Deep Active Inference for Continuous Environments

SFIS Presentation

Tomasz Kawiak

Faculty of Electrical Engineering, Automation, Computer Science and Biomedical Engineering, AGH
Field of study: Computer Science and Intelligent Systems
Specialization: Artificial Intelligence and Data Analysis

April 28, 2025

1 Why Deep Active Inference Matters?

2 Prerequisites and Foundations

3 Deep Active Inference with Monte Carlo Tree Search

4 Deconstructing Deep Active Inference

5 Results

Why Deep Active Inference Matters?

Why Deep Active Inference Matters?

- **Unified Brain Theory Meets Modern AI:** Bridges the gap between theoretical neuroscience (Active Inference / Free Energy Principle) and deep learning techniques.
- **Beyond Reward Maximization:** Provides a principled approach to goal-directed behavior based on minimizing uncertainty (surprise) and maintaining homeostasis, potentially leading to more robust and adaptable agents.
- **Addressing AI Challenges:** Offers potential solutions to limitations in current AI, such as:
  - Sample efficiency in learning.
  - Explainability and interpretability.
  - Robustness to novel situations and uncertainty.
- **Potential Applications:**
  - More human-like robotics and autonomous systems.
  - Improved decision-making in uncertain environments.

Motivation

A common goal in cognitive science and artificial intelligence is to emulate biological intelligence, to gain new insights into the brain and build more capable machines. A widely-studied neuroscience proposition for this is the **free-energy principle**, which views the *brain as a device performing variational (Bayesian) inference*.

Specifically, this principle provides a framework for understanding biological intelligence, termed active inference, by bringing together perception and action under a single objective:

minimizing free energy across time

Prerequisites and Foundations

Variational Inference VI

Variational inference is a technique in Bayesian statistics that approximates complex posterior distributions with simpler, tractable distributions.

VI *transforms the problem of computing the posterior distribution into an optimization problem*. Instead of directly calculating the posterior distribution $p(\theta|D)$, we define a family of simpler distributions $q(\theta)$ and find the one that is closest to the true posterior.

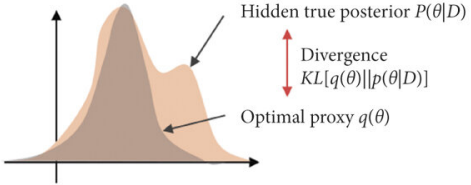

Figure 1: The variational inference for analyzing the optimal posterior distribution $p(\theta|D)$ by estimating a relatively simpler distribution $q(\theta)$. Source [1]

Given observations x, we can build a latent variable model with the variable z (we call it **latent** as it is not directly observed) such that the distribution of interest $p(x)$ can be written as:

$$p(x) = \int_z p(x|z)p(z)dz \tag{1}$$

We condition our observations on some variables we don't know. Therefore, the probability of observations will be the multiplication of the conditional probability and the prior probability of those unknown variables. Subsequently, we integrate out all cases of z to get the distribution of interest $p(x)$.

Unfortunately, this integral is often intractable because it is performed over the whole latent space, which is impractical when latent variables are continuous.

Another issue is how to obtain function that transforms $p(z) \rightarrow p(x)$. For this one could use a neural network with parameters θ to approximate distribution $p(x|z)$, which gives us $p_\theta(x|z)$.

Amortized Variational Inference

To avoid integrating over the whole latent space we can infer any information about z after observing a sample $x_i \in \mathcal{X}$. We can use $q_i(z)$ to approximate the distribution of z given i-th observation.

Sadly, there is a major drawback, namely, the number of parameters of $q_i(z)$ will scale up with the size of the set of observations *because we build individual distribution after observing each data*.

To alleviate this problem, we introduce another neural network with parameters ϕ to parameterize an approximation of $q_i(z)$, i.e. $q_\phi(z|x) \approx q_i(z) \forall x_i \in \mathcal{X}$ such that the increase of the number of parameters is **amortized**.

In other words, we replace all of those $q_i(z)$ distributions with a single parameterized network $q_\phi(z|x)$. This network learns to map from any input observation x to an appropriate distribution over the latent variables z.

