A Step-by-Step Tutorial on Active Inference and its Application to Empirical Data

Ryan Smith[1*], Karl J. Friston[2], Christopher J. Whyte[3*]

*These authors contributed equally.

Author affiliations:

[1]Laureate Institute for Brain Research, Tulsa, OK, USA.
[2]Wellcome Centre for Human Neuroimaging, Institute of Neurology, University College London, WC1N 3AR, UK.
[3]MRC Cognition and Brain Sciences Unit, University of Cambridge, Cambridge, UK.

Corresponding author:
Ryan Smith
Laureate Institute for Brain Research
6655 S Yale Ave, Tulsa, OK 74136, USA
Email: rsmith@laureateinstitute.org

Abstract

The active inference framework, and in particular its recent formulation as a partially observable Markov decision process (POMDP), has gained increasing popularity in recent years as a useful approach for modelling neurocognitive processes. This framework is highly general and flexible in its ability to be customized to model any cognitive process, as well as simulate predicted neuronal responses based on its accompanying neural process theory. It also affords both simulation experiments for proof of principle and behavioral modelling for empirical studies. However, there are limited resources that explain how to build and run these models in practice, which limits their widespread use. Most introductions assume a technical background in programming, mathematics, and machine learning. In this paper we offer a step-by-step tutorial on how to build POMDPs, run simulations using standard MATLAB routines, and fit these models to empirical data. We assume a minimal background in programming and mathematics, thoroughly explain all equations, and provide exemplar scripts that can be customized for both theoretical and empirical studies. Our goal is to provide the reader with the requisite background knowledge and practical tools to apply active inference to their own research. We also provide optional technical sections and several appendices, which offer the interested reader additional technical details. This tutorial should provide the reader with all the tools necessary to use these models and to follow emerging advances in active inference research.

Keywords: Active Inference; Computational Neuroscience; Bayesian Inference; Learning; Decision-Making; Machine Learning

Introduction

Active inference, and in particular its recent application to partially observable Markov decision processes (POMDPs; defined below), offers a unified mathematical framework for modelling perception, learning, and decision making (Da Costa, Parr, et al., 2020; Friston, Parr, & de Vries, 2017; Friston, Rosch, Parr, Price, & Bowman, 2018; Parr & Friston, 2018b). This framework has become increasingly influential within psychology, neuroscience, and machine learning. Over the last decade there have been many articles that offer either 1) broad intuitions about the workings and potential implications of active inference (e.g., (Badcock, Friston, Ramstead, Ploeger, & Hohwy, 2019; A. Clark, 2013, 2015; J. E. Clark, Watson, & Friston, 2018; Hohwy, 2014; Pezzulo, Rigoli, & Friston, 2015, 2018; Smith, Badcock, & Friston, 2020)), or 2), technical presentations of the mathematical formalism and how it continues to evolve (e.g., (Da Costa, Parr, et al., 2020; Friston, FitzGerald, et al., 2016; Friston, FitzGerald, Rigoli, Schwartenbeck, & Pezzulo, 2017; Friston, Parr, et al., 2017; Hesp, Smith, Allen, Friston, & Ramstead, 2020; Parr & Friston, 2018b)). However, for those first becoming acquainted with this field, the former class of articles does not provide sufficient detail to instill a thorough understanding of the framework, leading to potential misunderstanding and potentially inaccurate empirical predictions. At the other extreme, the latter class of articles is highly technical and requires considerable mathematical expertise, familiarity with notational conventions, and the broader ability to translate the mathematical formalism into empirical predictions relevant to a given field of study. This has made the active inference literature less accessible to a broader audience who might otherwise benefit from engaging with it. To date, there are also relatively few materials available for students seeking to gain the practical skills necessary to build active inference models and apply them to their own research aims (although some very helpful material has been prepared by Philipp Schwartenbeck: https://github.com/schwartenbeckph).

The goal of this paper is to provide an accessible tutorial on the POMDP formulation of active inference that is easy to follow for readers without upper-level undergraduate/graduate-level training in mathematics and machine learning, while simultaneously offering basic mathematical understanding—as well as the practical tools necessary to build and use active inference models for their own purposes. We review the conceptual and formal foundations and provide a step-by-step guide on how to use code in MATLAB (provided in **supplementary materials**) to build active inference (POMDP) models, run simulations, fit models to empirical data, perform model comparison, and perform further steps necessary to test hypotheses using both simulated and fitted empirical data (all **supplementary code** can also be found at: https://github.com/rssmith33/Active-Inference-Tutorial-Scripts). We have tried to assume as little as possible about the reader's background knowledge, in hope of making these methods accessible to researchers (e.g., psychologists and neuroscientists) without a strong background in mathematics or machine learning. However, we have also included sections that provide additional technical detail, which the pragmatic reader can safely skip over and still follow the practical tutorial aspects of the paper. We have also provided additional material in **supplementary materials** with:

1) Definitional material to help the non-expert reader who would like to attempt the technical sections.
2) Additional mathematical detail for interested readers with a stronger technical background.

3) Pencil and paper exercises that help build an intuition for the behavior of these models.
4) A stripped down but well commented version of the standard model inversion script used for running simulations that can serve as a springboard for readers seeking a deeper understanding of the code that implements the models.

Throughout the article, we will refer to the associated MATLAB code, assuming the reader is working through the paper and the code in parallel.

While we assume as little mathematical background as possible, some limited knowledge of probability theory, calculus, and linear algebra will be necessary to fully appreciate some sections of the tutorial. Building models in practice also requires some basic familiarity with the MATLAB programming environment. We realize that this mathematical background is nontrivial. However, to minimize these potential hurdles, we 1) provide thorough explanations when presenting the mathematics and programming (with further expansion in **supplementary materials**), 2) include hands on examples/exercises in the companion MATLAB code, and 3) provide pencil and paper exercises (see **Pencil_and_paper_exercise_solutions.m** code) that readers can work through themselves. In total, this tutorial should offer the reader the necessary resources to:

1) Acquire a basic understanding of the mathematical formalism.
2) Build generative models of behavioral tasks and run simulations.
3) Fit models to behavioral data and recover model parameters on an individual basis, which can then be used for subsequent (e.g., between-subjects) analyses. Our hope is that this will increase the accessibility and use of this framework to a broader audience.

Note, however, that our focus is specifically on the POMDP formulation, which models time in discrete steps and treats beliefs and actions as discrete categories (referred to as "discrete state-space" models). This means that we do not cover a number of other topics associated with active inference and the broader free energy principle from which it is derived. For example, we do not cover "continuous state-space" models, which can be used to model perception of continuous variables (e.g., brightness; for a tutorial, see (Bogacz, 2017)) as well as motor control processes (e.g., controlling continuous levels of muscle contraction; see (Adams, Shipp, & Friston, 2013; Buckley, Sub Kim, McGregor, & Seth, 2017)). Nor do we cover "mixed" models, in which discrete and continuous state-space models can be linked – allowing decisions to be translated into motor commands (e.g., see (Friston, Parr, et al., 2017; Millidge, 2019; Tschantz et al., 2021)). We also do not cover work on free energy minimization in self-organizing systems or the basis of the free energy principle in physics. The most thorough technical introduction to the physics perspective can be found in (Friston, 2019); a less technical (but still rigorous) introduction is presented in (Andrews, 2020)[1].

[1] This other work appeals to a number of common constructs discussed in the free energy principle literature that are also not covered here, but which the reader may have come across previously. One such construct is a "Markov blanket", which is a mathematical way of describing the boundary that separates the internal states of an organism from the external environment (although note that this term is sometimes used in different ways; see (Bruineberg, Dolega, Dewhurst, & Baltieri, 2020)). Another related construct is a "non-equilibrium steady state (NESS) density", which describes the states an organism must have a high probability of occupying if it is to

Thus, the focus of this tutorial is somewhat narrow and practical. Our aim is to equip the reader with the understanding and tools necessary to build models in practice and apply them in their own research.

The paper is organized as follows. In **Part 1**, we introduce the reader to the terms, concepts, and mathematical notation used within the active inference literature, and present the minimum mathematics necessary for a basic understanding of the formalism (as applied in a practical experimental setting). In **Part 2**, we introduce the reader to the concrete structure and elements of POMDPs and how they are solved. In **Part 3**, we provide a step-by-step description of how to build a generative model of a behavioral task (a variant on commonly used explore-exploit tasks), run simulations using this model, and interpret the outputs of those simulations. In **Part 4**, we introduce the reader to the neural process theory associated with active inference and walk the reader through generating and interpreting the outputs of neural simulations. In **Part 5**, we introduce the reader to learning processes in active inference. In **Part 6**, we introduce hierarchical models and illustrate how they can be used to simulate established electrophysiological responses in a commonly used auditory oddball paradigm. Finally, in **Part 7** we describe how to fit behavioral data to a model and derive individual-level parameter estimates and how they can be used for further group-level analyses.

1. Basic Terminology, Concepts, and Mathematics

The active inference framework is based on the premise that perception and learning can be understood as minimizing a quantity known as **variational free energy** (VFE), and that action selection, planning, and decision-making can be understood as minimizing **expected free energy** (EFE), which quantifies the VFE of various actions based on expected future outcomes. To motivate the use and derivation of these quantities, we need to first introduce the reader to Bayesian inference and explore its relation to the notion of active inference.

1.1 Mathematical Foundations: Bayes' Theorem and Active Inference

As an initial note to readers with less mathematical background, a full understanding of the equations presented below will not be necessary to begin building models and applying them to behavioral data. Often, building and working with models in practice is a great way to get an intuitive grasp of the underlying mathematics. So, if some of the equations below have unfamiliar notation and become hard to follow, do not get discouraged. An intuitive grasp of the concepts described in this section will be enough to learn the practical applications in the subsequent sections. That said, we also explain the equations and notation in this section assuming minimal mathematical background.

maintain its existence – where this can be understood as maintaining the integrity of its Markov blanket (i.e., keeping the boundary intact that separates an organism from its environment). The POMDP scheme described in this tutorial does not explicitly appeal to these constructs; however, one can think of an agent's preferred observations in POMDPs as those that keep it within the high-probability states consistent with its continued existence (i.e., those that would keep its Markov blanket intact).

We start by highlighting that the term 'active inference' is based on two concepts. The first is the idea that organisms **actively** engage with (e.g., move around in) their environments to gather information, seek out 'preferred' observations (e.g., food, water, shelter, social support, etc.), and avoid non-preferred observations (e.g., tissue damage, hunger, thirst, social rejection, etc.). The second concept is **Bayesian inference**, a statistical procedure that describes the optimal way to update one's beliefs (understood as probability distributions) when making new observations (i.e., receiving new sensory input). Specifically, beliefs are updated in light of new observations using **Bayes' theorem**, which can be written as follows:

$$p(b|o,m) = \frac{p(o|b,m)p(b|m)}{p(o|m)}$$

Starting on the right-hand side of the equation, the term $p(b|m)$ indicates the probability (p) of different possible beliefs (b) under a model of the world (m). This 'prior belief' (the **'prior'**) encodes a probability distribution ('Bayesian belief') before making a new observation (o). For example, b might refer to beliefs about the probability of an object having different possible shapes, such as a square vs. a circle vs. a triangle, and so forth. The term $p(o|b,m)$ is the **'likelihood'** term and encodes the probability within a model that one *would* make a particular observation *if* some belief were true (e.g., observing a straight line is consistent with a square shape but not with a circular shape). The symbol (|) means 'conditional on' and is also often read as 'given' (e.g., the probability of *o given b*). The term $p(o|m)$ is the '**model evidence**' (also called the '**marginal likelihood**') and indicates how consistent an observation is with a model of the world in general (i.e., across all possible beliefs). Finally, the term $p(b|o,m)$ is the **'posterior'** belief, which encodes what one's new belief (i.e., adjusted probability distribution over possible beliefs) optimally *should be* after making a new observation.

In essence, Bayes rule describes how to optimally update one's beliefs in light of new data. Specifically, to arrive at a new belief (your posterior), you must: 1) take what you previously believed (your prior), 2) combine it with what you believe about how consistent a new observation is with different possible beliefs (your likelihood), and 3) consider the overall consistency of that observation with your model (i.e., how likely that observation is under *any* set of possible beliefs included in your model; the model evidence, $p(o|m)$). The last step ensures that your posterior belief remains a proper probability distribution that sums to 1 (i.e., normalization). For a simple numerical example of Bayesian inference in the context of perception see **Figure 1**.

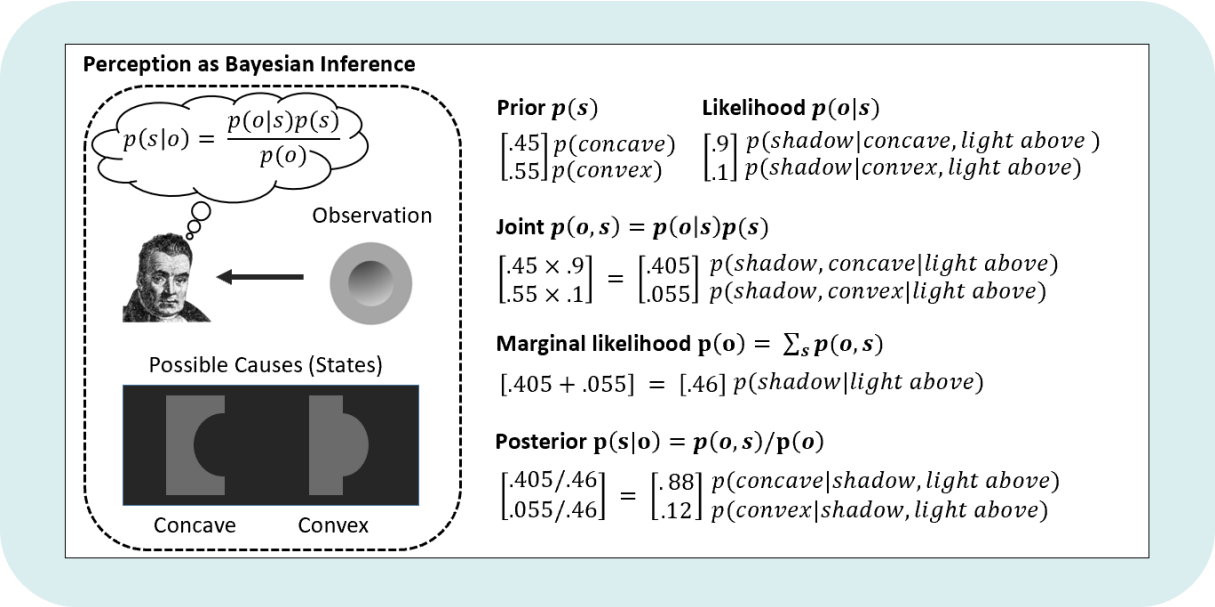

Figure 1. Simple example of perception as Bayesian Inference (based on (Ramachandran, 1988)). Please note that, while we have not explicitly conditioned on a model (m) in the expression of Bayes' rule shown in the left panel (as in the text), this should be understood as implicit (i.e., priors and likelihoods are always model-dependent). In this example, we take a 'brains eye view' and imagine that we are presented with the shaded gray disk (the 'observation') in the left panel of the figure. Due to the shading pattern, the central portion of the disk is typically perceived as concave (as shown in the bottom left), but it can also be perceived as convex (and typically is perceived as convex if rotated 180º). This is because the brain is equipped with a strong (unconscious) belief that light sources typically come from above. Given this assumption, the apparent shadow in the upper portion of the disk is much more likely to arise from a concave surface. To capture this mathematically, on the right we consider 'concave' and 'convex' as the two possible hidden states or 'causes' of sensory input (i.e., the shadow on the gray disk). We want to know whether the shadow pattern on the disk (observation) is caused by a concave or convex surface. The optimal way to infer the hidden state (concave or convex) is to use Bayes' theorem. For the sake of this example, assume we believe the chances of observing a concave vs. convex surface in general are almost equal, with a slight bias toward expecting a convex surface (i.e., encoded in the prior distribution shown above). The likelihood is a different story. The apparent shadow is much more consistent with a concave surface if light is coming from above (i.e., encoded in the likelihood distribution). To infer the posterior probability, we multiply the likelihood and prior probabilities, giving us the joint distribution. We then sum the probabilities in the joint distribution, yielding the total probability of the observation across the possible hidden states (the marginal likelihood). Finally, we divide the joint distribution by the marginal likelihood to reach the posterior. The posterior tells us that the most probable hidden state is a concave surface (corresponding to what is most often perceived). Thus, even though the two-dimensional gray disk alone is equally consistent with a convex or concave surface, the assumption that light is coming from above (encoded in the likelihood) most often leads us to perceive a concave 3-dimensional shape.

