Hierarchical Active Inference Agent with RG-Flow and Poisson-Dirichlet Process

Christian Eidahl: CSCI 8980: AI for Sequential Decision Making, Spring 2025
University of Minnesota

1 Introduction

Active inference is a framework for sequential decision making that offers an alternative to reinforcement learning (RL) by leveraging generative models of the environment and treating action selection as inference. This project explores an active inference agent that operates in uncertain environments using a hierarchical generative model. The central problem addressed is how to enable an agent to explore and act under uncertainty by maintaining a latent model of its environment's dynamics across multiple spatial and temporal scales.

The main goal of this project is to use model-based planning and unsupervised representation learning for decision making. By using a learned generative model of the environment, the agent can predict outcomes, reason about latent causes, and plan actions without relying on reward maximization in the typical RL sense. Instead of learning a policy solely from trial-and-error, the agent leverages active inference principles to minimize uncertainty and surprise. This approach relates to the course in the three following ways: (i) **Model-based planning**: the agent uses an internal model (a form of world model) to simulate and evaluate action outcomes; (ii) **Generative modeling**: techniques like normalizing flows (e.g. RG-Flow) are used to learn latent representations of observations; (iii) **Active inference vs RL**: the agent plans by minimizing variational free energy, providing an information-theoretic alternative to reward-driven RL for guiding behavior.

This project aims to demonstrate how a hierarchical generative model can be used for active exploration and control in a dynamic environment, and combines RG-Flow, a multi-scale flow-based model fit to a sparse latent distribution, with a Poisson-Dirichlet Process (PDP) to automatically cluster observations within the latent space, to build an agent that can continually refine its understanding of the environment online while exploring.

2 Problem Statement

The agent's task is formulated as a partially observable Markov decision process (POMDP). The environment at each discrete time step t is characterized by:

Component	Description
$s_t \in \mathcal{S}$	Hidden state at time t, not directly observable by the agent.
$o_t \in \mathcal{O}$	Observation received at time t, generated from the hidden state (e.g., sensor reading or image).
$a_t \in \mathcal{A}$	Action taken by the agent at time t to influence the state (e.g., control input).
$p(s_{t+1} \mid s_t, a_t)$	State transition dynamics defining how the hidden state evolves, including natural dynamics and action effects.
$p(o_t \mid s_t)$	Observation likelihood defining how observations are produced from states (the generative model for observations).

Table 1: POMDP Setup

Unlike a standard POMDP for RL, in active inference a prior preference or "goal" distribution over outcomes, denoted $p(o_t^*)$, instead of an explicit reward function. For example, in a CartPole balancing task, observations where the pole is upright would have higher prior probability (reflecting a desired outcome). The agent's objective is not to maximize cumulative reward, but to select actions that make future observations align with these preferred outcomes while reducing uncertainty about the state.

The agent maintains a generative model that mirrors the POMDP: it encodes a belief about state transitions and observations. The generative model over a trajectory can be written as $\tau_{0:T} = \{s_{0:T}, o_{0:T}\}$ (with actions chosen by some policy π) as:

$$p_\theta(s_{0:T}, o_{0:T} \mid \pi) = p_\theta(s_0) \prod_{t=0}^{T-1} p_\theta(o_t \mid s_t)\, p_\theta(s_{t+1} \mid s_t, a_t),$$

This formulation can be extended to a hierarchical or "deep" model with multiple layers of latent variables, For this project, the states within the hierarchical model, s_t are multilevel latent objects governing states over different timescales or spatial dimensions.

The agent cannot directly observe the environment's true state s_t so it maintains an approximate posterior $q_\phi(s_{0:t} \mid o_{0:t})$ over states and trajectories given its observations. The quality of this posterior is measured by the variational free energy (VFE), a measure of the discrepancy between the agent's beliefs and the environments true generative model. At time t, given new observation o_t, the agent updates its beliefs by minimizing the VFE:

$$F_t = \mathbb{E}_{q_\phi(s_{0:t})}\big[\ln q_\phi(s_{0:t} \mid o_{0:t}) - \ln p_\theta(s_{0:t}, o_{0:t})\big],$$

which is equivalent to minimizing the Kullback–Leibler divergence $\mathrm{D}_{\mathrm{KL}}(q_\phi(s_{0:t}) \,\|\, p_\theta(s_{0:t} \mid o_{0:t}))$. This process adjusts the agent's internal state estimates to better explain the observations.

To select actions at the current timestep, the agent employs active inference by choosing a_t to minimize the expected free energy of future observations. This means the agent will use its internal generative model to roll out a trajectory and evaluates the expected free energy $\mathbb{G}(a_t)$ for each possible action a_t, and then chooses the action that minimizes this value, planning ahead to fulfill its goals in a principled Bayesian manner.

