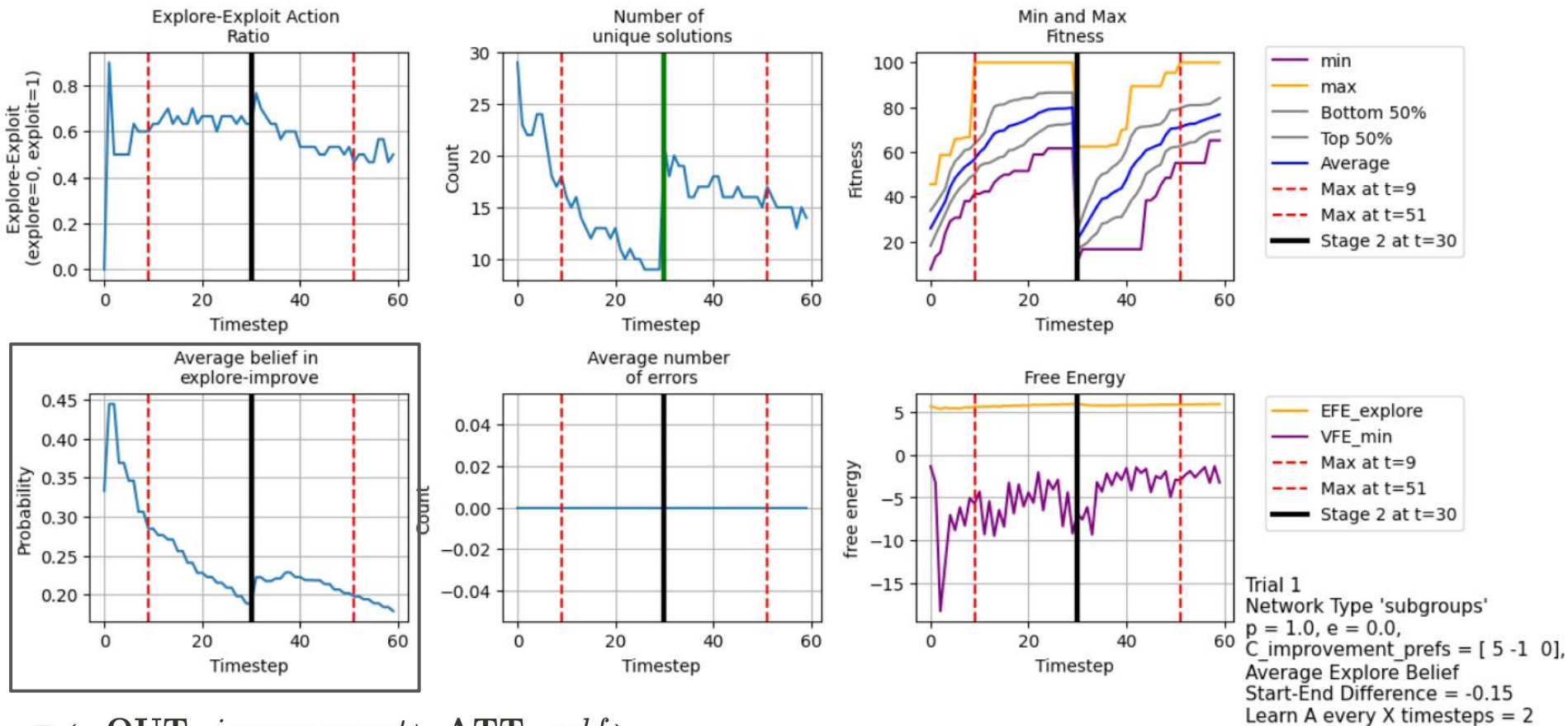
# 'Trial 0': random network, fully connected, 30 agents



Trial 0  
 Network Type 'random'  
 p = 1.0, e = 0.0,  
 C\_improvement\_prefs = [ 5 -1 0],  
 Average Explore Belief  
 Start-End Difference = -0.16  
 Learn A every X timesteps = 2

$$P(\mathbf{o}^{\text{OUT}} = \text{improvement} \mid \mathbf{s}^{\text{ATT}} = \text{self})$$

# 'Trial 1': 6 groups of 5 agents, 30 agents in total



$$P(o^{OUT=improvement} | s^{ATT=self})$$



# Other considerations

## **Recreate the original simulation in its entirety**

- what would happen if agents can actually see the best solution in their network (own or neighbor's) and have just two actions, explore and exploit?
  - 'realism' here: original simulation assumes all agents have full access to other agents' solutions (if no errors, then also fully 'understand' the solutions) at each  $t$
- Assuming uniform habits, would upweight both policies' priors (0.5, 0.5)

## **Revise hidden states, observations, actions, and/or learning**

- Only one hidden state for 'attention' was used with precise beliefs about transitions
- Adding an additional outcome state with uncertain beliefs (if explore then improvement, etc.), and learning the transitions, might change behavior (including VFE, as transitions with uncertainty will make the complexity term non-zero)
- pymdp supports learning A, B, and D matrices; possible to learn other parameters via manually-defined functions possible (see ActInf literature)
- hierarchical models (higher-order layers learn states to be supplied as priors for lower layers)

**Experiment with other factors in the literature**, e.g., error-in-copying (option is available in tutorial code), superiority bias (only keep exploited solution if its fitness is *substantially* better), etc. (Boroomand et. al, 2023).