In this context, the concept of a model (m) is key. As briefly introduced above, we here focus specifically on a **generative model**, which is a model of how **observations** (sensory inputs) are generated by objects and events outside of the brain that cannot be known directly (typically termed '**hidden states**' or '**hidden causes**'; e.g., a baseball generating a specific retinal activation pattern). In simple generative models, these variables include sets of possible hidden states (s), priors over those states $p(s)$, sets of possible observations (o; often simply called '**outcomes**'), and a likelihood that specifies how states cause observations $p(o|s)$. The notion of hidden or unobservable states causing observable outcomes provides a nice perspective on **model inversion**. Namely, updating one's beliefs from prior to posterior beliefs is like inverting the likelihood mapping – that is, moving from $p(o|s)$ to $p(s|o)$. In other words, starting with a mapping from causes *to* consequences and then using it to infer the causes *from* consequences.

Importantly, models can include multiple types/sets of states and outcomes (i.e., different state and outcome spaces). For example, one set of states could encode possible shapes, while another set of states could encode possible object locations; and one set of possible observations could come from vision, while another set of possible observations could come from audition. If independent of one another, these sets of states and outcomes will be called different '**hidden state factors**' and different '**outcome modalities**'. Formally, the generative model is then defined in terms of the joint distribution over states and outcomes $p(o, s)$ – that is, the probability distribution over all possible combinations of states and outcomes. Based on the product rule in probability theory, this can be decomposed into the separate terms just mentioned:

$$p(o, s) = p(o|s)p(s)$$

If there is only one set of states and outcomes, this joint distribution is a 2-dimensional distribution. If there are more sets, it becomes a higher-dimensional distribution that, while harder to visualize (and more time consuming to compute), can be treated in the same way.

Although Bayesian inference is used to model perception and learning in related frameworks (e.g., predictive coding; see (Bogacz, 2017)), active inference extends this in two ways. First, it models categorical inference (e.g., the presence of a cat vs. a dog), as opposed to continuous inference (i.e., variables that take a continuous range of values, such as speed, direction of motion, contrast, etc.). Second, it models the inference of optimal action sequences during decision-making (i.e., inferring a probability distribution over possible action options, which can be thought of as encoding the estimated probability of achieving one's goals if one chose each action). In planning, possible sequences of actions (called '**policies**') are denoted by the Greek letter pi (π), so the generative model is extended to:

$$p(o, s, \pi) = p(o|s, \pi)p(s|\pi)p(\pi)$$

We will return to the prior over policies $p(\pi)$ later. For now, we simply note that active inference models can include additional elements that control (for example) how much randomness is present

in decision-making and how habits can be acquired and influence decisions. They can also be extended to include learning. We will return to these extensions in later sections.

Thus far, we have covered how a generative model can incorporate beliefs about the observations, states, and policies that a simulated decision-maker (commonly referred to as an '**agent**') uses to select actions. However, to make decisions, an agent also requires a means of assigning higher value to one policy over another. This in turn requires that some states and/or observations are preferred over others. One of the more (superficially) counterintuitive aspects of active inference is the way it formalizes preferences. This is because there are no additional variables labelled as 'rewards' or 'values'. Instead, preferences are cast as a specific type of prior probability distribution, $p(o)$. This '**prior preference distribution**' is most often used to encode how preferred some observations are over others. Note that it can also be formulated to encode preferences for states, but we will focus here on preferences for observations. The value of a policy is in turn specified as a probability distribution encoding the likelihood of selecting one policy over others. This probability is based on how likely each policy is to generate the most preferred observations – formally, the observations that provide the most evidence for the model (i.e., the observations most strongly 'expected' by the model).

In one sense, this can simply be considered a kind of mathematical 'trick' to bring preferences and policy values fully within the domain of Bayesian belief updating—a kind of **planning as inference** (H Attias, 2003; Botvinick & Toussaint, 2012; Kaplan & Friston, 2018). This formal treatment of preferences and values as 'Bayesian beliefs' (i.e., probability distributions) in the mathematics need not necessarily be reified psychologically, nor must it be if one wishes to use active inference models in practice. In other words, not all beliefs at the mathematical level of description need to be equated with psychological-level beliefs; some Bayesian beliefs in the formalism can instead correspond to rewarding or desired outcomes at the psychological level. However, many articles have considered the possibility (or use language suggesting) that the formalism may have deeper implications. Specifically, the active inference literature often discusses how prior preferences may be thought of as encoding the observations that are implicitly 'expected' by an organism in virtue of its phenotype (i.e., the observations an organism must seek out to maintain its survival and/or reproduction). For example, consider body temperature. Humans can only survive if body temperatures continue to be observed within the range of 36.5 – 37.5 degrees Celsius. Thus, the human phenotype implicitly entails a high prior probability of making such observations. If a human agent perceives (i.e., infers) that body temperature has (or is going to) deviate from 'expected' temperatures, it will infer which policies are most likely to minimize this deviation (e.g., seek shelter when it is cold). In this sense, body temperatures within survivable ranges (for the human phenotype) are the least 'surprising'. However, it is important to stress that this is not equivalent to the conscious experience of surprise; minimizing the type of 'phenotypic surprise' discussed in active inference is better mapped onto psychological states associated with achieving desired or rewarding observations. Traditional beliefs (in the psychological sense) can instead be identified with priors/posteriors over states $p(s)$ and with the observations expected given policies, $p(o|\pi)$. Regardless of how one views the meaning of the formal identification of preferences with expectations, active inference can more broadly (and less controversially) be seen as suggesting that the brain just *is* (or that it 'implements' or 'entails') a generative model of the body and external environment of the organism.

Having outlined the caveats above, we now expand on how preferences (as prior beliefs) drive policy selection. Deviations from prior preferences are formalized in terms of an information-theoretic quantity known as self-information or 'surprisal' (often also just called 'surprise'; we avoid this term here to minimize confusion with psychological surprise). Surprisal is most commonly written as the negative log probability of an outcome or observation $-\ln p(o)$, which can be thought of in physical terms as a potential energy that systems generally minimize. Consistent with the intuitive notion of surprise, lower probability events generate higher surprisal values (e.g., $-\ln(.5) = 0.69, \text{while } -\ln(.9) = 0.1$). Since an individual's preferences are modelled here as having a high prior probability, maximizing preferences corresponds to minimizing surprisal in both perception and action selection.

Surprisal plays a central role in furnishing a normative model of action and perception. We have just seen how minimizing surprisal maximizes prior preferences; however, there are at least two complementary perspectives on minimizing surprisal. For example, the average surprisal (i.e., self-information) in information theory is known as entropy. This means that minimizing surprisal also minimizes the entropy or dispersion of sensory outcomes, which – from a physiological perspective – can be thought of as homeostasis (i.e., minimizing variability within internally sensed bodily states, and thus keeping them within homeostatic ranges). Another view considers surprisal as the negative log marginal likelihood or **model evidence**. On this reading, minimizing surprisal is equivalent to maximizing the evidence for one's model of the world. In philosophy, this is sometimes interpreted as a kind of self-evidencing (Hohwy, 2016).

To maximize preferred outcomes, agents must maintain a generative model of the hidden causes of observations that is sufficiently accurate to anticipate and avoid observations that generate high surprisal. As we have seen, the optimal procedure for updating beliefs in light of new observations is Bayes theorem. However, for anything but the simplest distributions, Bayes theorem is computationally intractable. This is because evaluating $p(o)$ – the marginal likelihood (denominator) in Bayes' theorem and the basis of surprisal – requires us to sum the probabilities over all the states in the generative model (i.e., based on the sum rule of probability; **Figure 1**). For discrete distributions, as the number of dimensions (and possible values) increases, the number of terms that have to be summed increases exponentially. In the case of continuous distributions, it requires the evaluation of integrals that have no closed-form (analytic) solutions. Instead, approximation techniques are required to solve this problem. This is where VFE is crucial, as it is a computationally tractable quantity that provides an upper bound on surprisal (i.e., the negative log of the marginal likelihood). That is, minimizing VFE allows agents to maximize the posterior probability of states, by minimizing a quantity that is always greater than or equal to surprisal (and thereby maximizing model evidence). The next two sections flesh this out in more detail (also see **Figure 2**) and extend the idea of approximate inference to action selection.

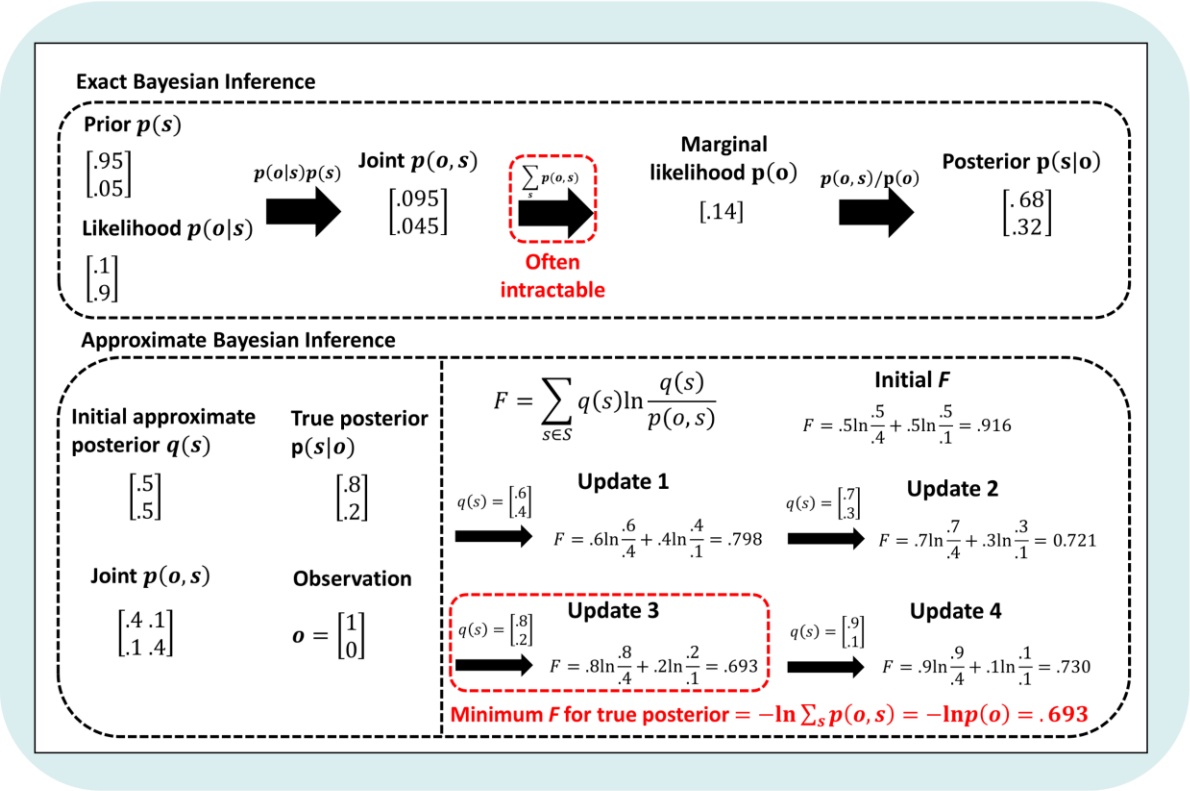

Figure 2. Simple example of exact versus approximate Bayesian Inference. Exact inference requires the evaluation of the marginal likelihood, which, for anything but the simplest distributions, is either computationally intensive or intractable. Instead, variational inference minimizes VFE (here denoted by F), which scores the difference between an (initially arbitrary) posterior distribution and a target distribution (here the exact posterior; for an introduction to variational inference, see **supplementary materials**). By iteratively updating the approximate posterior to minimize F (usually via gradient descent, see main text for details), a distribution can be found that approximates the exact posterior. When a minimum is found for F, the associated $q(s)$ will approximate the true posterior. Here, we have shown an example of iterative updating for the simplest distribution possible to illustrate the concept. As shown in the bottom left, in this example we start the agent with an (initially arbitrary) approximate posterior distribution $q(s) = [.5\ .5]^T$ – which can be thought of as an initial guess about what the true posterior belief $p(s|o)$ should be after a new observation (o). We then define a generative model with the joint probability, $p(o,s)$, where the true posterior we wish to find is $p(s|o) = [.8\ .2]^T$ given the observation $o = [1\ 0]^T$. On the bottom right, under 'Initial F', we first solve for F using our initial $q(s)$. Under 'Update 1', we then find (by searching neighboring values) a nearby value for $q(s)$ that leads to a lower value for F, and we repeat this process in 'Update 2'. In 'Update 3', $q(s)$ is equal to $p(s|o)$, which corresponds to finding a minimum value for F (i.e., where the remaining value above zero corresponds to surprisal, $-\ln p(o)$). The fact that F has reached a minimum can be seen in 'Update 4', where continuing to change $q(s)$ causes F to again increase in value. Thus, by finding the posterior $q(s)$ over states that generates the minimum for F, that $q(s)$ will

also best approximate the true posterior. In **supplementary code** we have included a script **VFE_calculation_example.m** that will allow you to define your own priors, likelihoods, and observations and calculate F for different $q(s)$ values.

1.2 Non-Technical Introduction to Solving Partially Observable Markov Decision Processes via Free Energy Minimization

The specific type of generative model used here is a **partially observable Markov decision process** (POMDP). A Markov decision process describes beliefs about abstract states the world, how they are expected to change over time, and how actions are selected to seek out preferred outcomes or rewards based on beliefs about states. Here, a 'state' is an abstract term, which can refer to any kind of possible belief content (e.g., the belief that one is at school, the belief that one sees a tree, the belief that one feels sad, etc.). This class of models assumes the 'Markov property', which simply means that beliefs about the current state of the world are all that matter for an agent when deciding which actions to take (i.e., that all knowledge about past states is implicitly 'packed into' beliefs about the current state). The agent then uses its model, combined with beliefs about the current state, to select actions by making predictions about possible future states. To be 'partially observable' means that the agent can be uncertain in its beliefs about the state of the world it is in. In this case, states are referred to as 'hidden'. The agent must infer how likely it is to be in one hidden state or another based on observations (i.e., sensory input) and use this information to select actions.

In active inference, these tasks are solved using a form of approximate inference known as **variational inference** ((Hagai Attias, 2000; Beal, 2003; Parr, Markovic, Kiebel, & Friston, 2019); for a brief introduction see **supplementary materials**). Broadly speaking, the idea behind variational inference is to convert the intractable sum or integral required to perform model inversion into an optimization problem that can be solved in a computationally efficient manner. This is accomplished by introducing an approximate posterior distribution over states (denoted $q(s)$) that makes simplifying assumptions about the nature of the true posterior distribution. For example, it is common to assume that hidden states under the approximate distribution do not interact (i.e., are independent), which gives the approximate distribution a much simpler mathematical form. Such assumptions are often violated, but the approximation is usually good enough in practice. The next step in variational inference is to measure the similarity between the approximate distribution, $q(s)$, and the generative model, $p(o, s)$, using a measure called the **Kullback–Leibler (KL) divergence**. We will discuss the KL divergence in more detail in a later section. For now, it is sufficient to think of the KL divergence as a measure of the dissimilarity between two distributions. It is zero when the distributions match, and it gets larger the more dissimilar the distributions become. VFE corresponds to the surprisal we want to minimize plus the KL divergence between the approximate distribution and the posterior distribution. In variational inference, we systematically update $q(s)$ until we find the value that minimizes VFE, at which point $q(s)$ will approximate the true posterior, $p(s|o)$. In **Figure 2** we provide a simple example of calculating VFE under different values for $q(s)$ to provide the reader with an intuition for how this works (this example can also be reproduced and customized in the **VFE_calculation_example.m** script provided in **supplementary materials**). For

ease of calculation, this figure uses the following expression for VFE (note that $VFE = F$ when presented in equations):

$$F = \sum_{s \in S} q(s) \ln \frac{q(s)}{p(o, s)}$$

However, this expression does not make it obvious how minimizing VFE will lead $q(s)$ to approximate the true posterior. As discussed further in the technical sections, this can be seen more clearly by algebraically manipulating VFE into the following form, which is more often seen in the active inference literature:

$$F = E_{q(s)}[\ln \frac{q(s)}{p(s|o)}] - \ln p(o)$$

For details on how we move between different expressions for F, see the optional technical section (Section 1.4). Here the $E_{q(s)}$ term indicates that $q(s)[\ln \frac{q(s)}{p(s|o)}]$ is evaluated for each value of $q(s)$ and then the resulting values are summed (i.e., equivalent to the $\sum_{s \in S} q(s)$ term in the previous expression). Based on this form of the equation, we can see that, because $\ln p(o)$ does not depend on $q(s)$, the value of F will become smaller as the value of $q(s)$ approaches the value of the true posterior, $p(s|o)$ – since the former is divided by the latter and the log of one is zero.