Overall the problem setup is: the agent must actively maintain an accurate belief over a partially observed, evolving state and choose actions that keep the belief and observations in alignment with the agent's prior preferences. By structuring the agent's generative model hierarchically (Section 3) and using variational inference, this project creates an agent that is capable of model-based planning under uncertainty from large observation spaces.

3 Approach

The approach for this project combines (1) RG-Flow, a hierarchical normalizing flow model for learning disentangled latent factors of observations at multiple scales and (2) a Poisson-Dirichlet Process (PDP) model for nonparametric clustering of latent states, enabling the model to grow its state space as needed. These are integrated into the HMM_agent class, which implements a multi-layer hidden Markov model (HMM) for both state estimation and action selection.

3.1 RG-Flow for Mapping Raw Data to a Hierarchical Latent Space

RG-Flow (Renormalization Group Flow) is a type of normalizing flow that imposes a hierarchical, multi-scale structure on the latent space. Normalizing flow models learns a bijective transformation $f : x \leftrightarrow z$ between data x (e.g. an image) and a latent vector z with a base distribution. RG-Flow augments this by breaking z into parts corresponding to different spatial scales of x. This project extends this model to map sequences of images to a latent space instead of just single images as well as fitting the data to a sparser prior in the form of a Gamma(1.5,.75), to encourage further disentangling of latent variables and allow the PDP to more easily cluster features. The number of hierarchical layers in the RG-Flow model will determine how many layers there are in the HMM. For this project there are 4 spatial layers and 2 temporal layers. The idea behind this is that a latent variable at a higher level will act as a coarse summary of the latent variables below.

In this agent each latent group of variables is clustered into an observation by the PDP which is then used as an observation for the HMM. Each layer can be seen as a level of representation: the lowest-level latents encode local, fine-scale details of the observation, while higher-level latents encode more abstract, coarse features (global structure). By training an RG-Flow model on observations (or using a pre-trained one), the agent obtains a function:

$$z_t^{(L)}, z_t^{(L-1)}, \ldots, z_t^{(1)} = f_{\text{enc}}(o_t),$$

where $z_t^{(\ell)}$ denotes the latent code at level ℓ (with $\ell = L$ being the topmost, most abstract layer, and $\ell = 1$ the lowest). The dimension of $z_t^{(\ell)}$ and its interpretability correspond to the information content at that scale of the data. Importantly, RG-Flow's design ensures that these latents are disentangled to a degree — each $z^{(\ell)}$ affects the observation in a localized or scale-specific manner. For example, in an image-based environment, $z_t^{(L)}$ might control global properties (like background or overall shape), whereas $z_t^{(1)}$ might control small local details.

3.2 Nonparametric Clustering with the Poisson–Dirichlet Process

Continuous latent vectors $z_t^{(\ell)}$ from RG-Flow are discretized into an evolving set of observations $o_t^{(\ell)}$ via a two-parameter Poisson–Dirichlet Process. The latent variables are assumed to fall into types that will represent features about the data that has been encoded. Because the flow model assumes that its latent space is parameterized by independent latent variables, each type of latent vector can be represented as a vector of means and variances no covariance following a gaussian distribution.

The clustering procedure is as follows:

Stick-breaking weights Given discount a and concentration b ($0 \le a < 1$, $b > -a$), the stick-breaking proportions are

$$v_k = \frac{1-a}{1-a+b+(k-1)\,a}, \quad p_k = v_k \prod_{j=1}^{k-1}(1-v_j).$$

Density-based candidate selection A local density estimate is computed for each data point (as the inverse of the mean distance to its nearest neighbors). Only points exceeding a density quantile threshold τ_{density} are considered for new-cluster proposals. These are ranked by their residual (negative best-fit log-likelihood under existing clusters), and a subset is chosen by enforcing a minimum separation constraint among proposals.

Free-energy objective Cluster assignments minimize the variational free energy

$$F = \sum_{k=1}^{K} F_{\text{NIG}}(n_k, \kappa_k, \alpha_k, \beta_k) + \sum_{i=1}^{N} \sum_{k=1}^{K} r_{ik} \big(\ln r_{ik} - \ln p_k\big),$$

where the first term is the Normal–Inverse–Gamma prior contribution and the second is the likelihood/entropy cost of soft assignments r_{ik}.

Candidate evaluation From up to M top proposals, all subsets (subject to a maximum cluster budget) are enumerated. Any subset that lowers F is accepted, increasing K and updating the cluster set.