Amortization

- Refers to transforming what would traditionally be a separate optimization for each data point (or each time step) into a shared optimization across all data points (or time steps).
- Instead of optimizing variational parameters from scratch at every inference, one learns a recognition model – a neural network – whose weights (amortization parameters) ϕ produce approximate posterior distributions in a single forward pass.
- Recognition model $q_\phi(z|x)$ acts as a proxy for the expensive per-instance optimization with ϕ representing weights of that proxy model.
- Amortized parameters ϕ encapsulate all inferences in a fixed-size model, enabling real-time inference.

Evidence Lower Bound ELBO

Objective function to be optimized

Represents a lower bound on the marginal log-likelihood (or evidence) of the observed data. As earlier mentioned, the goal of variational inference is to put forward a family of distributions and then find the one that is closest to the target.
For the measure of closeness, we can use the Kullback-Leibler divergence (KL divergence) defined as average difference between two log probabilities:

$$D_{KL}[Q(x)||P(x)] \triangleq E_{Q(x)}[\ln Q(x) - \ln P(x)] \tag{2}$$

where E indicates average or expectation.

To successfully setup the optimization for Variational Inference and to find the approximation of the posterior we need to define ELBO, since we cannot compute KL divergence directly.

Active Inference

Under Active Inference, all cognitive operations are conceptualized as inference over *generative model*- a construct that generates predictions about observations.
Generative model can be formulated as the joint probability $P(y, x)$ of observations y and hidden states x that generate those observations. The latter are referred to as *hidden states* or *latent variables* as they are not directly observed.
This joint probability can be decomposed as follows:

- Prior $P(x)$: prior beliefs about the hidden states- agent's knowledge about hidden states prior to observing sensory data.
- Latter $P(y|x)$: likelihood of the observations given the hidden states- agent's knowledge of how observations are generated from states.

Single quantity that Active Inference agent minimize through perception and action is variational free energy (VFE). This means performing variational inference, which in turn implies substituting two intractable parameters:

- posterior $P(x|y)$
- log model evidence $P(y)$

with two quantities that approximate them respectively:

- approximate posterior $Q(x)$
- variational free energy $F[Q, y]$

Thanks to this substitution the problem of Bayesian inference is transformed into an optimization problem, where the agent minimizes the variational free energy with respect to the approximate posterior.

Variational Free Energy

In Active Inference agent minimize *Variational Free Energy* (VFE), which is formally the negative of the *Evidence Lower Bound* (ELBO) of the (log) model evidence. We can interpret Free Energy minimization as finding the best explanation for sensory data, which must be simplest (minimally complex) explanation that is able to accurately account for the data.
Denoted as $F[Q, y]$, where Q is the approximate posterior and y is the observed data:

$$F[Q, y] = D_{KL}[Q(x)||P(x|y)] - \ln P(y) \tag{3}$$

where $P(y)$ is the model evidence (or marginal likelihood) of the observed data, $P(x|y)$ likelihood of the observations given the hidden states.

Variational free energy has a retrospective aspect, as it is a measure of how well the generative model explains the observed data. We can use it to evaluate the quality of the generative model and its parameters. We can think of it as a *training loss function that we want to minimize*.

Quick Notation Note

Please note that from now on we will be using different notation to be more familiar with notation found in reinforcement learning, where o_τ is the observation at time τ, s_τ is the hidden state at time τ, and t is the current time step.
From this we can draw a parallel to the Active Inference framework by noting that the generative model $P(o_\tau|s_\tau)$ is equivalent to the likelihood mapping $P(y|x)$, where y is the observation and x is the hidden state:

$$P(y, x) \equiv P(o_\tau, s_\tau)$$

where y is the observation and o_τ is the observation at time τ, and x is the hidden state and s_τ is the hidden state at time τ.

Expected Free Energy

- Expected Free Energy (EFE) extends Active Inference to include prospective form of cognition: **planning**.
- Planning a sequence of actions requires considering future observations that will be generated by those actions.
- Each possible sequence of actions is termed a *policy*.
- Active Inference treats planning and decision-making as a process of inferring what to do, which brings planning into the realm of Bayesian inference.
- In active inference, agents choose an action given by their EFE. In particular, any given action is selected with a probability proportional to the accumulated negative EFE of the corresponding policies $G(\pi)$.