Within the active inference framework, the task of both perception and learning is to minimize VFE in order to find (approximately) optimal posterior beliefs after each new observation. Perception corresponds to posterior state inference after each new observation, while learning corresponds to more slowly updating the priors and likelihood distributions in the model over many observations (which facilitates more accurate state inference in the long run). It is important to note, however, that minimizing VFE is not simply a process of finding the best fitting approximation on every trial. Sensory input is inherently noisy, and simply finding the best fitting posterior on each trial would lead to fitting noise, which would result in exaggerated and metabolically costly updates. In statistics, this is known as overfitting. Fortunately, VFE minimization naturally avoids this problem. Put into words, VFE measures the *complexity* of a model minus the *accuracy* of that model. Here, the term 'accuracy' refers to how well a model's beliefs predict sensory input (i.e., the goodness of fit), while the term 'complexity' refers to how much beliefs need to change to maintain high accuracy when new sensory input is received (i.e., VFE remains higher if beliefs need to change a lot to account for new sensory input). Perception therefore seeks to find the most parsimonious (smallest necessary) changes in beliefs about the causes of sensory input that can adequately explain that input.

Analogously, the task of action selection and planning is to select policies that will bring about future observations that minimize VFE. The problem is, of course, that future outcomes have not yet been observed. Actions must therefore be selected such that they minimize *expected* free energy (EFE). Crucially, EFE scores the expected cost (i.e., a lower value indicates higher reward) minus the expected information gain of an action. This means that decisions that minimize EFE seek to both maximize reward and resolve uncertainty. When beliefs about states are very imprecise or uncertain,

actions will tend to be information-seeking. Conversely, selected actions will tend to be reward-seeking when confidence in beliefs about states is high (i.e., when there is no more uncertainty to resolve and the agent is confident about what to do to bring about preferred outcomes).

However, as we will see later, if the magnitude of expected reward is sufficiently high, actions that minimize EFE can become 'risky' – in that they seek out reward in the absence of sufficient information (i.e., reward value outweighs information value). In general, the imperative to minimize EFE is especially powerful in accounting for commonly observed behaviors in which, instead of seeking immediate reward, organisms first gather information and then maximize reward when they are confident about states of the world (e.g., turning on a light before trying to find food). It can also capture interesting behaviors that occur in the absence of opportunities for reward, where organisms appear to act simply out of 'curiosity' (Barto, Mirolli, & Baldassarre, 2013; Oudeyer & Kaplan, 2007; Schmidhuber, 2006). Further, variations in the level of expected reward during EFE minimization can capture interesting individual differences in behavior, as exemplified in the example of 'risky' behaviour just mentioned. Note that this crucial aspect of active inference effectively dissolves the 'explore-exploit dilemma' (discussed further below), because the imperatives for exploration (information seeking) and exploitation (reward seeking) are just two aspects of expected free energy, and whether exploratory or exploitative behaviors are favored in a given situation depends on current levels of uncertainty and the level of expected reward.

As we shall see in later sections, the posterior over policies is informed by both VFE and EFE. For now, we simply note that this is because VFE is a measure of the free energy of the present (and implicitly the past), whilst EFE is a measure of the free energy of the future. This is important, because while some policies may lead to a minimization of free energy in the future, they may not have led to a minimization of free energy in the past (and are therefore suboptimal policies when evaluated overall). In other words, EFE scores the likelihood of pursuing (i.e., the value assigned to) a particular course of action based upon future outcomes, while VFE reflects the likelihood of (i.e., the value assigned to) a course of action based upon past outcomes. This means the posterior distribution over policies is a function of both VFE and EFE, where these quantities respectively furnish retrospective and prospective action policy evaluations. At the psychological level, one can intuitively think of VFE as asking "how good has this action plan turned out so far?", while EFE asks "how good do I expect things to go if I continue to follow this action plan?".

When viewed from the perspective of potential neuroscientific applications, another crucial benefit of VFE and EFE is that they can be computed in a biologically plausible manner. This has inspired **neural process theories** that specify ways in which sets of neuron-like nodes (e.g., neuronal populations), with particular patterns of synapse-like connection strengths, can implement perception, learning, and decision-making through the minimization of these quantities. These neural process theories postulate several neuronal populations whose activity represents:

1) **Categorical probability distributions** (a special case of the multinomial distribution) over the possible states of the world. For those without background in probability theory, these distributions assign one probability value to each possible interpretation of sensory input, and all these probability values must add up to a value of 1. In other words, this

assumes the world must be in this state or that state, but not both at the same time, and assigns one probability value to the world being in each possible state. (Note: in other papers you may come across the notation $Cat(x)$, which simply indicates that a distribution x is a categorical distribution).

2) **Prediction-errors**, which signal the degree to which sensory input is inconsistent with current beliefs. Prediction errors drive the system to find new beliefs – that is, adjusted probability distributions – so that they are more consistent with sensory input, and therefore minimize these error signals.
3) Categorical probability distributions over possible **policies** the agent might choose.

These neural process theories also include simple, **coincidence-based learning mechanisms** that can be understood in terms of Hebbian synaptic plasticity (i.e., which involve adjusting the strength of the connections between two neurons when both neurons are activated simultaneously; (T. H. Brown, Zhao, & Leung, 2010)), and **message passing algorithms** (discussed in detail below) that model the connectivity and firing rate patterns of neuronal populations organized into cortical columns. Such theories afford precise quantitative predictions that can be tested using neuroimaging and other electrophysiological measures of neuronal activation during specific experimental paradigms.

1.3 The Generative Process vs. the Generative Model

One final concept that can be a source of confusion is the distinction between a **generative process** and a **generative model** (see **Figure 3**). A generative model, as discussed above, is constituted by *beliefs* about the world and can be inaccurate (sometimes referred to as 'fictive'). In other words, explanations for (i.e., beliefs about) how observations are generated do not have to represent a veridical account of how they are actually generated. Indeed, explanations for sensory data are generally simpler than the processes generating those data. In contrast, the generative process describes the veridical 'ground truth' about the causes of sensory input. For example, a model might hold the prior belief that the probability of seeing a pigeon vs. a hawk while at a city park is [.9 .1], whereas the true probability in the generative process may instead be [.7 .3]. This distinction is important in practical modelling when one wants to simulate behavior under false beliefs and unexpected observations (e.g., when modelling delusions or hallucinations).

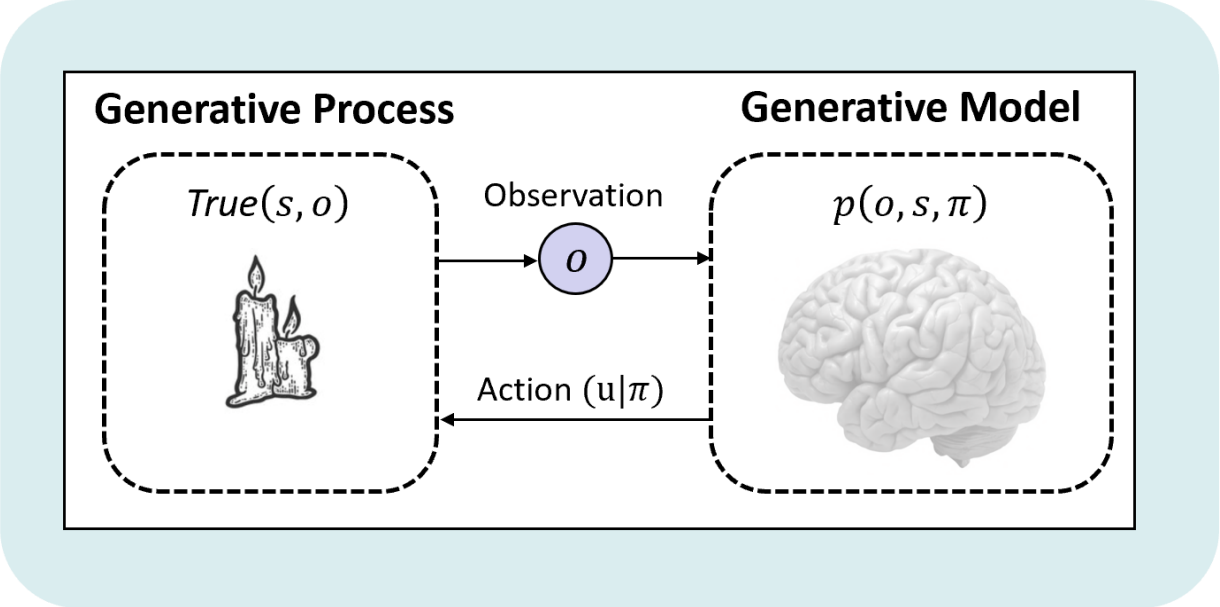

Figure 3. Visual depiction of the distinction between the generative process and the generative model, as well as their implicit coupling in the perception-action cycle. The generative process describes the true causes of the observations that are received by the generative model, which inform posterior beliefs about those causes. In turn, the generative model selects the policy (action or sequence of actions) with the highest posterior probability (given preferred outcomes), which couples the agent back to the generative process by changing the true state of world.

1.4 Technical Introduction to Variational Free energy and Expected Free Energy (optional)

In the previous subsections we introduced two quantities, *EFE* and *VFE*, which were described in largely qualitative terms. In this section, we consider the formal details behind the above descriptions. Readers without strong mathematical background can safely skip much of this section (if so desired) and move on to the next section without significant loss of understanding. That is, they should still be able to follow the rest of the paper. For those who choose to read this section, we provide accessible explanations of each equation in the hope that as many people as possible will be able to follow and learn from the material.

Before formally defining VFE and EFE, it will be helpful to first familiarize the reader with the relevant machinery, starting with the **KL divergence** (D_{KL}), or 'relative entropy'. As discussed in previous subsections, this is a measure of the similarity, or dissimilarity, between two distributions. The KL divergence between two distributions, $q(x)$ and $p(x)$, is written as follows (note that ln is the natural logarithm):

$$D_{KL}[q(x)||p(x)] = \sum_{x \in X} q(x) \ln \frac{q(x)}{p(x)}$$

This equation states that the KL divergence is found by taking each value of x in range X (for which p and q assign values), calculating the value of the right-hand quantity, and then summing the resulting values (for a concrete numerical example, see **Figure 2**). From the perspective of information theory, the KL divergence can be thought of as scoring the amount of information we would need to reconstruct $p(x)$ given full knowledge of $q(x)$. Note that 'information' is here measured in a quantity called *nats*, because we are working with the natural log, as opposed to log base 2 where information would be quantified in *bits*.

As mentioned above, because calculating model evidence is generally not possible, we instead minimize VFE, which is constructed to be an upper bound on negative (log) model evidence. As we noted above, this is also called 'surprisal' in information theory: $-\ln p(o)$. In other words, by minimizing VFE, one can minimize the negative model evidence – or, more intuitively, one can maximize model evidence (i.e., by finding beliefs in a model for which observations provide the most evidence).

Before turning to the definition of VFE, it will be helpful to clarify some notational conventions and concepts. Specifically, using the sum rule of probability we can express model evidence in terms our generative model as: $p(o) = \sum_{s,\pi} p(o,s,\pi)$. That is, $p(o)$ is the sum of the probabilities of observations for every combination of states and policies in the model. Next, one can multiply and divide the joint distribution, $p(o,s,\pi)$, by the (initially arbitrary) approximate posterior distribution $q(s|\pi)$. By definition, multiplying and then dividing by $q(s|\pi)$ does not change the value of this distribution. However, this trick ends up being quite useful as we will see. For mathematical convenience, one can take the negative logarithm of the resulting term, leading to:

$$-\ln p(o) = -\ln \sum_{s,\pi} \frac{p(o,s,\pi)q(s,\pi)}{q(s,\pi)} = -\ln E_{q(s,\pi)}\left[\frac{p(o,s,\pi)}{q(s,\pi)}\right]$$

Here, $E_{q(s,\pi)}$ denotes the expected value or 'expectation' of a distribution. This can be thought as a kind of weighted average, where each value of one distribution (here $q(s,\pi)$) is multiplied by the associated value in another distribution (here $\left[\frac{p(o,s,\pi)}{q(s,\pi)}\right]$), and each of the resulting values is summed to get the expected value of the latter distribution. Note that, although written as a summation, the calculation of F in **Figure 2** represents a numerical example. Readers should note the formal similarity between the KL divergence and the expectation value. This is no accident, as the KL divergence is simply the expected difference of the log of two distributions, where the expectation is taken with respect to the distribution in the numerator. Note that, because $\ln\frac{x}{y} = -\ln\frac{y}{x}$, the distributions in the numerator are often swapped around, as we have done below. In the active inference literature, it is very common to move between KL divergence notation and expectation notation. As such, it can be useful to keep the following identities in mind:

$$D_{KL}[q(x)||p(x)] = E_{q(x)}\ln\frac{q(x)}{p(x)} = \sum_{x\in X} q(x)\ln\frac{q(x)}{p(x)} = -\sum_{x\in X} q(x)\ln\frac{p(x)}{q(x)}$$

With a clear idea of expectation values and the KL divergence now in mind, we move onto the definition of VFE. Specifically, leveraging the mathematical result referred to as **Jensen's inequality** (Kuczma & Gilányi, 2009) – which states that the expectation of a logarithm is always less than or equal to the logarithm of an expectation – we arrive at the following inequality:

$$-\ln p(o) = -\ln E_{q(s,\pi)}\left[\frac{p(o,s,\pi)}{q(s,\pi)}\right] \leq -E_{q(s,\pi)}\left[\ln \frac{p(o,s,\pi)}{q(s,\pi)}\right] = F$$

On the right-hand side of this equation is VFE (always denoted by F in equations), which is defined in terms of the KL divergence – that is, expected difference of the respective logs – between the generative model $p(o,s,\pi)$ and the approximate posterior distribution $q(s,\pi)$. When the approximate posterior distribution and the generative model match, VFE is equal to zero (i.e., when $q = p,\ \ln\left(\frac{\mathrm{p(o,s,\pi)}}{\mathrm{q(s,\pi)}}\right) = 0$). The equality on the left-hand side of the equation holds because $q(s,\pi)$ is inside the logarithm and can therefore be cancelled, leaving only $p(o,s,\pi)$, which, as we saw above, can be converted into surprisal when summing over hidden states. From this, we can see that, by minimizing VFE, we minimize an upper bound on negative log model evidence. This means that VFE will always be greater than or equal to $-\ln p(o)$, which entails that by minimizing the value of VFE, the model evidence $p(o)$ will either increase or remain the same (i.e., it will be maximized, as the logarithm is a monotonically increasing function).

Therefore, all that is needed to perform approximate Bayesian inference in perception, learning, and decision-making is a tractable approach to finding the value of s (i.e., the approximate posterior distribution over s) that minimizes VFE. This can be accomplished by performing a **gradient descent** on VFE. Gradient descent is a technique that starts by picking some initial value (prior) for s and then calculates VFE for this value. It then calculates VFE for neighboring values of s and identifies the neighboring values for which VFE decreases most. It then samples from values of s that neighbor those values and continues to do so iteratively until a minimum VFE value is found (i.e., where VFE no longer decreases for any neighboring values). At this point, an approximation to the optimal beliefs for s have been found (given some set of observations o). On a final terminological note, because VFE is a function that is defined in terms of probability distributions, which are themselves functions, VFE is sometimes referred to as a '**functional**', which is simply the mathematical term for a function of a function.