Posterior updates For each cluster k with effective count n_k, weighted mean $\bar{x}_k$ and variance s_k^2, the posterior hyperparameters are updated in closed form:

$$\begin{aligned}
\kappa_k' &= \kappa_k + n_k, \\
m_k' &= \frac{\kappa_k m_k + n_k \bar{x}_k}{\kappa_k + n_k}, \\
\alpha_k' &= \alpha_k + \tfrac{n_k}{2}, \\
\beta_k' &= \beta_k + \tfrac{n_k\, s_k^2}{2} + \frac{\kappa_k n_k}{2(\kappa_k + n_k)} (\bar{x}_k - m_k)^2.
\end{aligned}$$

Capacity control The total number of clusters is capped at $\max K$, and only active clusters are retained to ensure computational efficiency.

The PDP soft assigns each data point to a cluster so that its output is a probability vector over the state space. This is used as the observation at each level of the HMM. The convenient part of using soft assignments is even if a point doesn't strongly belong to any cluster, the combined probability of it belonging to multiple clusters will inform the model of where it is within the latent space.

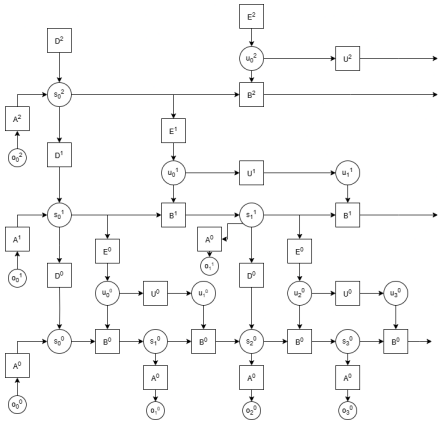
3.3 Hidden Markov Model and Inference Process

The HMM learns and plans as follows:

1. **Encode observations:** Map input data to the latent space via the RG-Flow. The PDP fits to and soft assigns the data point to its current learned clusters and passes the clustered latent variable to the HMM as an observation in the form of probability distribution over all currently discovered states. If the data point causes the PDP to add new clusters then the shapes of all prior matrices are updated so that they grow with the size of the state space.

2. **Episode initialization:** At episode start, latent state and policy beliefs are reset according to the mean of the priors defined in Figure 1.

3. **Iterative belief update:** Perform sweeps through all layers; at each layer, combine messages from observation likelihoods ($A^{(\ell)}$), within-layer transitions ($B^{(\ell)}$), and cross-layer couplings ($D^{(\ell)}$, $E^{(\ell)}$) to refine posterior distributions. This is determined to converge when the decrease in the total free energy of the model between sweeps falls below a small tolerance threshold.



Matrix	Definition	Shape	Update rule	(Message)
A	$p(o_t \mid s_t)$	$\|O\| \times \|S\|$	$A \leftarrow A + \Delta A$	$\Delta A = o_t \times (s_t)$
B	$p(s_{t+1} \mid s_t, u_t)$	$\|S\| \times \|S\| \times \|U\|$	$B \leftarrow B + \Delta B$	$\Delta B = (s_t) \otimes (u_t) \otimes (s_{t+1})$
C	$p(o)$	$\|O\|$	$C \leftarrow C + \Delta C$	$\Delta C = (o_t)$
D	$p(s_t^{(\ell)} \mid s_t^{(\ell+1)})$	$\|S^{(\ell)}\| \times \|S^{(\ell+1)}\|$	$D \leftarrow D + \Delta D$	$\Delta D = (s_t^{(\ell+1)}) \times (s_t^{(\ell)})$
E	$p(u_t^{(\ell)} \mid s_t^{(\ell+1)})$	$\|U^{(\ell)}\| \times \|S^{(\ell+1)}\|$	$E \leftarrow E + \Delta E$	$\Delta E = (s_t^{(\ell+1)}) \times (u_t^{(\ell)})$
U	$p(u_{t+1} \mid u_t)$	$\|U\| \times \|U\|$	$U \leftarrow U + \Delta U$	$\Delta U = (u_{t+1}) \times (u_t)$

Figure 1: Temporal Hierarchical HMM structure (top) and prior matrices with shapes, simplified update rules, and corresponding belief messages (bottom). The direction of arrows show the flow of information as the model plans and acts. The starting states at each level are inferred from observations and prior beliefs at t=0. Prior beliefs are updated before the model rolls out a trajectory to determine the path that will minimize free energy. A encodes the models belief about what observations map to states. B is the mapping from s_t to s_{t+1} given a policy u_t, C is the preference vector that encodes preferences over observations. D is the mapping from a higher state to the starting state of the layer below it. E is the prior over policy given the state of the layer above. U maps the current policy to the policy at the next timestep. Priors are shared for for all states within a layer and each layer has its own set of priors. Priors are dirichlet distributions whose concentration parameters are a running sum of what information the model has seen and how it has acted. All prior concentration parameters are initialized as uniform matrix of 1's with a small amount of noise.