We define *Expected Free Energy* G for a policy π as:

$$G(\pi) = \sum_{\tau=t+1}^{T} G_\tau(\pi) \tag{4}$$

where $P(o_\tau|s_\tau)$ is the likelihood mapping also called a *Generative Model*, and $G_\tau(\pi)$ is the expected free energy at time τ.

$G_\tau(\pi)$ can be decomposed into two components:

$$
\begin{aligned}
G_\tau(\pi) &= \mathbb{E}_{P(o_\tau|s_\tau)Q(s_\tau|\pi)}[\ln Q(s_\tau|\pi) - \ln P(o_\tau, s_\tau)] \\
&= \mathbb{E}_{P(o_\tau|s_\tau)Q(s_\tau|\pi)}[\ln Q(s_\tau|\pi) - \ln P(s_\tau|o_\tau) - \ln P(o_\tau)] \\
&\approx \mathbb{E}_{P(o_\tau|s_\tau)Q(s_\tau|\pi)}[\ln Q(s_\tau|\pi) - \ln Q(s_\tau) - \ln P(o_\tau)] \quad (\text{using } Q(s_\tau) \approx P(s_\tau|o_\tau)) \\
&= \underbrace{D_{\mathrm{KL}}[Q(s_\tau|\pi)\|Q(s_\tau)]}_{\text{Epistemic Value (Information Gain)}} \underbrace{- \mathbb{E}_{P(o_\tau|s_\tau)Q(s_\tau|\pi)}[\ln P(o_\tau)]}_{\text{Extrinsic Value (Log Preference)}}
\end{aligned}
\tag{5}
$$

where Q is the approximate posterior (variational/recognition distribution), parameterized by ϕ; P is the generative model; $\mathbb{E}_{P(o_\tau|s_\tau)Q(s_\tau|\pi)}$ is the expectation over the joint distribution of observations and hidden states, which captures the agent's predictions about future observations and states when following a specific policy.

- **Epistemic Value**:
  - encourages exploration by reducing uncertainty in state transitions.
  - This value measures how much posterior beliefs $Q(s_\tau|\pi)$ (after following π) would differ from the prior beliefs $Q(s_\tau)$—i.e., the expected reduction in uncertainty
  - A high epistemic value means the chosen policy is expected to yield observations that greatly clarify unknown aspects of the world.
- **Extrinsic Value**:
  - encourages exploitation by maximizing the expected log preference of the observations.
  - This value measures how much the expected observations $P(o_\tau)$ are preferred by the agent.
  - A high extrinsic value means the chosen policy is expected to yield observations that are highly preferred by the agent.

When uncertainty is high, i.e. variational distribution $Q(s_\tau)$ is far from the prior $P(s_\tau|o_\tau)$, the epistemic value will dominate the expected free energy, driving exploration.

Once the agent has learned about the environment, i.e. the variational distribution $Q(s_\tau)$ is close to the prior $P(s_\tau|o_\tau)$, the extrinsic value will dominate, driving exploitation.

As earlier stated, action is selected with a probability proportional to the accumulated negative EFE of the corresponding policies $G(\pi)$ (eq. 4). However, computing $G(\pi)$ for all possible policies is computationally expensive, since it involves making an exponentially-increasing number of predictions for T -steps into the future, and computing all the terms.

Another major issue is that during the calculation of $G(\pi)$, the agent needs to calculate intractable distribution of $P(o_\tau)$.

The question arises: **How can we compute $G(\pi)$ efficiently?**

Monte Carlo Methods

Class of computational algorithms that use random sampling to approximate high dimensional integrals & expectations that are otherwise analytically intractable.

At their core, Monte Carlo methods rely on the law of large numbers, which states that as the number of samples increases, the sample mean converges to the expected value.

For a function $h(x)$ and a random variable X with probability density function $f_X(x)$, the expectation can be written as:

$$E[h(X)] = \int h(x) f_X(x) dx \tag{6}$$

Monte Carlo Sampling

In MC sampling we generate N independent samples $\{x^{(i)}\}_{i=1}^{N}$ from the distribution $f_X(x)$ and approximate the expectation as:

$$E[h(X)] \approx \frac{1}{N} \sum_{i=1}^{N} h(x^{(i)}) \tag{7}$$

- The accuracy of the approximation improves with the number of samples N.