Note that in active inference we calculate VFE with respect to each available policy individually (denoted by F_π). This is because different policies, through their impact on hidden states in the generative process, make certain observations more likely than others. For example, consider a situation where I believe there is a chair to my left and a table to my right. Conditional on having chosen to look left, it is more likely that I will observe a chair than a table. This means that observing the chair acts as evidence that I have chosen the policy of looking left. Because observations provide evidence for policies in this way, both the approximate posterior $q(s|\pi)$, and the generative model $p(o,s|\pi)$, are conditioned on policies. Going forward, VFE will also be presented in terms of F_π (as shown on the following page).

As we will discuss below, one way in which the brain may accomplish gradient descent on VFE during perception is through the minimization of prediction-error. The reason for this can be brought out by doing some algebraic shuffling to express VFE as a measure of **complexity** minus **accuracy** (i.e., as touched upon earlier):

$$
\begin{aligned}
F_\pi \quad &= E_{q(s|\pi)}\left[\ln \frac{q(s|\pi)}{p(o,s|\pi)}\right] \\
&= E_{q(s|\pi)}[\ln q(s|\pi) - \ln p(o,s|\pi)] \\
&= E_{q(s|\pi)}[\ln q(s|\pi) - \ln p(s|\pi)] - E_{q(s|\pi)}[\ln p(o|s,\pi)] \\
&= E_{q(s|\pi)}[\ln q(s|\pi) - \ln p(s|o,\pi)] - \ln p(o|\pi) \\
&= D_{KL}[q(s|\pi)||p(s|\pi)] - E_{q(s|\pi)}[\ln p(o|s)]
\end{aligned}
$$

The first line expresses VFE in terms of the expected log difference (KL divergence) between the approximate posterior and generative model. In the second line we use log algebra to express the division as a subtraction ($\ln \frac{x}{y} = \ln x - \ln y$). In the third line we use the product rule of probability ($p(o,s|\pi) = p(s|\pi)p(o|s,\pi)$) to take the likelihood term out of the first expectation term. The fourth line rearranges line 1 (again using the product rule: $p(o,s|\pi) = p(s|o,\pi)p(o|\pi)$) to show that VFE is always greater than surprisal (i.e., is an upper bound on surprisal) with respect to a policy (i.e., greater than $-\ln p(o|\pi)$). In machine learning, the sign of VFE is usually switched, so that it becomes an evidence lower bound, also known as an ELBO (Winn & Bishop, 2005).

The fifth line re-expresses the third line, but uses more compact notation for the first term and drops the dependency on policies in the second term (i.e., because the likelihood mapping does not depend on policies). The first term on the right-hand side is the KL divergence between prior and posterior beliefs. This value will be larger if one needs to make larger revisions to one's beliefs, which is the measure of complexity introduced earlier. A greater complexity means there is a greater chance of changing beliefs to explain random aspects of one's observations, which can reduce the future predictive power of a model (analogous to 'overfitting' in statistics). The second term on the right reflects predictive accuracy (i.e., the probability of observations given model beliefs about states). The brain will therefore minimize VFE if it minimizes prediction error (maximizing accuracy) while not changing beliefs more than necessary.

On a technical note, the gradients of VFE with respect to the approximate beliefs can always be expressed as a mixture of prediction errors. This is because complexity is the average difference between posterior and prior beliefs, while accuracy is the difference between predicted and observed outcomes. This licenses a description of active inference as **prediction error minimization** (Burr & Jones, 2016; A. Clark, 2017; Fabry, 2017; Hohwy, Paton, & Palmer, 2016), corresponding to the minimization of these two differences.

When inferring optimal actions, however, one cannot simply consider current observations. This is because actions are chosen to bring about preferred future observations. This means that, to infer optimal actions, a model must predict sequences of *future* states and observations for each possible policy, and then calculate the expected free energy (EFE) of those different sequences of future states and observations. As a model of decision-making, EFE also needs to be calculated relative to preferences for some sequences of observations over others (i.e., how rewarding or punishing they will be). In active inference, this is formally accomplished by equipping a model with (typically fixed) prior expectations over observations that play the role of preferences[2]. To see how this works, consider two possible policies that correspond to two different sequences of states and observations, where one sequence of observations is preferred more than the other. Since 'preferred' here formally translates to 'expected by the model', then the policy expected to produce preferred observations will be the one that maximizes the accuracy of the model (and hence minimizes EFE). This means that the probability (or value) of each policy can be inferred based on how much expected observations under a policy will maximize model accuracy (i.e., preferred observations). When preferred observations are treated as implicit expectations definitive of an organism's phenotype (e.g., those consistent with its survival, such as seeking warmth when cold, or water when thirsty) this has also been described as 'self-evidencing' (Hohwy, 2016). To score each possible policy in this way, EFE (denoted G_π in equations) can be expressed as follows:

$$
\begin{aligned}
G_\pi \quad &= E_{q(o,s|\pi)}[\ln q(s|\pi) - \ln p(o,s|\pi)] \\
&= E_{q(o,s|\pi)}[\ln q(s|\pi) - \ln p(s|o,\pi)] - E_{q(o|\pi)}[\ln p(o|\pi)] \\
&\approx E_{q(o,s|\pi)}[\ln q(s|\pi) - \ln q(s|o,\pi)] - E_{q(o|\pi)}[\ln p(o)] \\
&= -E_{q(o,s|\pi)}[\ln q(s|o,\pi) - \ln q(\mathrm{s}|\,\pi)] - E_{q(o|\pi)}[\ln p(o)] \\
&= D_{KL}[q(o|\pi)||p(o)] + E_{q(s|\pi)}[\mathrm{H}[p(o|s)]]
\end{aligned}
$$

The first line expresses EFE as the expected difference between the approximate posterior and generative model. Note that because EFE is calculated with respect to expected outcomes that (by definition) have not yet occurred, observations enter the expectation operator E_q as random variables. The second line uses the product rule of probability, $p(o,s|\pi) = p(s|o,\pi)p(o|\pi)$ to rearrange EFE into two terms that emphasize the inferential imperatives required to minimize EFE. In the third line we replace the true posterior ($\ln p(s|o,\pi)$) with an approximate posterior ($\ln q(s|o,\pi)$), and then drop the conditionalization on π in the second term (i.e., $E_{q(o|\pi)}[\ln p(o|\pi)] \rightarrow E_{q(o|\pi)}[\ln p(o)]$). This is a central move within active inference. Namely, $p(o)$ is used to encode preferred observations, and the agent seeks to find policies expected to produce those observations.

[2]Note that in some papers, preferences are formulated over states instead of outcomes. In this case, one might wonder how an agent can have two priors over states at the same time (one for beliefs and one for preferences). Although the technical details are beyond the scope of this paper, in this case one must think more explicitly in terms of an agent having two models - one of true states of the world and one of preferred states (cast as priors in each model, respectively). Policy selection then attempts to minimize the divergence between the two by bringing true states to match preferred states. For a derivation of the corresponding *EFE* expression, see **Appendix 1**.

The agent's preferences are independent of the policy being followed, which allows us to drop the conditionalization on π. The first term on the right-hand side of line 3 is the *epistemic value,* or the expected *information gain* of a state when it is conditioned on expected observations. The second term is *pragmatic value*, which, as just mentioned, scores the agent's preferences for particular observations. To make the intuition behind epistemic value more apparent, the fourth line flips the terms inside the first expectation so that it becomes prefixed with a negative sign (i.e., $p(x)[\ln p(x) - \ln q(x)] = -p(x)[\ln q(x) - \ln p(x)]$). Because the epistemic value term is now subtracted from the total, it is clear that to minimize EFE overall an agent must maximize the value of this term by selecting policies that take it into states that maximize the difference between prior and posterior beliefs; that is, maximize the difference between $\ln q(s|o,\pi)$ and $\ln q(s|\pi)$. In other words, the agent is driven to seek out observations that reduce uncertainty about hidden states (Parr & Friston, 2017a). For example, if an agent were in a dark room, the mapping between hidden states and observations would be entirely ambiguous, so it would be driven to maximize information gain by turning on a light before seeking out preferred observations (i.e., as it would be unclear how to bring about preferred outcomes before the light was turned on).

The final line shows the most common expression of EFE in the active inference literature (for a full description of how you get from line 1 to this final line, see **supplementary materials**). The first term on the right-hand side of this equation scores the anticipated difference (KL divergence) between 1) beliefs about the probability of some sequence of outcomes given a policy, and 2) preferred outcomes (i.e., those expected a priori within the model). This term is sometimes referred to as 'risk' (or expected complexity), but it can more intuitively be thought of as beliefs about the probability of reward for each choice one could make. That is, the lower the expected divergence between preferred outcomes and those expected under a policy, the higher the chances of attaining rewarding outcomes if one chose that policy. The second term one the right-hand side of the equation is the expected value of the **entropy** (H) of the likelihood function, where $\mathrm{H}\big(p(o|s)\big) = -\sum p(o|s)\ln[p(o|s)]$. Entropy is a measure of the dispersion of a distribution, where a flatter (lower precision) distribution has higher entropy. A higher-entropy likelihood means there are less precise predictions about outcomes given beliefs about the possible states of the world. This term is therefore commonly referred to as a measure of 'ambiguity' (or expected inaccuracy). Policies that minimize ambiguity will try to occupy states that are expected to generate the most precise (i.e., most informative) observations. Putting the risk and ambiguity terms together means that minimizing EFE will drive selection of policies that maximize both reward and information gain. Typically, seeking information will occur until the model is confident about how to achieve preferred outcomes, at which point it will choose reward-seeking actions. Importantly, as briefly mentioned earlier, the expression for EFE above entails that stronger (more precise) preferences for one outcome over others will have the effect of down-weighting the value of information, leading to reduced information seeking (and vice-versa if preferences are too weak or imprecise). This affects how optimally a model solves what is called the 'explore-exploit dilemma' (Addicott, Pearson, Sweitzer, Barack, & Platt, 2017; Friston, Lin, et al., 2017; Parr & Friston, 2017a; Schwartenbeck et al., 2019; Wilson, Geana, White, Ludvig, & Cohen, 2014) – that is, the difficult judgement of whether or not one "knows enough" to trust one's beliefs and act on them to seek reward or whether to first act to gather more information (see example simulations below). For a detailed description and step-by-step

derivation of the most common formulations of EFE found in the active inference literature, see **supplementary materials**.

2. Building and Solving POMDPs

2.1 Formal POMDP Structure

In this first subsection, we introduce the reader to the abstract structure and elements of an active inference POMDP, which is the standard modelling approach in active inference research at present. A warning: upon initial exposure, gaining a full understanding of this abstract structure can feel daunting. However, after we put together a model of a concrete behavioral task (in Section 3), this structure – and how to practically use it – tends to become much clearer. One further clarification on notation: vectors (i.e., single rows or columns of numbers) are denoted with italics, while matrices (i.e., multiple rows and columns of numbers) are denoted with bold (although, see footnote to **Table 2** for adjusted notation when indicating prior vs. posterior distributions).

The term POMDP denotes two major concepts. As described above, the first is partial observability, which means that observations may only provide probabilistic information about hidden states (e.g., observing green leaves could be evidence for both bushes or trees). The second is the Markov property, which simply means that, when making decisions, all relevant knowledge about distant past states is implicitly included within beliefs about the current state. This assumption can be violated, but it allows modelling to be more tractable and is 'good enough' in most cases. When dealing with violations of the Markovian assumption – such as when modelling memory – it is necessary to model several Markovian processes that evolve over different timescales. We will see an implicit example of this when we turn to learning the parameters of POMDP models in later sections.

A POMDP includes both trials and time points (τ) within each trial (sometimes called 'epochs'). At the first time point in a trial ($\tau = 1$), the model starts out with a prior over categorical states, $p(s_{\tau=1})$, encoded in a vector denoted by D – one value per possible state (e.g., the presence of a triangle vs. a square). When there are multiple state factors, there will be one D vector per factor. As touched on above, multiple state factors are necessary to account for multiple types of beliefs one can hold simultaneously (e.g., beliefs about an object's location and beliefs about its identity).

At each subsequent time point, the model has prior beliefs about how one state will evolve into another depending on the chosen policy, $p(s_{\tau+1}|s_\tau, \pi)$, encoded in a 'transition matrix' denoted by **B –** one column per state at t and one row per state at $\tau + 1$. There will be one **B** matrix per state factor, unless state transitions for a state factor are dependent on policies, in which case there will also be one **B** matrix per possible action (i.e., state transition under a policy) for that state factor (described further below).

The likelihood function, $p(o_\tau|s_\tau)$, is encoded in a matrix denoted by **A –** one column per state at τ and one row per possible observation at τ. When there are multiple outcome modalities, there will be one **A** matrix per outcome modality. As touched upon above, multiple outcome modalities are necessary

to account for parallel channels of sensory input (e.g., one for possible visual inputs and one for possible auditory inputs).

Preferred outcomes $p(o_\tau)$ are encoded in a matrix denoted by **C** – one column per time point and one row per possible observation. When there are multiple outcome modalities, there will be one **C** matrix per modality.

Prior beliefs about policies $p(\pi)$ are encoded in a (column) vector E – one row per policy. These priors can be thought of as encoding habits (i.e., actions that are more likely to be chosen independent of expectations about the future or current observations that would favor one policy over another).[3]

Each allowable action (u) is encoded as a possible state transition (one of several **B** matrices that can be chosen for a state factor). In this case, each possible action (**B** matrix) is encoded in a vector U, and the possible sequences of actions (where each allowable sequence defines a policy) are encoded in a matrix denoted by **V** (one row per time point, one column per policy, and a third dimension for state factor). Note that the possible actions encoded in the vector U are also sometimes referred to as '**control states**' in the active inference literature.

The free energy and expected free energy for each policy is denoted by vectors F and G, respectively. The degree to which G controls policy selection is modulated by a further parameter γ (a single number; i.e., a scalar). This parameter is a **precision estimate for the expected free energy** over policies. It can be thought of as encoding a prior belief about the confidence with which policies can be inferred (i.e., how reliable beliefs about the best policy are expected to be). It is often called the 'prior policy precision' parameter; however, it is important to note that this is not the same thing as the precision of posterior beliefs over policies ($\boldsymbol{\pi}$). This is because $\boldsymbol{\pi}$ also depends on the vectors E (habits) and F (see **Table 2**) – which means, for example, that $\boldsymbol{\pi}$ could be precise even if γ were low (Hesp et al., 2020). For this reason, it is better to think of γ as an 'expected free energy (G) precision' parameter as opposed to a policy precision parameter per se. If no habits are present (i.e., if E is a flat distribution), lower γ values lead to more randomness in policy selection. In the presence of strong habits, lower γ values increase how much habits influence policy selection, because the influence of G is reduced relative to E (see equation for $\boldsymbol{\pi}$ in **Table 2**). You can simulate these dynamics yourself by specifying values for γ, E, F, and G within the **EFE_Precision_Updating.m** code provided in **supplementary materials**.

When beliefs about policies are informed by outcomes in the past, the prior value for γ is updated into a posterior γ estimate (via updates to its hyperparameter β; see **Table 2** and further description below). This precision parameter is thus updated after each observation based on whether the variational and expected free energy of the policies increases or decreases after that observation, leading to reduced or increased expected precision values, respectively. However, it is important to note that there are situations in which policies are only considered from the current time step into

[3] Note here that the (fixed-form) prior beliefs about policies $p(\pi)$ encoded in the E vector are not the same as the prior or posterior beliefs over policies (π) that depend on E, F, G, and γ (see **Table 2**). Going forward, we denote the latter distributions using the symbol π (bolded for posteriors), omitting the $p(\pi)$ notation used to refer to the former (fixed-form) policy priors in E.

the future (such as 'shallow' or 'short-sighted' policies that, at each time step, only 'look ahead' one time step to consider the immediate consequences of different actions). In such cases, the past does not inform the relative probabilities of these policies (i.e., they are just 'reset' at each time step) – and the expected precision reduces to the prior value for γ (and is not updated). (Note: below, and in the **supplementary code**, we show how 'shallow', one-step policies vs. 'deep', multi-step policies can be included by specifying policies with the variable U vs. V).