4. **Action selection:** Once beliefs have stabilized, perform short-horizon rollouts under the learned dynamics (B) and preferences (C); select the action that minimizes the expected free energy. Planning begins with the uppermost temporal layer so that all lower states are conditioned on the predicted state of the layer above it.

5. **Predictive propagation:** Store current beliefs as "previous" for the next step and update state and policy distributions via one-step predictive equations to initialize the next inference cycle.

4 Results

To show that the agent is capable of learning and planning within the Cartpole environment it is compared against a baseline of a random agent. The agent is first trained on 500 collected episodes of random exploration so that it's priors can be learned and the PDP can fit to the latent space. The preference vector sums the observation weighted by the current reward at each timestep, so that the agent will learn to prefer those observations. The agent then goes online and operates within the environment. Performance is then compared with a random agent over 20 episodes.

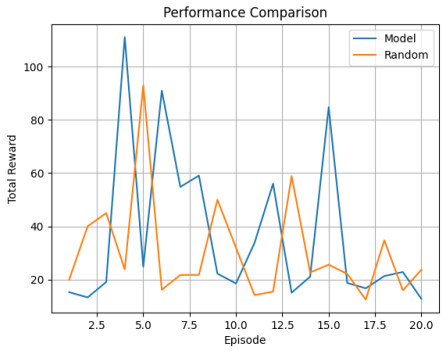


Figure 2: Total reward per episode for the learned CartPole agent vs. a random baseline over 20 episodes.

See Github for gifs of samples from the RG-flow model and the agent acting in the environment. .

5 Analysis

The model seems to be able to perform a better than random but struggles to maintain performance over a long period of time within a single episode or between episodes. Considering the model is only trained on 200 episodes this is fairly good performance. This may be happening because the learning rates aren't tuned well so when it switches to running online it could be corrupting the priors when only updating based on a single sample. Another reason is that the PDP seems to be underfitting in the latent space for the upper spatial and temporal layers. Ambiguity in the latent space will present itself in poor updates and action choice. This could be because the higher level latent variables of the RG-flow model struggle to disentangle as each variable has to encode a large amount of information, so a fix could be to relax the assumption that the latents are independent and incorporate that into my PDP. Another reason is that the episodes that the model performed poorly on were not represented in the data that was used to warm up the model priors, so the model hasn't learned how to predict and act in those states. A big drawback of this current experimental setup is that the agent is relying on random actions to collect training data when one of the strengths of the active inference framework is the exploration term that's baked into the free energy objective. Saving a replay buffer as the agent explores could be one way to improve online learning rather than rely on random actions. This information seeking behavior could also be leading the agent towards unknown states as it operates online.

6 Conclusion

Initially, this project had a much larger scope and was going to try to train on the Atari 100k dataset and see how well it would compare with other models, as well as how well it could generalize if it only sees a small subset of the dataset. In hindsight that was too much for a single semester and even this scaled back version to only learn to play cartpole turned out to be too large. Early results are promising but I wasn't left with enough time to compare this model with others or human performance. I think a more realistic goal for this semester would have been to create the active inference agent using discrete observations on a simple grid world environment, as simply programming the HMM turned out to be a massive task on its own. The main roadblocks I faced during this project were getting the RG-flow model to fit a dataset to a gamma distribution well. Another roadblock was getting the PDP to fit to data well. There were instances at the start of building

the PDP where I thought it was working well but that was only because I was testing on artifically generated data that the PDP was predisposed to fit well too and failed horribly when I tried using it with the RG-flow model initially. The main struggle for this entire project was that I had a big picture idea of how each piece of the model would integrate together but not on a low level until late in the semester. This is partly also due to the fact that I was using this as an opportunity to get a deeper understanding active inference framework as that pertains to my research area of interest, and only really understood it on a conceptual level before. The main thing I learned is that I should be visualizing everything I can about the model, as simple print statements of losses or cluster assignments can be very misleading as to what the model is actually doing.

Overall I'm satisfied with where this project has ended up as it shows that this model setup can learn and plan on raw sensory data, and gives me a solid foundation to build for future research.

References

[1] Hu, H., Wu, D., You, Y., Olshausen, B., & Chen, Y. (2022). *RG-Flow: A hierarchical and explainable flow model based on renormalization group and sparse prior*. Machine Learning: Science and Technology,

[2] Friston, K., et al. (2024). *From pixels to planning: scale-free active inference*. arXiv:2407.20292

[3] Buntine, Wray, and Marcus Hutter. "A Bayesian view of the Poisson-Dirichlet process." arXiv preprint arXiv:1007.0296 (2010).