# Thank you for your time!

- [andrewbpashea@gmail.com](mailto:andrewbpashea@gmail.com)
- <https://www.linkedin.com/in/andrewbpashea/>
- **pymdp** : <https://pymdp-rtd.readthedocs.io/en/latest/index.html>
- **Active Inference Institute** : <https://www.activeinference.org/>
  - Open weekly textbook group, collaborative learning
  - Projects of varying scales
  - RxInfer / BIASlab : <https://rxinfer.ml/>
    - working with the Engineering group at All exploring using the **julia** programming language to optimize computations in ActInf simulations
- Additional projects:
  - computational psychiatry project to study dynamics of belief updating in individuals clinically diagnosed with PTSD, modeling from qualitative data
  - Prisoner's Dilemma with Active Inference agents

# **Appendix**

# ABC(DEFG)'s of Inference

“F”: Variational free energy

- a functional of  $Q$  (agent beliefs) and observations environment)

We then compute VFE for

$$F[Q, o] = D_{KL}[Q(s) || P(s|o)] - \ln P(o)$$

Box 3.2

Free energy in statistical physics and Active Inference

Evidence

The notion of free energy is widely used in statistical physics to characterize (for example) thermodynamic systems. Although Active Inference uses exactly the same equations, it applies them to characterize the *belief state* of an agent (in relation to a generative model). Hence, when we talk of an Active Inference agent minimizing its (variational) free energy, we are referring to processes that change its belief state, not (for example) the particles of its body. To avoid misunderstandings, we use the term *variational free energy*, hence adopting a terminology that is more common in machine learning. Another more subtle point is that the concept of free energy is often used in the context of equilibrium statistical thermodynamics. Active Inference targets living organisms—or nonequilibrium steady state systems that are open—that feature continuous, reciprocal exchanges with the environment. This is an exciting novel field (Friston 2019a).

# ABC(DEFG)'s of Inference

**“F”**: Variational free energy

So far, we have discussed perception and action within a Bayesian scheme that aims to minimize surprise. Yet, exact Bayesian inference supporting perception and action is computationally intractable in most cases, because two quantities—the *model evidence* ( $P(y)$ ) and the *posterior probability* ( $P(x|y)$ )—cannot be computed for two possible reasons. The first is that for complex models, there may be many types of hidden states that all need marginalizing out, making the problem computationally intractable. The second is that the marginalization operation might require analytically intractable integrals. Active Inference appeals to a *variational* approximation of Bayesian inference that is tractable.

**“G”**: Expected free energy

consequences of possible courses of action include seeing a dead end after turning right or seeing the exit after a sequence of three left turns. Each possible sequence of actions is termed a *policy*. This highlights an important distinction made in Active Inference between an action and a policy. The former is something that directly influences the outside world, while the latter is a hypothesis about a way of behaving. The implication is that Active Inference treats planning and decision-making as a process of inferring what to do. This brings planning firmly into the realm of Bayesian inference and means we must specify priors and likelihoods as before (sec-

# ABC(DEFG)'s of Inference

**“G”**: Expected free energy

based on the premise that agents minimize the expected free energy of future outcomes. Crucially, the negative free energy or quality of a policy can be decomposed into extrinsic and epistemic (or intrinsic) value. Minimizing expected free energy is therefore equivalent to maximizing extrinsic value or expected utility (defined in terms of prior preferences or goals), while maximizing information gain or intrinsic value (or reducing uncertainty about the causes of valuable outcomes). The resulting scheme resolves the exploration-exploitation dilemma: Epistemic value is maximized until there is no further information gain, after which exploitation is assured through maximization of extrinsic value.

consequences of possible courses of action include seeing a dead end after turning right or seeing the exit after a sequence of three left turns. Each possible sequence of actions is termed a *policy*. This highlights an important distinction made in Active Inference between an action and a policy. The former is something that directly influences the outside world, while the latter is a hypothesis about a way of behaving. The implication is that Active Inference treats planning and decision-making as a process of inferring what to do. This brings planning firmly into the realm of Bayesian inference and means we must specify priors and likelihoods as before (sec-

# ABC(DEFG)'s of Inference

“F”: Variational free energy

Notice the similarity between ‘surprise’ and preferences (priors over outcomes)