All the model variables are summarized in **Table 1**. When a new outcome is observed (indicated by a value of 1 in one row within an outcome vector o_τ), the model then infers posterior distributions (bolded) over policies $\boldsymbol{\pi}$ and states $\mathbf{s}_{\pi,\tau}$ (given the choice of each policy). Solutions for inference in the POMDP are shown in **Table 2**.

Table 1. Model variables

Model variable	General definition	Model specification for explore-exploit task (described in Section 3)
o_τ	Observable outcomes at time τ.	Outcome modalities: 1. Hints (no hint, hint-left, hint-right) 2. Reward (start, lose, win) 3. Observed behavior (start, take hint, choose left, choose right)
s_τ	Hidden states at time τ. One vector of possible state values for each state factor (i.e., each independent set of states; e.g., visual vs. auditory states).	Hidden state factors: 1. Context (left machine is better vs. right machine is better) 2. Choices (start, take hint, choose left, choose right)
π	A vector encoding the distribution over action policies reflecting the predicted value of each policy. Each policy is a series of allowable actions in a vector U, where actions correspond to different state transitions (i.e., different **B** matrices) that can be chosen by the agent for each state factor. Policies are	Allowable policies include the decision to: 1. Stay in the start state 2. Get the hint and return to the start state 3. Get the hint and then do nothing 4. Get the hint and then either choose the left or right machine

	chosen by sampling from this distribution.	5. Immediately choose the left or right machine
A matrix $p(o_\tau \mid s_\tau)$	A matrix encoding beliefs about the relationship between hidden states and observable outcomes (i.e., the probability that specific outcomes will be observed given specific hidden states). When there is more than one outcome modality, there is one **A** matrix per outcome modality.	Encodes beliefs about the relationship between: 1. Probability that the hint is accurate in each context 2. Probability of reward in each context 3. Identity mapping between choice states and observed behavior
B matrix $p(s_{\tau+1} \mid s_\tau, \pi)$	A matrix encoding beliefs about how hidden states will evolve over time (transition probabilities). For states that are under the control of the agent, there are multiple **B** matrices, where each matrix corresponds to one action (state transition) that the agent may choose at a given time point (if consistent with an allowable policy). When there is more than one hidden state factor, there is one or more **B** matrices per state factor (depending on policies).	Encodes beliefs that: 1. Context does not change within a trial 2. Transitions from any choice state to any other are possible, depending on the policy.
C matrix $p(o_\tau)$	A matrix encoding the degree to which some observed outcomes are preferred over others (technically modeled as prior expectations over outcomes). When there is more than one outcome modality, there is one **C** matrix per outcome modality. Rows indicate possible observations; columns indicate time points.	Encodes the stronger preference for wins than losses. Wins are also more preferred at the second time point than the third time point.
D vector $p(s_1)$	A vector encoding beliefs about (a probability distribution over) initial hidden states. When there is more than one hidden state factor, there is one *D* vector per state factor.	The agent begins in an initial state of maximal uncertainty about the context state (prior to learning), but complete certainty that it will start in the 'start' choice state.

E vector $p(\pi)$	A distribution encoding beliefs about what actions will be chosen a priori (a prior probability distribution over policies, implemented as a vector assigning one value to each policy), based on the number of times different actions have been chosen in the past.	The agent has no initial habits to choose one slot machine or another (prior to learning).

Table 2. Matrix formulation of model update equations*

Model update component	Update equation	Model-specific description for explore-exploit task (described in Section 3)
Updating beliefs about initial states expected under each allowable policy	$\boldsymbol{s_{\pi,\tau=1}} = \sigma\left(\frac{1}{2}\left(\ln D + \ln \mathbf{B}^{\dagger}_{\boldsymbol{\pi},\boldsymbol{\tau}} s_{\pi,\tau+1}\right) + \ln \mathbf{A}^{\boldsymbol{T}} o_{\tau}\right)$	Updating beliefs about whether the left vs. right slot machine will win and beliefs about initial choice state (here, always the 'start' state). The term $\left(\ln D + \ln \mathbf{B}^{\dagger}_{\boldsymbol{\pi},\boldsymbol{\tau}} s_{\pi,\tau+1}\right)$ corresponds to prior beliefs in Bayesian inference, based on beliefs about the probability of initial states D and the probability of transitions to future states $\ln \mathbf{B}^{\dagger}_{\boldsymbol{\pi},\boldsymbol{\tau}} s_{\pi,\tau+1}$ under a policy. The term $\mathbf{A}^{\boldsymbol{T}} o_{\tau}$ corresponds to the likelihood term in Bayesian inference, evaluating how consistent observed outcomes are with each possible state. See main text for more details.
Updating beliefs about subsequent states expected under each allowable policy over time	$\boldsymbol{s_{\pi,\tau>1}} = \sigma\left(\frac{1}{2}\left(\ln \mathbf{B}_{\pi,\tau-1}\, s_{\pi,\tau-1} + \ln \mathbf{B}^{\dagger}_{\boldsymbol{\pi},\boldsymbol{\tau}} s_{\pi,\tau+1}\right) + \ln \mathbf{A}^{\boldsymbol{T}} o_{\tau}\right)$	Updating beliefs about whether the left vs. right slot machine will win and beliefs about choice states *after* the initial time point (here, depending on the choice to take the hint or select one of the slot machines). The term $\left(\ln \mathbf{B}_{\pi,\tau-1}\, s_{\pi,\tau-1} + \ln \mathbf{B}^{\dagger}_{\boldsymbol{\pi},\boldsymbol{\tau}} s_{\pi,\tau+1}\right)$ corresponds to prior beliefs in Bayesian inference, based on beliefs about the probability of transitions from past states $\ln \mathbf{B}_{\pi,\tau-1} s_{\pi,\tau-1}$ and the probability of transitions to future states

		$\ln\mathbf{B}^{\dagger}_{\boldsymbol{\pi},\boldsymbol{\tau}}s_{\pi,\tau+1}$ under a policy. The term $\ln\mathbf{A}^{\boldsymbol{T}}o_{\tau}$ corresponds to the likelihood term in Bayesian inference, evaluating how consistent observed outcomes are with each possible state. See main text for more details.
Probability of selecting each allowable policy	$\pi_0 = \sigma(\ln E - \gamma G)$ $\boldsymbol{\pi} = \sigma(\ln E - F - \gamma G)$	Prior distribution over policies (π_0), and posterior distribution over policies ($\boldsymbol{\pi}$). Actions are sampled from the posterior distribution over policies. The prior distribution is made up of the learned prior over policies encoded in the E vector (reflecting the number of times a policy has previously been chosen) and the expected free energy of each allowable policy (G). The posterior distribution is determined by E, the expected free energy of each allowable policy (G), and the variational free energy (F) under each policy. The influence of expected free energy is modulated by an expected precision term (γ), which effectively encodes prior confidence in expected free energy.
Expected free energy of each allowable policy	$\boldsymbol{G}_{\pi} = \sum_{\tau}\big(\mathbf{A}s_{\pi,\tau} \cdot \big(\ln \mathbf{A}s_{\pi,\tau} - \ln \mathbf{C}_{\tau}\big) - diag(\mathbf{A}^{T} \ln \mathbf{A}) \cdot s_{\pi,\tau}\big)$	This quantity evaluates the value of each policy based on beliefs about the ability of each policy to lead to the most desired outcomes (i.e., observing a win on the task). Achieving the most desired outcomes corresponds to minimizing the KL divergence between the observations expected given each policy ($\mathbf{A}s_{\pi,\tau} = o_{\pi,t}$) and preferred observations ($\mathbf{C}_{\boldsymbol{\tau}}$). Note that the $diag()$ function simply takes the diagonal elements of a matrix and places them in a row vector.

Free energy of each allowable policy	$$\boldsymbol{F}_{\pi} = \sum_{\tau} s_{\pi,\tau} \cdot \Big(\ln s_{\pi,\tau} - \frac{1}{2}\big(\ln \mathbf{B}_{\pi,\tau-1}\, s_{\pi,\tau-1} + \ln \mathbf{B}^{\dagger}_{\pi,\tau} s_{\pi,\tau+1}\big) - \ln \mathbf{A}^{\boldsymbol{T}}\, o_{\tau} \Big)$$	This quantity evaluates the evidence that inferred states provide for each policy based on new observations at each time point (e.g., the amount of surprise when observing a loss after choosing a specific slot machine). See the first two rows in this table on updating beliefs about states for an explanation of how each term in the equation relates to Bayesian inference.
Expected free energy precision	$$P(\gamma) = \Gamma(1, \beta)$$ $$\boldsymbol{\beta} = \beta + (\boldsymbol{\pi} - \pi_0) \cdot (-G)$$ $$\dot{\boldsymbol{\beta}} = \beta + (\boldsymbol{\pi} - \pi_0) \cdot (-G) - \boldsymbol{\beta}$$ $$\boldsymbol{\gamma} = 1/\boldsymbol{\beta}$$	The β term is a prior on the expected free energy precision term (γ). Specifically, it is the 'rate' parameter of a gamma distribution (Γ). The γ term controls the precision of G. It modulates the influence of G on policy selection, based upon the difference between the prior distribution over policies and the posterior distribution (see above). The difference between these terms reflects the extent to which new observations (scored by F) make policies more or less likely. If the vector encoding the posterior over policies increases in magnitude in comparison to the prior, and still points in the same direction, the difference vector between the posterior and the prior will point in the same direction as the $-G$ vector (i.e., less than a 90° angle apart; see **Figure 8**), thereby increasing the impact of G on policy selection. Alternatively, if the vector encoding the difference between the prior and posterior does not align with the vector encoding the negative free energy of each policy, the dot product will be large and negative, causing a negative beta update, and thereby reducing the impact of G on policy selection. Variational updates to beta ($\dot{\boldsymbol{\beta}}$), alter beta such that G contributes to the posterior over policies in an optimal manner. These updates are usually discussed in relation to dopamine and reward prediction errors in the

		reinforcement learning literature. Note that β is the initial prior (which is not updated), and $\boldsymbol{\beta}$ is the initial posterior, which is updated through variational updates ($\dot{\boldsymbol{\beta}}$) that converge on a new posterior estimate for $\boldsymbol{\gamma} = 1/\boldsymbol{\beta}$. For a derivation of these equations, see Appendix in (Sales, Friston, Jones, Pickering, & Moran, 2019).

***Note that $B^{\dagger}$ denotes the transpose of B with normalized columns (i.e., columns that sum to 1). Also note that you may commonly see the dot $(\cdot)$ notation used in the active inference literature to denote transposed matrix multiplication, such as $A \cdot s$, which means $A^{T}s$. The σ symbol indicates a softmax operation (for an introduction see Appendix 3) which allows vector values to make up a proper probability distribution. Bolded, non-italicized variables indicate matrices, while bolded, italicized variables indicate posterior values or distributions. Subscripts indicate conditional probabilities; e.g., $s_{\pi,\tau} = p(s_{\tau}|\pi)$.**

2.2 Graphical Models

In many papers in the active inference literature, POMDPs are represented using **graphical models**. Graphical models, such as the graphs shown in **Figures 4-6**, are a useful method for visually depicting how variables in a model depend on one another. When models include probability distributions over variables, these graphs can be used to represent the conditional relationships between variables. These types of probabilistic graphical models are particularly useful in the context of active inference because they provide a clear visual summary of the computational architecture of the models, and the way (biologically plausible) message passing algorithms (described below) can be used to update beliefs. Here we consider two types of graphical models – **Bayesian networks** (or '**Bayes nets**', see **Figure 4**; and for an introduction, see chapter 8 of (Bishop, 2006)) and **Forney-style (normal) factor graphs** (Dauwels, 2007; Loeliger, 2004). For readers interested in a more detailed introduction to the use of Forney-style factor graphs in active inference, we recommend the excellent tutorial introduction in the following (de Vries & Friston, 2017).

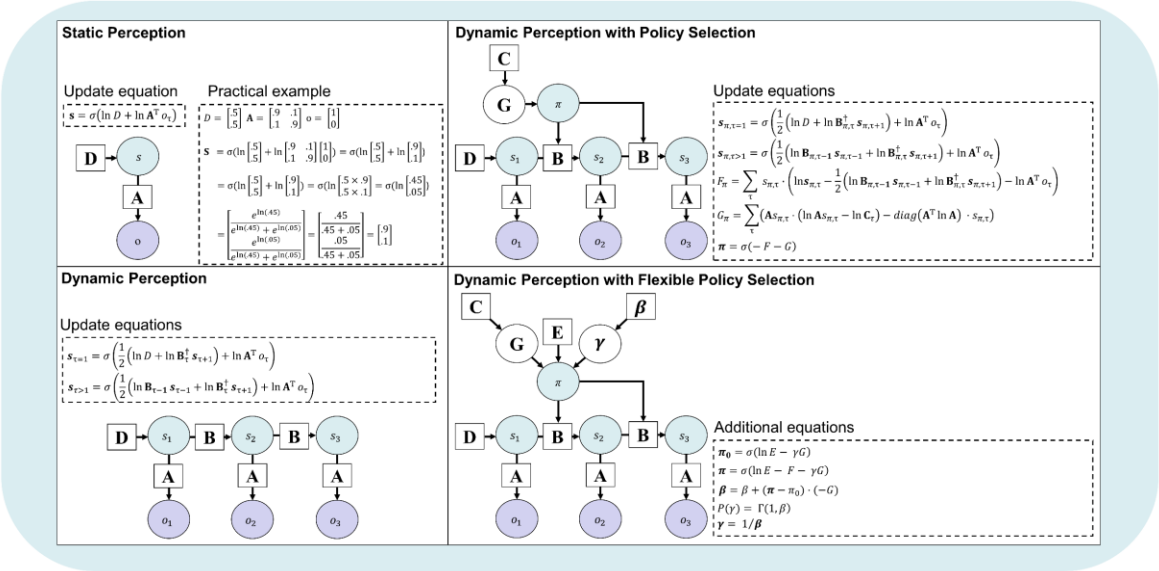

Figure 4. Bayesian network representations of state estimation (perception) and policy selection. Each graph depicts a generative model of the causes of observations, which can be inverted to perform inference. **Top left**. *Static perception* with a worked example. Variables: s = states, o = observations, **A** = likelihood mapping between states and outcomes, D = initial state priors. This is equivalent to exact Bayesian inference. For the worked example, note that in this case $\mathbf{A}^T = \mathbf{A}$ because the matrix is symmetric. **Bottom left**. *Dynamic perception*. Transition matrices (**B**) have been added to describe (beliefs about) how states change over time. Subscripts for observations and states correspond to time point in a trial (denoted by *tau*; τ). Importantly, when $\tau > 1$, the **B** matrix from the previous τ (i.e., $\tau - 1$) functions as an empirical prior, playing the same role as the D vector at $\tau = 1$. **Top right**. *Dynamic perception with policy selection*. Each policy (π) entails a different sequence of actions, which corresponds to different transitions between states (i.e., different **B** matrices). Based on VFE and EFE (which in turn depends on prior preferences, **C**), the highest probability will be assigned to policies expected to maximize the Bayesian model evidence for the entire graphical model (which will maximize the probability of preferred observations). **Bottom right**. *Dynamic perception with flexible policy selection*. This final model includes an expected precision term γ, which adjusts the contribution of **G** (expected free energy) to the posterior distribution over policies (according to the agent's confidence in policy selection, parameterized by β). A prior over policies (**E**) is also included, which can be used to model habit formation. See main text for further variable descriptions. See **Table 2** for further explanation of these equations.

When depicting active inference models with Bayes nets (**Figure 4**), the circles ('nodes') correspond to variables (e.g., observations, hidden states, and policies), while the arrows connecting nodes ('edges') show the dependencies between variables represented by each node. For example, the arrows in **Figure 4** going from the 's_τ node' (i.e., states at time τ) to the 'o_τ node' (i.e., outcomes at time τ) means that the value of o_τ depends on the value of s_τ. This entails that if one knows something

about observations, then one can infer something about the hidden states that cause them (i.e., the hidden states that generate those observations in the generative model).