Monte Carlo Tree Search

MCTS extends MC sampling to sequential decision-making problems, in which different potential future trajectories of states are explored in the form of a search tree, giving emphasis to the most likely future trajectories.
MCTS builds a search tree incrementally by repeatedly running simulations. In each iteration, the process is generally divided into 4 phases:

1. **Selection**: Starting from the root node, a tree policy is used to traverse the tree, selecting child nodes that balance exploration and exploitation until a leaf node is reached.
2. **Expansion**: If the leaf node is not a terminal state, one or more child nodes are added to the tree, representing possible future states.
3. **Simulation**: A simulation (or rollout) is performed from the newly added node to a terminal state or until the horizon is reached. The result is an estimated cumulative cost for that action sequence
4. **Backpropagation**: The result of the simulation is propagated back up the tree, updating the values of the nodes along the path taken during selection.

Hidden Markov Model (HMM)

- A statistical model representing a system assumed to be a Markov process with **unobserved (hidden)** states. Core assumptions:
  - *Markov Assumption*
  - *Output Independence Assumption*
- We only observe outputs that are probabilistically dependent on the current hidden state.
- Formally defined by the tuple $\lambda = (S, O, A, B, \pi)$:
  - $S = \{s_1, \ldots, s_N\}$: The set of N hidden states.
  - $O = \{o_1, \ldots, o_M\}$: The set of M possible observations; directly measurable outputs.
  - $A = \{A_{ij}\}$: The state transition probability matrix, where $A_{ij} = P(q_t = s_j | q_{t-1} = s_i)$ denotes probability of transitioning from state i at time $t-1$ to state j at time t.
  - $B = \{B_{kj}\}$: Probability of observing a particular output given that the system is in a specific state, where B_{kj} represents the probability of observing output k when the hidden state is j.
  - $\pi = \{\pi_i\}$: State probabilities, where each element π_i indicates probability of starting in state s_i.

Deep Active Inference with Monte Carlo Tree Search

Amortization parameters

Instead of optimizing variational parameters from scratch at every inference, one learns a recognition model – a neural network – whose weights (amortization parameters) produce approximate posterior distributions in a single forward pass.

Active Inference frames perception and action as joint variational inference over a generative model parameterized by θ and a recognition model parameterized by ϕ.

- Generative model $\theta = \{\theta_o, \theta_s\}$:
  - θ_o: parametrizes the generative model $P_{\theta_o}(o_t|s_t)$
    - θ_o are the amortization parameters of the state-decoder network $P_{\theta_o}(o_t|s_t)$ that maps the current latent state s_t to the distribution over observations o_t.
  - θ_s: parametrizes the transition function $P_{\theta_s}(s_\tau|s_t, a_t)$
    - θ_s are the amortization parameters of the plain feed-forward network that predicts the next state s_τ given the current state s_t and action a_t.
- Recognition model $\phi = \{\phi_s, \phi_a\}$:
  - ϕ_s: parametrizes the recognition model $Q_{\phi_s}(s_t)$
    - ϕ_s are the amortization parameters of the state-encoder network $Q_{\phi_s}(s_t)$
  - ϕ_a: parametrizes the recognition model $Q_{\phi_a}(a_t)$
    - ϕ_a are the amortization parameters of the habitual policy network $Q_{\phi_a}(a_t)$, which maps inferred states to an action distribution, embodying habitual behavior learned through experience.

Note that since tasks used in this presentation have discrete action spaces $\mathcal{A}$, the recognition model $Q_{\phi_a}(a_t)$ is a neural network with parameters ϕ_a and $|\mathcal{A}|$ softmax output units.