## 3.4.1 Variational Free Energy, Model Evidence, and Surprise

A first important equivalence is between the *maximization of model evidence* (or marginal likelihood) in Bayesian inference and the *minimization of variational free energy*—both of which minimize surprise. This equivalence becomes evident when one appeals to a specific approximate solution to intractable problems of inference—variational inference. Variational inference recasts the inference problem as an optimization problem by minimizing free energy. The minimum of the free energy is the point at which the approximation of the exact solution is at its best. Expressing this formally sheds light on the relations between the three quantities:

$$\underbrace{\mathfrak{I}(y|m)}_{\text{Surprise}} = -\ln \underbrace{P(y|m)}_{\text{Model evidence}} \leq \underbrace{D_{\text{KL}}[Q(x) || P(x|y, m)] - \ln P(y|m)}_{\text{Variational free energy}} \quad (3.2)$$



# ABC(DEFG)'s of Inference

**F**: Variational free energy

Notice the similarity between ‘surprise’ and preferences (priors over outcomes)

**Table 3.1**

Statistical physics, Bayesian inference, and information theory—and their cognitive interpretations

| Statistical physics                                                  | Bayesian inference and information theory                                                  | Cognitive interpretation |
|----------------------------------------------------------------------|--------------------------------------------------------------------------------------------|--------------------------|
| Minimize variational free energy                                     | Maximize model evidence (or marginal likelihood); minimize surprisal (or self-information) | Perception and action    |
| Minimize expected free energy; Hamiltonian principle of least Action | Infer the most likely (or less surprising) course of action                                | Planning as inference    |
| Attain nonequilibrium steady-state                                   | Perform approximate Bayesian inference                                                     | Self-evidencing          |
| Gradient flows on energy functions; gradient descent on free energy  | Gradient ascent on model evidence; gradient descent on surprisal                           | Neuronal dynamics        |

# Single agent inference

Now we can feed the agent a single observation and infer the approximate posterior  $Q(s)$  via free energy minimization.

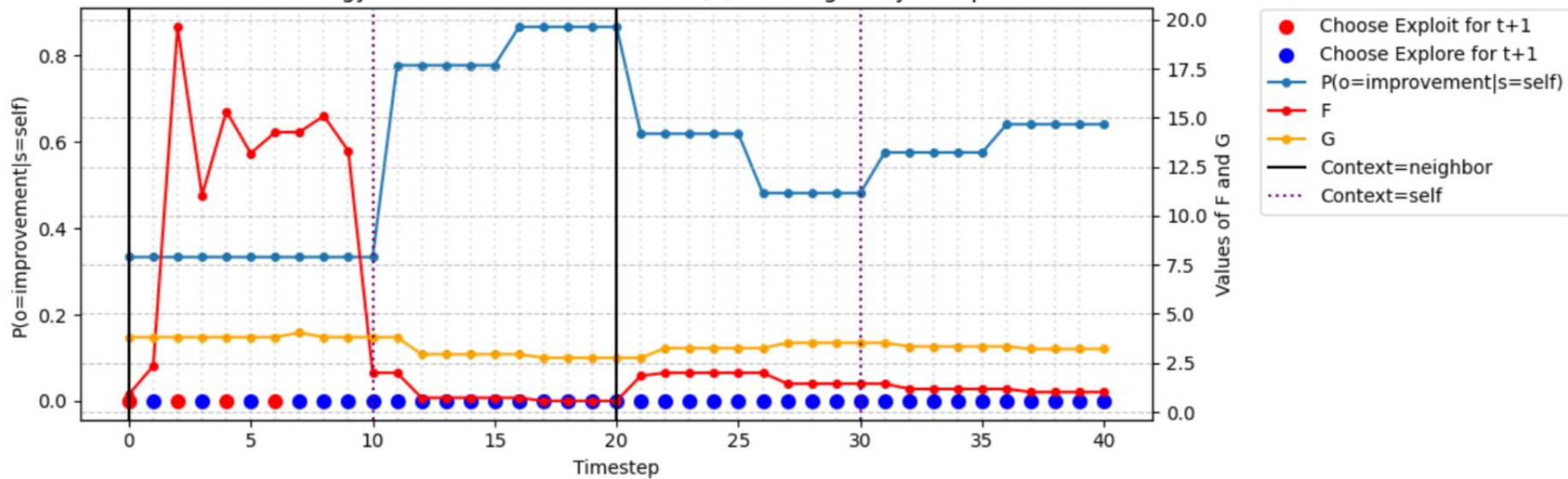
```
q_pi = softmax(G * gamma - F + lnE)

return q_pi, G
```



# Single agent inference

Free Energy Minimization with Likelihood (A) Learning every 5 steps



# References

---

*See speaker notes*

----