Readers familiar with Bayesian networks will note that the form of the graphical models shown in **Figure 4** is slightly unusual, as squares denoting the factors that mediate the conditional relationships have been placed on top of the edges (e.g., the **A** and **B** matrices). To review the previous section, in this graphical model observations at each time step (purple) are generated by hidden states (green) via a mapping encoded by the likelihood matrix **A**. The **B** matrix determines state transitions, which function as prior beliefs about how hidden states will change from one time point to the next. The **D** vector serves as the prior for initial states. When state transitions are under the control of the agent (i.e., via action), the **B** matrix depends upon action policies (π). The probability distribution over policies depends upon learned priors over policies (E) and the EFE of each allowable policy (G). The EFE depends on a prior distribution over observations (**C**), which encodes the agent's preferences for some observations over others (i.e., this dependency entails that policies with the lowest EFE will be those expected to generate the most preferred observations). The influence of EFE is also modulated by the expected free energy precision term (γ), which encodes confidence in current EFE estimates. This is in turn influenced by the value of β (an initial prior over γ; see **Table 2** for a description). To help readers gain an intuition for inference in graphical models, **Figure 4** builds up a full POMDP starting from a graphical representation of perception at a single time point using Bayes rule (with a worked example). It then adds the evolution of states over time, followed by their dependence on policies, and the dependence of policies on the variables just described.

The defining characteristic of a generative model is that it can be used to generate data (i.e., observations). The conditional dependencies depicted in the Bayesian network in the bottom right of **Figure 4** show how observations are generated by a POMDP. Starting at the top of the network, a policy is first selected via a softmax function of the aforementioned variables: $\pi = \sigma(\ln E - \gamma G)$[4]. Next, policy-dependent transition probabilities encoded in the **B** matrix (or the D vector at $\tau = 1$) generate hidden states, which in turn generate observations at each time point. The likelihood (**A**) matrix determines which observations are generated by each hidden state.

Recall that to perform inference we must invert the generative model (i.e., infer the most likely states and policies given each new observation). This is where normal factor graphs (**Figure 5**) are crucial, as they can be leveraged to both derive and visualize a suite of message passing algorithms (see below) for Bayesian inference. Normal factor graphs represent a factorization of the generative model. Recall that generative models are formally defined as the joint probability distribution over observations, states, and policies of the POMDP across time, $p(o_{1:T}, s_{1:T}, \pi)$. **Factorization** means that this joint probability can be defined as the product of several conditionally independent distributions. In POMDPs, the factorization assumes that each state only depends on the state at the previous time step (i.e., the so-called Markov property). This is described by the following equation,

[4] Note that this is the prior distribution from which policies are selected. The posterior distribution over policies also incorporates *VFE:* $\pi = \sigma(\ln E - F - \gamma G)$.

which shows a factorization of the joint distribution into prior distributions over states and policies, and the distributions representing the likelihood and state transitions.

$$p(o_{1:T}, s_{1:T}, \pi) = p(s_1)p(\pi)\prod_{\tau}^{T} p(o_\tau|s_\tau)p(s_\tau|s_{\tau-1}, \pi)$$

For the unfamiliar reader, please note that the symbol $\prod_{\tau}^{T}(\cdot)$ indicates taking the product of each of the distributions to the right of it for each time point *tau* (τ) to the final time point T.

For reference, we note here that the distinction between τ and another variable for time (t) in active inference models is quite important, and a common source of confusion for many non-initiated readers of the active inference literature. We discuss this further below in a non-technical section, but here we briefly note that τ refers to the identity of a time point for which an agent has beliefs, while t refers to the time at which beliefs are being updated by new observations (e.g., allowing beliefs *about* earlier time points to be retrospectively updated by observations *at* later time points).

Also note that there is a direct correspondence between the factorized distribution shown in the equation above and the factors included in the normal factor graph in **Figure 5**. Specifically, $p(s_1) =$ **D,** $p(o_\tau|s_\tau) =$ **A,** $p(s_\tau|s_{\tau-1}, \pi) =$ **B,** and $p(\pi) = E$. Each of these distributions is associated with a factor node. Each edge represents the probability distribution over the variable that needs to be inferred (i.e., the marginal posteriors over states $s_{\pi,\tau}$, and policies π). Edges connect factors that exchange messages with the same variables (e.g., **D**, **A**, and **B** are all connected by the variable s_1). When a variable only exchanges messages with one factor (e.g., policies) the edge is a half edge (i.e., only connected on one end). Finally, observed variables are associated with small, filled squares.

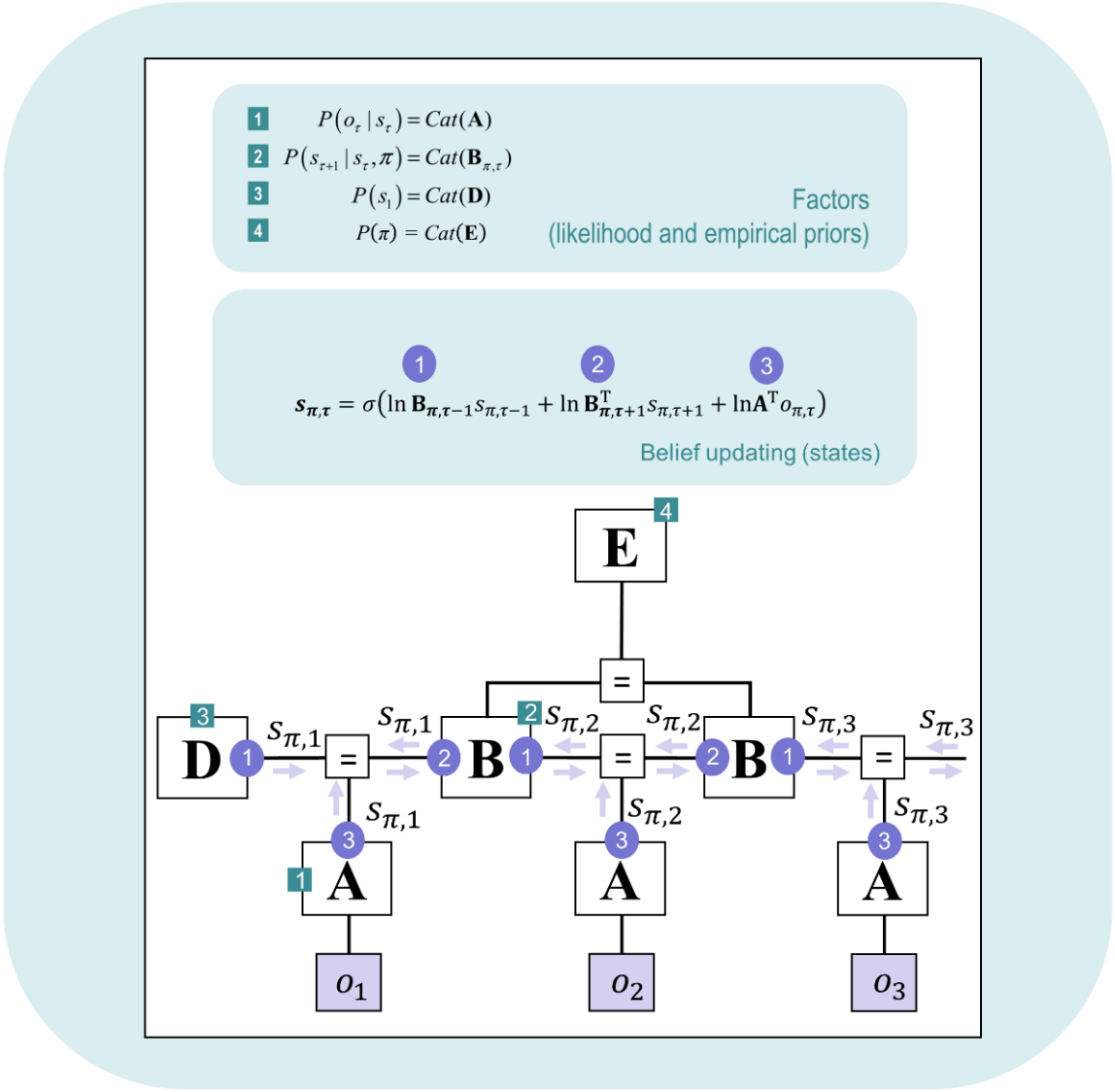

Figure 5. **Top.** Equations specifying the factors that constitute the factorized generative model. Numbers in the green squares highlight the correspondence between the equations and the factors in the generative model that are represented within the normal factor graph in the bottom panel. Here, $Cat()$ indicates a categorical distribution. **Middle.** Belief update equation for the marginal posterior over states derived from variational message passing. Purple numbers indicate the correspondence between terms within the update equations and the messages passed between each factor shown in the factor graph in the bottom panel. **Bottom.** Normal factor graph representation of the factorized POMDP. In contrast to the Bayes net representation shown in **Figure 4**, nodes (large white boxes) represent factors, whereas the edges (lines connecting each box) represent the sufficient statistics of the marginal posteriors that are passed as messages between factors (i.e., edges represent the common variables that participate in the factors they connect, such as posteriors over states under each policy for each time point, $s_{\pi,t}$). Adapted from (Friston, Parr, et al., 2017).

2.3 Technical Section on Variational and Marginal Message Passing (optional)

For the reader without strong mathematical background, this section can also be safely skipped. Although, as always, we have made efforts to fully explain all equations for unfamiliar readers.

To invert the model (i.e., condition on observations to infer *approximate* posteriors over states and policies) via the minimization of VFE, simplifying assumptions need to be made about the approximate posterior (recall that our estimate using VFE will be approximate, since exact inference is intractable in most real-word cases). Here we use the **mean field approximation**, which assumes that the approximate posterior factorizes into the product of (independent) marginal distributions; here, factorizing into a prior distribution over policies and the distributions over states expected under each policy at each time point (Bishop, 2006). (Here the limitation is that, if there are pairwise [or more complex] interactions between variables, they will be ignored by the mean field approximation). For the factorization to be exact, it must match the factorization of the true posterior. We write the approximate posterior as follows.

$$p(s_{1:T}|o_{1:T},\pi) \approx q(s_{1:T},\pi) \; = q(\pi)\prod_{\tau}^{T} q(s_\tau|\pi)$$

Note that, by convention, approximate posterior distributions are denoted with the variable q. Also again recall that T corresponds to the final time point in a trial, such that this posterior distribution is over the values of states across time points under each policy – and this distribution itself evolves over time. This means that an observation at a later time can change posterior beliefs about states at earlier times (i.e., retrospective inference).

With this factorization in hand, we can employ an algorithm called **variational message passing** to infer the marginal posteriors, $q(s_\tau|\pi)$, at each edge of the graph, and then combined into a global posterior $q(s_{1:T},\pi)$ using the equation just presented. Variational message passing can be summarized in terms of the following 5 steps:

1. Fix the value of observed variables (i.e., observations).
2. Choose an edge V corresponding to some hidden variable x_τ you want to infer (e.g., hidden states).
3. Initialize the values of:
    a. The approximate marginal posteriors for that variable $q(x_\tau)$, which take on the values of the posterior from the previous time step.
    b. The messages $\mu(x_\tau)$, which take on values set by each factor node.
4. Pass a message from each factor node N to each of its connecting edges V (written $\mu_{N\to V}$).
5. Update the marginal posterior represented by each edge according to the following rule, $q(x_\tau) \propto \vec{\mu}(x_\tau)\overleftarrow{\mu}(x_\tau)$.

Steps 4 to 5 are then repeated until the difference between updates converges to some acceptably low value. For unfamiliar readers, the '$\propto$' symbol denotes proportionality, meaning that the ratio between variables is always constant. We can change from the proportionality sign to an equals sign

by explicitly introducing a constant (k) into the equation, so $x \propto y$ becomes $x = k \cdot y$. For probability distributions, the constant is the normalization factor which ensures that a distribution sums to one.

Formally, each message conveys the exponentiated expected log value of each factor $\vec{\mu}(x_\tau) \propto \exp E_q[\ln g(x_\tau)]$ for unknown variables, where $g(x_\tau)$ denotes the function represented by each factor (Dauwels, 2007); for observed variables, the message simply conveys the known value of the factor, which can easily be calculated (i.e. when an observation is received, the value of the **A** matrix factor is simply $\mathbf{A}^{\boldsymbol{T}}$o). More intuitively, these messages constitute the sufficient statistics of the marginal posterior represented by the edge. That is, the information encoded by the messages is sufficient to construct the approximate posterior (below we will make this more concrete in relation to POMDPs). If a variable participates in more than one factor (e.g., as with hidden states) then the edges converge on an equality node denoted by '=', which means that copies of this same variable are associated with each edge. The posterior at each edge $q(x_\tau)$ is normalized by applying a softmax function. The normalized product of the messages arriving at each edge is then the approximate posterior associated with each edge. Note that, unlike the Bayes net, the normal factor graph has undirected edges, which highlights the bidirectional nature of message passing (see light purple arrows in the bottom portion of **Figure 5**). Using these update rules, we arrive at the following update equations for the marginal posteriors over states in our POMDP models, which we will call messages 1-3 (purple circles on the factor graph at each location in the figure where edges meet factors):

$$\text{Message 1: } \ln\vec{\mu_B}(s_\tau|\pi) = E_{q(s_{\tau-1}|\pi)}[\ln p(s_\tau|s_{\tau-1},\pi)]$$

$$\text{Message 2: } \ln\overleftarrow{\mu_B}(s_\tau|\pi) = E_{q(s_{\tau+1}|\pi)}[\ln p(s_\tau|s_{\tau+1},\pi)]$$

$$\text{Message 3: } \ln\mu_A(s_\tau|\pi) = \ln p(o_\tau|s_\tau)$$

Note the straightforward relation between these messages and Bayes' theorem in the factor graph in **Figure 5**. Message 1 corresponds to the prior from the previous time point, Message 2 corresponds to prior information from the future time point (e.g., allowing retrospective inference), and message 3 corresponds to the likelihood. So, for example, if we take the edge corresponding to the posterior for $s_{\pi,2}$ (in the middle of the graph), this posterior will then correspond to integrating priors (**B**) with likelihoods (**A**) and then normalizing to convert back to a proper probability distribution (i.e., as in Bayes' theorem).

Importantly, the active inference framework has also proposed a plausible mapping between message passing algorithms and message passing between neurons in the brain. In this proposed mapping, neuronal membrane dynamics (depolarization and resulting firing rates) represent the continually updated posteriors over variables (edges), while the patterns of synaptic connection strengths implement factors (which implement functions that transform the incoming messages; see (Parr & Friston, 2018a)). To make the relationship to neuronal dynamics more explicit, one can set up an ordinary differential equation, based on variational message passing, that performs a gradient descent on VFE by introducing an auxiliary variable encoding state prediction error ($\varepsilon_{\pi,\tau}$). This prediction error scores the difference between the log prior probability of each hidden state (the posterior from the previous timestep) and the log probability of each hidden state following a round

of message passing (i.e., when a new observation has been received). All of this is, of course, conditioned upon a specific policy (denoted by the subscript). We can then substitute in a 'depolarization' variable, $v_{\pi,\tau}$, to stand in for the log posterior over states; $v_{\pi,\tau} = \ln(s_{\pi,\tau})$. Adopting the matrix notation for the messages, this becomes:

$$\varepsilon_{\pi,\tau} = \ln\mathbf{B}_{\pi,\tau-1}s_{\pi,\tau-1} + \ln\mathbf{B}^{\boldsymbol{T}}_{\pi,\tau}s_{\pi,\tau+1} + \ln\mathbf{A}^{\boldsymbol{T}}o_{\pi,\tau} - \ln s_{\pi,\tau}$$

$$v_{\pi,\tau} = \ln(s_{\pi,\tau})$$

$$\dot{v}_{\pi,\tau} = \varepsilon_{\pi,\tau} = -\frac{\partial VFE_{\pi}}{\partial s_{\pi,\tau}}$$

$$s_{\pi,\tau} = \sigma(v_{\pi,\tau})$$

Note that the dot notation over the depolarization variable in the third equation ($\dot{v}_{\boldsymbol{\pi,\tau}}$) denotes the rate at which this variable changes at each time point (i.e., its temporal derivative). The key aspect of this differential equation is that the value of $\boldsymbol{s}_{\pi,\tau}$ (right-most term in the first equation) continues to change until the value of the state prediction error term $\varepsilon_{\pi,\tau}$ is minimized. In other words, the equation is set up such that it changes the value of $\boldsymbol{s}_{\pi,\tau}$ (in the direction of steepest descent) until it produces the lowest value of $\varepsilon_{\pi,\tau}$, at which point the resulting value of $\boldsymbol{s}_{\pi,\tau}$ will correspond to an approximate posterior over states. This is because $\varepsilon_{\pi,\tau} = 0$ is the attracting fixed point of the equation, meaning that the system tends to evolve towards $\varepsilon_{\boldsymbol{\pi,\tau}} = 0$, and that once it reaches this value it will remain there. This leaves us with a biologically plausible prediction-error minimization scheme that can perform posterior inference over states and can be instantiated in a relatively simple neural network (for more details, see (Parr & Friston, 2018a)).