Deriving Variational Free Energy for MCTS

First, we are going to transform aforementioned *Variational Free Energy F* in eq. 3 into a more tractable form:

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

Let's focus on the *KL Divergence* term:

$$\begin{aligned} D_{KL}[Q(x)||P(x|y)] &\triangleq E_{Q(x)}[\ln Q(x) - \ln P(x|y)] \\ &= E_{Q(x)}[\ln Q(x) - (\ln P(x, y) - \ln P(y))] \\ &= E_{Q(x)}[\ln Q(x) - \ln P(x, y)] + \ln P(y) \\ &= E_{Q(x)}[\ln Q(x)] - E_{Q(x)}[\ln P(x, y)] + \ln P(y) \end{aligned}$$

After substituting this into the original equation eq. 3, we get:

$$F[Q, y] = E_Q[\ln Q(x)] - E_{Q(x)}[\ln P(x, y)]$$

Since Negative Variational Free Energy is also known as an *Evidence Lower Bound* (ELBO) $\mathcal{L}$, we can write:

$$\mathcal{L} = -F[Q, y] = -E_Q[\ln Q(x)] + E_{Q(x)}[\ln P(x, y)] \tag{8}$$

Knowing that we can plug it into the equation eq. 3:

$$\mathcal{L} = D_{KL}[Q(x)||P(x|y)] - \ln P(y)$$
$$\ln P(y) = D_{KL}[Q(x)||P(x|y)] + \mathcal{L}$$

Since the true posterior $P(x|y)$ is intractable and hence we canot calculate KL divergence analytically, we get an important property of non-negativity of KL divergence.

$$\ln P(y) \geq \mathcal{L} \tag{9}$$

Calculating Variational Free Energy

Finally, we can exploit the above equation eq. 9 and eq. 3 to calculate the *Variational Free Energy* for each time-step t as:

$$\begin{aligned} F_t = &- E_{Q_\phi(s)(s_t)}[\ln P_{\theta_o}(o_t|s_t)] + D_{KL}[Q_{\phi_s}(s_t)||P_{\theta_s}(s_t|s_{t-1}, a_{t-1})] \\ &+ E_{Q_{\phi_s}(s_t)}[D_{KL}[Q_{\phi_a}(a_t)||P(a_t)]] \end{aligned} \tag{10}$$

where:

$$P(a) = \sum_{\pi:a_1=a} P(\pi) \tag{11}$$

is the summed probability of all policies π that start with action a. We assume that s_t is normally distributed and o_t is *Bernoulli distributed* (which means that the observation o_t at time t is a binary outcome (e.g., 0 or 1)), with all parameters given by a neural network, parameterized by θ_0,θ_s and ϕ_s for the observation, transition, and encoder models, respectively. The expectations over $Q_{\phi_s}(s_t)$ are taken via MC sampling, using a single sample from the encoder.

Expected Free Energy EFE

Now, let's consider EFE. At time step t and for a time horizon up to time T, EFE is defined as:

$$G(\pi) = \sum_{\tau=t}^{T} G(\tau, \pi) = \sum_{\tau=t}^{T} E_{\tilde{Q}}[\log Q(s_\tau, \theta|\pi) - \log \tilde{P}(o_\tau, s_\tau, \theta|\pi)] \tag{12}$$

where $\tilde{Q} = Q(o_\tau, s_\tau, \theta|\pi) = Q(\theta|\pi)Q(s_\tau|\theta, \pi)Q(o_\tau|s_\tau, \theta, \pi)$ and
$\tilde{P} = P(o_\tau, s_\tau, \theta|\pi) = P(o_\tau|\pi)Q(s_\tau|o_\tau)P(\theta|s_\tau, \theta, o_\tau)$

$$
\begin{aligned}
G(\pi, \tau) = &- \mathbb{E}_{\tilde{Q}}[\log P(o_\tau|\pi)] \\
&+ \mathbb{E}_{\tilde{Q}}[\log Q(s_\tau|\pi) - \log P(s_\tau|o_\tau, \pi)] \\
&+ \mathbb{E}_{\tilde{Q}}[\log Q(\theta|s_\tau, \pi) - \log P(\theta|s_\tau, o_\tau, \pi)]
\end{aligned} \tag{13}
$$

where $P(o_\tau|s_\tau, \theta)$ is the likelihood mapping also called a *Generative Model*, and $G(\pi, \tau)$ is the expected free energy at time τ.

Unpacking the EFE

1. **Reward-seeking term**:

$$-\mathbb{E}_{\tilde{Q}}[\log P(o_\tau|\pi)] \tag{14}$$

- This term is responsible for picking actions that make the agent's observations more likely.
- Measure of how well the agent's generative model predicts the observations it receives. The more likely the observations are given the policy, the lower the expected free energy.