Inferring policies makes use of an analogous process. Although it should be noted that the implementation of posterior inference over policies in the standard code does not make use of variational message passing. Still, we will stay with the message passing notation for didactic purposes. Recall that, under active inference, policies are selected based on their (expected) ability to generate preferred observations and maximize information gain. In addition, recall from the previous two sections that preferred observations are formally treated as being more probable *a priori*. Policies can then be thought of as beliefs about state transitions, where state transitions are more probable if they maximize the marginal probability of current observations, $\ln P(o_\tau|\pi)$, and the expected marginal probability of future observations conditioned upon the policy in question, $E_{q(o_\tau,s_\tau|\pi)}[\ln P(o_\tau|\pi)]$. Notice that this marginal likelihood of future observations is expressed in terms of an expectation value, which treats observations as random variables that need to be inferred (i.e., because they have not yet been given to the model). Also notice the similarity between this and the expression for EFE. This similarity is no accident, as we shall see shortly. Inferring these marginal distributions requires us to evaluate **partition functions** of the normal factor graph. This means summing over the variables represented by the edges enclosed in the red dotted lines in **Figure 6**. This operation is also sometimes called "closing the box" (Loeliger, 2004). For example, to obtain the marginal likelihood of current observations conditioned upon policies, and the expected future observations conditioned upon policies, we must evaluate the following summations:

$$\ln p(o_\tau|\pi) = \ln \sum_s p(o_\tau, s_\tau|\pi).$$

$$E_{q(o_\tau,s_\tau|\pi)}[\ln p(o_\tau|\pi)] = E_{q(o_\tau,s_\tau|\pi)}[\ln \sum_s p(o_\tau, s_\tau|\pi)]$$

As we have seen, however, such summations are often intractable. Instead, we evaluate the free energy functionals VFE and EFE, as they approximate the required marginal probabilities (as we saw in Section 1) and can be computed efficiently.

$$-\ln P(o_\tau|\pi) \approx VFE_{\pi,\tau}$$

$$-E_{q(o_\tau|\pi)}[\ln P(o_{\tau>t}|\pi)] \approx EFE_{\pi,\tau}$$

The posterior over policies can then be computed in a similar manner as the posterior over states. Specifically, we can express the messages sent from the **B** matrix factor nodes, and the E vector factor node, to the edges representing the posterior policies as follows (messages 1-3 shown in **Figure 6**).

$$q(\pi) \propto \mu_E(\pi) \cdot \vec{\mu_B}(\pi) \cdot \overleftarrow{\mu_B}(\pi)$$

$$\text{Message 1: } \ln \mu_E(\pi) = \ln E$$

$$\text{Message 2: } \ln \vec{\mu_B}(\pi) = VFE_{\pi,\tau}$$

$$\text{Message 3: } \ln \overleftarrow{\mu_B}(\pi) = EFE_{\pi,\tau}$$

Finally, if we normalize the messages by passing them through a softmax function, we arrive at an expression for the posterior over policies that (suppressing the precision term γ) corresponds to the equation shown earlier.

$$\boldsymbol{\pi} = \sigma(\ln E - F - G)$$

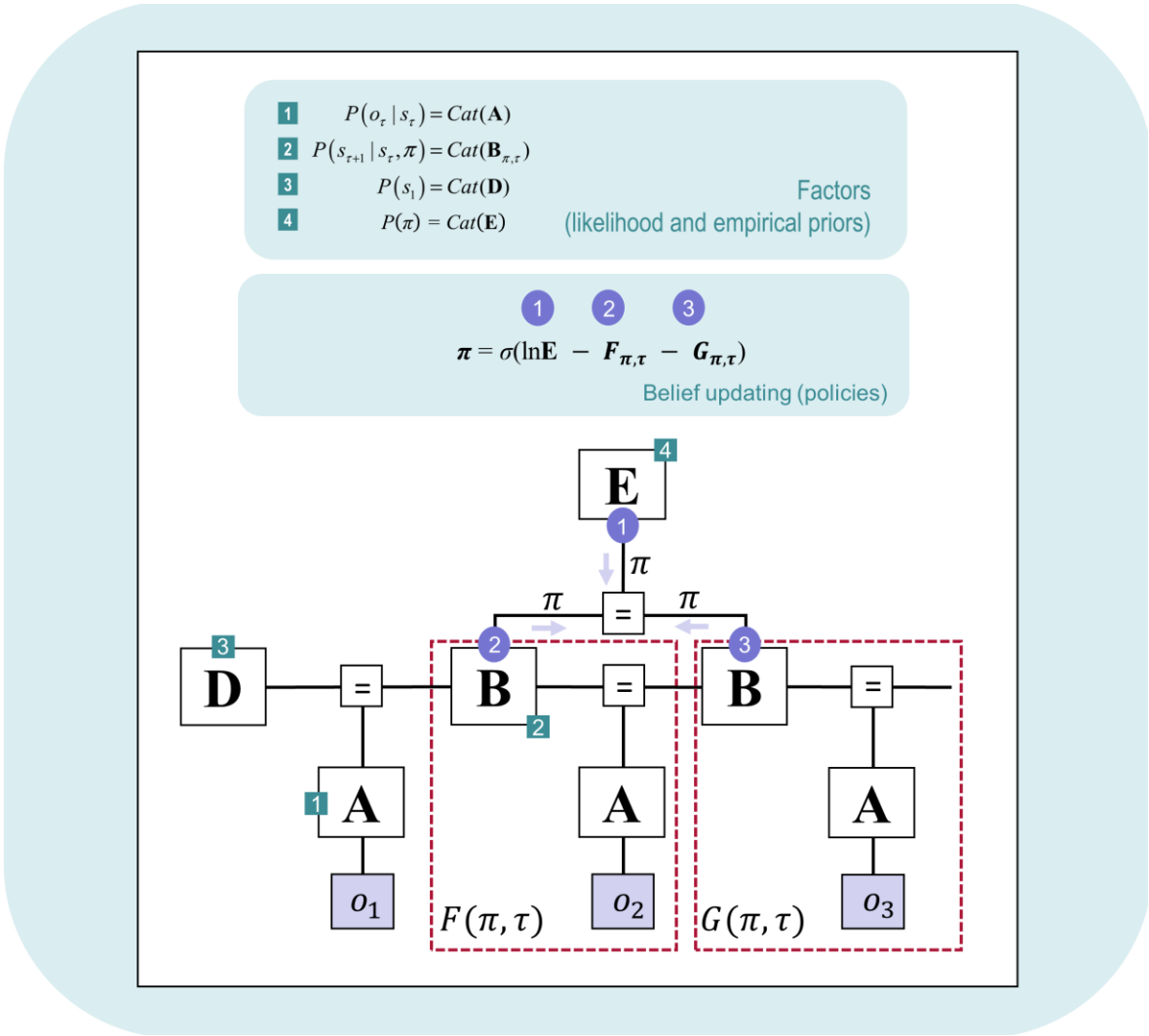

Figure 6 (adapted from (Friston, Parr, et al., 2017)). **Top**. As in **Figure 5,** these equations specify the factors that constitute the factorized generative model, and the numbers in the green squares highlight the correspondence between the equations and the factors represented by the normal factor graph below. **Middle** Belief update equation for inferring the posterior over policies. Purple numbers indicate the correspondence between terms within the update equations and the messages passed between each factor shown in the factor graph in the lower right. **Bottom**. Normal factor graph representation of the message passing that underlies posterior inference over policies. Red dotted lines show partition functions of the graph which are used to construct the free energy approximations to the marginal probability of observations conditioned on policies, $-\ln P(o_t|\pi) \approx F_{\boldsymbol{\pi},\boldsymbol{\tau}}$, and the expected marginal probability of observations conditioned on policies, $-E_{q(o_t,s_t|\pi)}[\ln P(o_{\tau>t}|\pi)] \approx G_{\pi,\tau}$. The factors $F_{\boldsymbol{\pi},\boldsymbol{\tau}}$ and $G_{\pi,\tau}$ then become the messages (shown by the purple arrows) sent from the **B** factors that converge on the equality constraint node (between **B** and E) along with the message sent from E, and after the application of a softmax function, become the posterior over policies.

At this juncture, it is important to highlight a more recent development in active inference research. Namely, because variational message passing and the mean-field approximation have known limitations, a recently improved algorithm called **marginal message passing** has been adopted, and is incorporated into the most recent software implementation (**spm_MDP_VB_X.m**; as well as in the tutorial version included as **supplementary materials**: **spm_MDP_VB_X_tutorial.m**). Briefly, marginal message passing represents a type of compromise between the computational efficiency of variational message passing and another widely used algorithm – called belief propagation – that is more computationally expensive but can perform exact (as opposed to approximate) inference under suitable conditions (for details, see (Parr et al., 2019)). A full explanation of marginal message passing is beyond the scope of this tutorial, as it first requires a more thorough introduction to both variational message passing and belief propagation. Here, we chose to introduce the reader to the mean field approximation in combination with variational message passing due to its simplicity and wide usage within the active inference literature, and to provide the interested reader with a foundation to build from when pursuing more details on these topics elsewhere.

However, the major resulting adjustment under marginal message passing is that the term for state prediction error $\boldsymbol{\varepsilon_{\pi,\tau}}$ becomes:

$$\varepsilon_{\pi,\tau} = \frac{1}{2}\left(\ln\left(\mathbf{B}_{\pi,\tau-1}s_{\pi,\tau-1}\right) + \ln\left(\mathbf{B}^{\dagger}_{\pi,\tau}s_{\pi,\tau+1}\right)\right) + \ln\mathbf{A}^{\boldsymbol{T}}o_{\tau} - \ln s_{\pi,\tau}$$

The result of adding the ½ to scale the influence of transition beliefs (**B**) is that the precision of transition probabilities is reduced. This prevents overestimation of the precision of posteriors – something that variational message passing is prone to do under conditions of uncertainty. Also note that $\mathbf{B}^{\dagger}$denotes the transpose of **B** with normalized columns (i.e., columns that sum to 1). As presented here, this modification may come across somewhat ad hoc. However, as with variational message passing, the update equations for marginal message passing can be derived in a principled manner (described in (Parr et al., 2019)).

2.4 Prediction Error Formulation

For those who skipped the previous section, recall that one strength of active inference is that it comes equipped with a biologically plausible instantiation in terms of prediction-error minimization. To make this explicit, one can introduce two types of prediction errors – 'state' and 'outcome' prediction errors – based on the equations for F_{π} and G_{π} in **Table 2**. State prediction errors were introduced in the previous (optional) section. These track how F_{π} changes over time as *beliefs about states* $\boldsymbol{s}_{\pi,\tau}$ are updated (i.e., reductions in F_{π} correspond to reductions in state prediction error):

State Prediction Errors:

$$\varepsilon_{\pi,\tau} = \frac{1}{2}\left(\ln\left(\mathbf{B}_{\pi,\tau-1}s_{\pi,\tau-1}\right) + \ln\left(\mathbf{B}^{\dagger}_{\pi,\tau}s_{\pi,\tau+1}\right)\right) + \ln\mathbf{A}^{T}o_{\tau} - \ln s_{\pi,\tau}$$

For those who skipped the technical section, note that $\mathbf{B}^{\dagger}$denotes the transpose of **B** with normalized columns (i.e., columns that sum to 1). In this equation, the combination of the two **B** matrices (combined with state beliefs) correspond to priors, whereas the **A** matrix (combined with

observations) corresponds to the likelihood. By finding posterior beliefs over states $s_{\pi,\tau}$ (on the far right) that minimize $\varepsilon_{\pi,\tau}$, F_π is minimized and $s_{\pi,\tau}$ becomes a stable posterior belief**.** To make this more concrete, consider a worked example with the following variable values entailed by a policy:

$$\mathbf{A} = \begin{bmatrix} .8 & .4 \\ .2 & .6 \end{bmatrix};\ \mathbf{B}_{\pi,\tau-1} = \begin{bmatrix} .8 & .2 \\ .2 & .8 \end{bmatrix};\ \mathbf{B}_{\pi,\tau} = \begin{bmatrix} .3 & .3 \\ .7 & .7 \end{bmatrix};\ o_\tau = \begin{bmatrix} 1 \\ 0 \end{bmatrix};\ s_{\pi,\tau} = \begin{bmatrix} .5 \\ .5 \end{bmatrix};\ v = \ln \begin{bmatrix} .5 \\ .5 \end{bmatrix}$$

As can be seen here, the likelihood (**A**) matrix indicates that outcome 1 (row 1) is more likely (i.e., $p = .8$) under state 1 (column 1). Under this policy, the agent believes it will most likely remain in its current state in the first state transition (i.e., p = .8 in $\mathbf{B}_{\pi,\tau-1}$; columns indicate states at time $\tau - 1$, rows indicate state at time τ) and more likely move to state 2 during the second state transition (i.e., p = .7 in $\mathbf{B}_{\pi,\tau}$). Further, the agent observes outcome 1 at time τ (o_τ) and had a prior expectation that both states were equally likely ($s_{\pi,\tau}$ and v). In this case, the error signal will be:

$$\varepsilon = \frac{1}{2}\ln \begin{bmatrix} .8 & .2 \\ .2 & .8 \end{bmatrix} \begin{bmatrix} .5 \\ .5 \end{bmatrix} + \frac{1}{2}\ln \begin{bmatrix} .5 & .5 \\ .5 & .5 \end{bmatrix} \begin{bmatrix} .5 \\ .5 \end{bmatrix} + \ln \begin{bmatrix} .8 & .2 \\ .4 & .6 \end{bmatrix} \begin{bmatrix} 1 \\ 0 \end{bmatrix} - \ln \begin{bmatrix} .5 \\ .5 \end{bmatrix}$$

$$= \frac{1}{2}\ln \begin{bmatrix} .5 \\ .5 \end{bmatrix} + \frac{1}{2}\ln \begin{bmatrix} .5 \\ .5 \end{bmatrix} + \ln \begin{bmatrix} .8 \\ .4 \end{bmatrix} - \ln \begin{bmatrix} .5 \\ .5 \end{bmatrix}$$

$$= \begin{bmatrix} -.9163 \\ -1.609 \end{bmatrix} - \begin{bmatrix} -.6931 \\ -.6931 \end{bmatrix}$$

$$= \begin{bmatrix} -.2231 \\ -.9163 \end{bmatrix}$$

This error signal will then update beliefs over states as follows:

$$v = v + \varepsilon = \begin{bmatrix} -.6931 \\ -.6931 \end{bmatrix} + \begin{bmatrix} -.2231 \\ -.9163 \end{bmatrix} = \begin{bmatrix} -.9163 \\ -1.6094 \end{bmatrix}$$

$$s = \sigma(v) = \begin{bmatrix} 0.6667 \\ 0.3333 \end{bmatrix}$$

Notice that in this example the variational update (i.e., single step of gradient descent) results in a negative value for the state prediction error term (i.e., $\varepsilon = \begin{bmatrix} -.2231 \\ -.9163 \end{bmatrix}$. As can be seen, this shifts the approximate posterior such that it will better minimize prediction error in the next variational update (i.e., here increasing the probability of occupying state 1).