We could compare this to the *reward* term in *Reinforcement Learning*, where the agent is rewarded for taking actions that lead to desirable outcomes.
In this case if we encode the reward as certain desirable observations having high prior $P(o_\tau|\pi)$, then minimizing this term is equivalent to maximizing expected log-probability of those observations – just as the RL agent would maximize expected reward.

2. **State-uncertainty term**:

$$\mathbb{E}_{\tilde{Q}}[\log Q(s_\tau|\pi) - \log P(s_\tau|o_\tau, \pi)] \tag{15}$$

- Mutual information between the agent's beliefs about its latent representation of the world, before and after making an observation.
- Reflects a motivation to explore areas of the environment that resolve state uncertainty.

3 **Policy-uncertainty term**:

$$\mathbb{E}_{\tilde{Q}}[\log Q(\theta|s_\tau, \pi) - \log P(\theta|s_\tau, o_\tau, \pi)] \tag{16}$$

- Tendency of Active Inference agents to reduce their uncertainty about model parameters via new observations.
- Referred to as *active learning*, *novelty* or *curiostiy* in the literature.

MC Sampling to the rescue

Sadly, two out of three terms in the EFE are intractable. Therefore, we will use *Monte Carlo* (MC) sampling to approximate the intractable terms.

$$G(\pi, \tau) = -\mathbb{E}_{Q(\theta|\pi)Q(s_\tau|\theta,\pi)Q(o_\tau|s_\tau,\theta,\pi)}[\log P(o_\tau|\pi)] \tag{17}$$

$$+\mathbb{E}_{Q(\theta|\pi)}[\mathbb{E}_{Q(o_\tau|\theta,\pi)}H(s_\tau|o_\tau,\pi) - H(s_\tau|\pi)] \tag{18}$$

$$+\mathbb{E}_{Q(\theta|\pi)Q(s_\tau|\theta,\pi)Q(o_\tau|s_\tau,\theta,\pi)}H(o_\tau|s_\tau,\theta,\pi) - \mathbb{E}_{Q(s_\tau|\pi)}H(o_\tau|s_\tau,\pi) \tag{19}$$

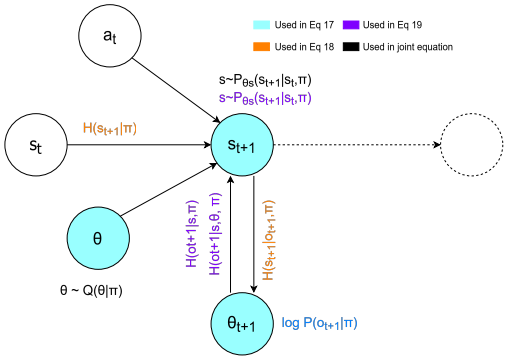

Figure 2: ulala

Action Selection

In active inference, agents choose an action given by their EFE. In particular, any given action is selected with a probability proportional to the accumulated negative EFE of the corresponding policies $G(\pi)$, defined in eq. 4.

We can write the process of action selection in active inference as sampling from the distribution:

$$P(\pi) = \sigma(-G(\pi)) = \sigma(-\sum_{\tau > t} G(\pi, \tau))$$

Where $\sigma(\cdot)$ is the softmax function.

Again we are faced with computational issues, since computing $G(\pi)$ for all possible policies is computationally expensive, since it involves making an exponentially-increasing number of predictions for T -steps into the future, and computing all the terms in eq. 17. We are going to employ two methods operating in tandem:

First, we employ Monte Carlo Tree Search (MCTS) to calculate the distribution over actions $P(a_t)$, defined in eq. 11, and control the agent's final action selection.

During the MCTS process, the agent generates a weighted search tree iteratively that is later sampled during action selection.
In each MCTS loop, one plausible state-action trajectory – starting from the current time-step t – is calculated.
For states that are explored for the first time, the distribution $P_\theta(s_{t+1}|s_t, a_t)$ is used.

States that have been explored are stored in the buffer search tree and accessed during later loops of the same planning process. The weights of the search tree $\tilde{G}(s_t, a_t)$ represent the agent's best estimation for EFE after taking action a_t in state s_t.

An upper confidence bound (UCB) for $G(s_t, a_t)$ is calculated as:

$$U(s_t, a_t) = \tilde{G}(s_t, a_t) + c_{\text{explore}} \cdot Q_{\phi_a}(a_t|s_t) \frac{1}{1 + N(a_t, s_t)} \tag{20}$$

where $N(a_t, s_t)$ is the number of times action a_t was explored from state s_t, and c_{explore} is a hyperparameter that controls exploration.

In each round, the EFE of the newly-explored parts of the trajectory is calculated and back-propagated to all visited nodes of the search tree. Additionally, actions are sampled in two ways. Actions from states that have been explored are sampled from $\sigma(U(s_t, a_t))$, while actions from new states are sampled from $Q_{\phi_a}(a_t)$.

Habitual Network

Our second approach to tackling the problem of computational cost in eq. 17 is to make use of amortized inference through a *Habitual Neural Network* that directly approximates the distributions over actions, which we parameterize by ϕ_a and denote as $Q_{\phi_a}(a_t)$.

In essence, $Q_{\phi_a}(a_t)$ acts as a variational posterior that approximates $P(a_t|s_t)$, with a prior $P(a_t)$ calculated by MCTS.
This means that this network *amortizes* the MCTS solution.
During learning, this network is trained to reproduce the last executed action q_{t-1} (selected by sampling $P(a_t)$) given the current state s_{t-1}

Through the combination of the approximation $Q_{\phi_a}(a_t)$ and the MCTS, the agent has at its disposal two methods of action selection.
We refer to $Q_{\phi_a}(a_t)$ as the *habitual network*, as it corresponds to a form of fast decision-making, quickly evaluating and selecting a action; in contrast with the more deliberative system that includes future imagination via MC tree traversals.

State Transition

State Transition Model: $P_{\theta_s}(s_t|s_{t-1}, a_{t-1})$ is a key element of the Active Inference framework, which belongs to the agent's generative model.

In this model, we take $s_t \backsim \mathcal{N}(\mu, \sigma^2/\omega_t)$, where the μ and σ^2 are respectively the linear and softplus units of a neural network with parameters θ_s applied to s_{t-1}; ω_t is the *precision factor* modulating the uncertainty on the agent's estimate of the hidden state of the environment.

To get μ we take the output (final hidden activation $h \in \mathbb{R}^d$) of the neural network used for parameterization of the transition function:

$$\mu = W_\mu h + b_\mu$$

where W_μ and b_μ are the weights and bias of the linear layer. The resulting μ is the vector of Gaussian means for each of the d latent dimensions of the state space.

To get σ^2 we take the output of the neural network and apply a softplus activation function:

$$\rho = W_\rho h + b_\rho, \quad \sigma = \text{softplus}(\rho) = \ln(1 + e^\rho)$$

All W_μ, b_μ, W_ρ, and b_ρ are parameters of the neural network with parameters θ_s.

We model the precision factor as a logistic (sigmoid) function of the belief update about the agent's current policy:

$$\omega_t = \frac{\alpha}{1 + e^{-\frac{b - D_{t-1}}{c}}} + d \tag{21}$$

where $D_t = D_{KL}[Q_{\phi_a}(a_t)||P(a_t)]$ is the Kullback-Leibler divergence between the habitual policy $Q_{\phi_a}(a_t)$ and the prior $P(a_t)$ (eq. 11), and α, b, c, and d are hyperparameters.

Understanding ω_t

In practice, the effect of ω_t is to incentivize disentanglement in the latent state representation s_t - the precision factor ω_t effectively pushes the state encoder $Q_{\phi_s}(s_t)$ to have independent dimensions. As training progresses and the habitual network becomes a better approximation of $P(a_t)$, ω_t is gradually increased, implementing a natural form of precision annealing.

Deconstructing Deep Active Inference

Deep Hidden Markov Model DHMM

Deep Critical Hidden Markov Model DCHMM

Deep Active Inference Agent

Results

Why it failed

Lessons Learned & Future Work

Thank you for your attention

Questions?

[1] Mu, B., Qin, B., Yuan, S. and Qin, X. 2020. A climate downscaling deep learning model considering the multiscale spatial correlations and chaos of meteorological events. *Mathematical Problems in Engineering*. 2020, (Nov. 2020), 1–17. DOI:https://doi.org/10.1155/2020/7897824.