In contrast to state prediction errors, outcome prediction errors track how G_π changes over time as *beliefs about policies* are updated (i.e., reductions in G_π correspond to reductions in outcome prediction error). In other words, when this type of prediction error is minimized, policies are identified that minimize the expected difference between predicted and preferred outcomes (and minimize ambiguity, which maximizes information gain).

Outcome Prediction Errors:

$$\boldsymbol{\varsigma}_{\pi,\tau} = \mathbf{A}s_{\pi,\tau} \cdot \left(\ln \mathbf{A}s_{\pi,\tau} - \ln \mathbf{C}_\tau\right) - diag(\mathbf{A}^T \ln \mathbf{A}) \cdot s_{\pi,\tau}$$

Here, the first term, $\mathbf{A}s_{\pi,\tau} \cdot (\ln \mathbf{A}s_{\pi,\tau} - \ln \mathbf{C}_\tau)$, corresponds to the difference between preferred outcomes and the outcomes expected under a policy (i.e., because $\mathbf{A}s_{\pi,\tau}$ corresponds to the observations expected under a policy, $o_{\pi,\tau}$). The second term, $diag(\mathbf{A}^T \ln \mathbf{A}) \cdot s_{\pi,\tau}$, corresponds to how much observations are expected to update beliefs if adopting a particular policy (i.e., it is the entropy term, where lower entropy entails greater information gain). Note that the $diag()$ function simply takes the diagonal elements of a matrix and places them in a row vector. Those who read the technical section on VFE and EFE will recognize these two terms as the matrix forms of the risk ($D_{KL}[q(o_\tau|\pi)||p(o_\tau)] \approx \mathbf{A}s_{\pi,\tau} \cdot (\ln \mathbf{A}s_{\pi,\tau} - \ln \mathbf{C}_\tau)$, and ambiguity ($E_{q(S|\pi)}[\mathrm{H}[p(o_\tau|s_\tau)] \approx -diag(\mathbf{A}^T \ln \mathbf{A}) \cdot s_{\pi,\tau}$) terms in EFE. Again, to make this more concrete we provide a worked example of each term under two possible policies, assuming the following variable values:

$$\mathbf{A} = \begin{bmatrix} .9 & .1 \\ .1 & .9 \end{bmatrix}; \quad \mathbf{C}_\tau = \begin{bmatrix} 1 \\ 0 \end{bmatrix}; \quad s_{\pi=1,\tau} = \begin{bmatrix} .9 \\ .1 \end{bmatrix}; \quad s_{\pi=2,\tau} = \begin{bmatrix} .5 \\ .5 \end{bmatrix}$$

In other words, the agent prefers outcome 1 (row 1), and the likelihood ($\mathbf{A}$) matrix indicates that state 1 (column 1) is more likely to generate outcome 1 (i.e., $p = .9$). Further, state beliefs under policy 1 ($s_{\pi=1,\tau}$) entail a higher probability of being in state 1 (i.e., $p = .9$) than state beliefs under policy 2 ($s_{\pi=2,\tau}$; i.e., $p = .5$). We can first calculate the risk (reward-seeking) term within the outcome prediction error for each policy:

Policy 1:

$$o_{\pi=1,\tau} = \mathbf{A}s_{\pi=1,\tau} = \begin{bmatrix} .82 \\ .18 \end{bmatrix}$$

$$\mathbf{A}s_{\pi=1,\tau} \cdot \left(\ln \mathbf{A}s_{\pi=1,\tau} - \ln \mathbf{C}_\tau\right) =$$

$$\begin{bmatrix} .82 \\ .18 \end{bmatrix}^T \cdot \left(\ln \begin{bmatrix} .82 \\ .18 \end{bmatrix} - \ln \begin{bmatrix} 1 \\ 0 \end{bmatrix}\right) = 2.4086$$

Policy 2:

$$o_{\pi=2,\tau} = \mathbf{A}s_{\pi=2,\tau} = \begin{bmatrix} .5 \\ .5 \end{bmatrix}$$

$$\mathbf{A}s_{\pi=2,\tau} \cdot (\ln \mathbf{A}s_{\pi=2,\tau} - \ln \mathbf{C}_\tau) =$$

$$\begin{bmatrix} .5 \\ .5 \end{bmatrix}^T \cdot \left(\ln \begin{bmatrix} .5 \\ .5 \end{bmatrix} - \ln \begin{bmatrix} 1 \\ 0 \end{bmatrix}\right) = 7.3069$$

Note that a negligibly small number (here, e^{-16}) is added to the values in $\mathbf{C}_\tau$ because $\ln(0)$ is undefined. As expected, because the approximate posterior over states for policy 1 makes the generation of preferred observations more likely, policy 1 has lower values for the risk term (i.e., leading to lower outcome prediction error, all else being equal).

Moving onto the ambiguity (information-seeking term), consider another example with the following variables:

$$\mathbf{A} = \begin{bmatrix} .4 & .2 \\ .6 & .8 \end{bmatrix}; \quad s_{\pi=1,\tau} = \begin{bmatrix} .9 \\ .1 \end{bmatrix}; \quad s_{\pi=2,\tau} = \begin{bmatrix} .1 \\ .9 \end{bmatrix}$$

In this case, the likelihood (**A**) matrix indicates that state 2 (column 2) has a more precise distribution than state 1 (column 1). In other words, observations are expected to provide more precise information about states when in state 2. As such, we expect outcome prediction errors to drive selection of the policy that will lead the agent toward state 2. In this case, policy 2 assigns a higher probability to state 2 (i.e., = .9 in row 2). We can confirm this by calculating the ambiguity term for each policy as follows:

Policy 1:

$$diag(\mathbf{A}^T \ln \mathbf{A}) \cdot s_{\pi=1,\tau} = [diag\left(\begin{bmatrix} .4 & .6 \\ .2 & .8 \end{bmatrix} \ln \begin{bmatrix} .4 & .2 \\ .6 & .8 \end{bmatrix}\right)]^T \cdot \begin{bmatrix} .9 \\ .1 \end{bmatrix}$$

$$= [diag(\begin{bmatrix} -.67 & -.78 \\ -.59 & -.50 \end{bmatrix}) \cdot \begin{bmatrix} .9 \\ .1 \end{bmatrix}]^T = [-.67 - .50] \cdot \begin{bmatrix} .9 \\ .1 \end{bmatrix}$$

$$= -.6558$$

Policy 2:

$$diag(\mathbf{A}^T \ln \mathbf{A}) \cdot s_{\pi=2,\tau} = [diag(\begin{bmatrix} .4 & .6 \\ .2 & .8 \end{bmatrix} \ln \begin{bmatrix} .4 & .2 \\ .6 & .8 \end{bmatrix})]^T \cdot \begin{bmatrix} .1 \\ .9 \end{bmatrix}$$

$$= [diag(\begin{bmatrix} -.67 & -.78 \\ -.59 & -.50 \end{bmatrix})]^T \cdot \begin{bmatrix} .1 \\ .9 \end{bmatrix} = [-.67 - .50] \cdot \begin{bmatrix} .1 \\ .9 \end{bmatrix}$$

$$= -.52$$

As expected, because the outcomes generated by state 1 are more ambiguous (i.e. less informative), and policy 2 assigns a higher probability to state 2 than policy 1, policy 2 better minimizes ambiguity.

It is important to stress that the risk and ambiguity terms for outcome prediction errors work synergistically, and one often has policies that minimize both risk and ambiguity. As can be seen in the outcome prediction error equation above, subtracting the ambiguity term (which is negative) from the risk term, which we have calculated separately here, will drive selection of policies that maximize both reward- and information-seeking by minimizing the overall resulting error.

While these formal calculations may appear somewhat involved (even for a single policy), an intuitive way to think about these two prediction errors is that minimizing state prediction error maximizes confidence in posterior beliefs, while minimizing outcome prediction error maximizes confidence in how to achieve goals or desires. To reproduce the worked examples above and allow the reader to calculate state and outcome prediction errors under different model parameters, we have provided the **Prediction_error_example.m** script in supplementary materials.

2.5 Brief clarification about time indexing

Another final note on the update equations shown above, especially for those that skipped the optional section on message passing, involves the common (and understandable) confusion regarding the distinction between beliefs *at* a time *t* and beliefs *about* a time τ (i.e., the Greek letter

tau; perhaps an unfortunate choice given that t and τ look so similar). Succinctly, to appreciate the need for this distinction, consider how prior expectations can also propagate backward in time (i.e., via the backward message, $\mathbf{B}^{\dagger}_{\boldsymbol{\pi},\boldsymbol{\tau}} s_{\pi,\tau+1}$, above), such that beliefs *about* one's state at time $\tau = 1$ can change when making a new observation *at* time $t = 2$. For example, consider a case in which you start out in one of two rooms (a green room or a blue room), but you do not know what color the walls are. Later, when you open your eyes and find out the room is painted blue, you will change your belief *now* about where you were *earlier* before you opened your eyes (i.e., you had been in the blue room the whole time). So, in active inference, the model updates its beliefs about states at all time points τ with each observation at each time point t. This therefore allows for *retrospective inference* (as well as *prospective inference)*. This is an important distinction to keep in mind when trying to understand simulation results (e.g., in terms of working memory for the past and future, i.e., postdiction and prediction).

In practice, this is accomplished by having entries of 0 for all elements of an observation vector when $t < \tau$. So, for example, consider a 'color' observation modality where observations could be 'blue' or 'green' (i.e., a vector with one element for each color). At time $t = 1$, the observed color for time $\tau = 2$ has not yet occurred. So, at $t = 1$:

$$o_{\tau=2} = \begin{bmatrix} 0 \\ 0 \end{bmatrix}$$

If blue were then observed at $t = 2$, the observation for $\tau = 2$ would be updated to:

$$o_{\tau=2} = \begin{bmatrix} 1 \\ 0 \end{bmatrix}$$

This vector would then remain unchanged for all future time points $t > 2$ (i.e., the observation is never 'forgotten' once it has taken place). The same thing would then occur for all subsequent observations (e.g., $o_{\tau=3} = \begin{bmatrix} 0 \\ 0 \end{bmatrix}$ at $t = 1$ and 2; but if green were observed at $t = 3$ then the vector would be updated to $o_{\tau=3} = \begin{bmatrix} 0 \\ 1 \end{bmatrix}$and remain that way for $t > 3$, etc.). This allows beliefs about states for all time points to be updated at each time point t when these observation vectors are updated.

3. Building Specific Task Models

3.1 Explore-Exploit Task

To make this structure more concrete, we will now use it to build models of specific behavioral tasks. We will first make use of an 'explore-exploit' task, in which a participant is asked to repeatedly choose one of two slot machines. This will also allow us to concretely demonstrate some of the unique resources offered by active inference when modelling behavior in a simple reinforcement learning context. To be sure, in many task contexts (e.g., when there is no uncertainty about states) active inference models can perform similarly to reinforcement learning models (Da Costa, Sajid, Parr,

Friston, & Smith, 2020)[5]. However, as discussed above, when tasks involve various types of uncertainty (e.g., about task condition or reward probabilities), active inference offers a unique approach for modelling information-seeking behavior that can lead to superior performance in some cases (Markovic, Stojic, Schwoebel, & Kiebel, 2021; Sajid, Ball, Parr, & Friston, 2021). Another resource offered by active inference, even when it performs similarly to other types of models, is its associated neural process theory (i.e., describing how neural signaling might implement variational or marginal message passing; see Section 4). The task models we build in this tutorial will further allow us to illustrate how active inference can be used to make testable empirical predictions about neural responses. As we will see, because active inference models integrate perception, learning, and decision-making in a single model architecture, this affords the generation of predictions about neural responses across a wide range of perceptual tasks in addition to reinforcement learning and decision-making tasks.

Every step we outline in this section for building the explore-exploit task is laid out in the accompanying MATLAB code (**Step_by_Step_AI_Guide.m**). This code is included in **supplementary materials**; this, and all **supplementary code**, can also be found at: https://github.com/rssmith33/Active-Inference-Tutorial-Scripts. While going through this section, we encourage the reader to work through this code in parallel.

In the beginning of the explore-exploit task, the participant is told that on each trial one machine will tend to pay out more often, but they will not know which one. They are also told that the better machine will not always be the same on each trial. They can choose to select one right away and possibly win $4. Or they can choose to press a button that gives them a hint about which slot machine is better on that trial. However, if they choose to take the hint, they can only win $2 if they pick the correct machine. Over many trials, the participant can learn which slot machine tends to pay out more often and either make safe or risky choices (i.e., take the hint or not).

To model this task, it can be helpful to start by specifying the sets of possible hidden states (state factors). In this case, one state factor corresponds to whether the left or right slot machine is more likely to win ('left-better context', 'right-better context'). The second state factor corresponds to the choice state ('start state', 'asking for the hint', 'choosing the left machine', 'choosing the right machine'). Moving to MATLAB code, we can set these up by specifying the priors over initial states with a set of vectors **D**, with one vector per state factor (i.e., where the factor number is specified in brackets). The general structure for these vectors is:

$$\boldsymbol{D}\{\textbf{state factor}\}(\mathbf{1},\textbf{state}) = [\textbf{vector}]'$$

In this case, we specify:

[5] Although note that, even in task contexts where they perform similarly, active inference models and reinforcement learning models make decisions in a different way. Specifically, while reinforcement learning models seek to maximize a reward signal, active inference models instead seek to reach a target distribution that is treated as rewarding (i.e., the distribution encoding the agent's preferred observations).

$\boldsymbol{D}\{\mathbf{1}\} = [\mathbf{0.5\ 0.5}]'$

$\boldsymbol{D}\{\mathbf{2}\} = [\mathbf{1\ 0\ 0\ 0}]'$

This says that the participant begins with the belief that the 'left-better' and 'right-better' contexts have equal probability (left and right entries, respectively), and with a fully precise belief that he/she will start the trial in the start state (from left to right: 'start', 'get hint', 'choose left', and 'choose right' states).

Next, we must specify the sets of possible observations (outcome modalities). Here, one set of observations corresponds to the hint ('no hint', 'machine-left hint', 'machine-right hint'). Another set of observations corresponds to decision outcomes ('start', 'losing', and 'winning'). Finally, the participant also observes his/her own choices ('start', 'asking for the hint', 'choosing the left machine', 'choosing the right machine'). In MATLAB, we can set these up by specifying the likelihood (**A**) matrices. There will always be one **A** matrix for each outcome modality. **Rows correspond to outcomes**, **columns correspond to the states in the first state factor**, and there is **an additional dimension for each additional state factor**. The general structure is therefore:

$\mathbf{A}\{\mathbf{outcome\ modality}\}(\mathbf{outcome}, \mathbf{factor\ 1}, \mathbf{factor\ 2}, \ldots, \mathbf{factor\ N}) = [\mathbf{matrix}]$

In this case, only the 'hint' state in state factor 2 generates a hint observation, so for the third dimension we specify a '2' as follows:

$$\mathbf{A}\{\mathbf{1}\}(:, :,\ \mathbf{2}) = \begin{bmatrix} \mathbf{0} & \mathbf{0} \\ \mathbf{1} & \mathbf{0} \\ \mathbf{0} & \mathbf{1} \end{bmatrix}$$

Here, columns from left to right correspond to the 'left-better' and 'right-better' states, while rows from top to bottom correspond to the 'no hint', 'machine-left hint', 'machine-right hint' observations. This matrix indicates that the hint is accurate with a probability of 1 (100% accuracy). For example, a 'machine-left hint' observation (row two) will be generated by the 'left-better context' (column one) with probability = 1. Here each column must add up to 1. For the other dimensions of state factor 2 (i.e., matrix dimension 3):

$\boldsymbol{for\ i = 1, 3, 4:}$

$$\mathbf{A}\{\mathbf{1}\}(:, :,\ \boldsymbol{i}) = \begin{bmatrix} \mathbf{1} & \mathbf{1} \\ \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} \end{bmatrix}$$

This indicates that all other choices state will generate the 'no hint' observation (i.e., a hint will never be observed in those states).

For the second outcome modality, the 'start' and 'hint' states generate 'start' observations (row 1):

$\boldsymbol{for\ i = 1, 2:}$

$$\mathbf{A}\{\mathbf{2}\}(:,:,\ \boldsymbol{i}) = \begin{bmatrix} \mathbf{1} & \mathbf{1} \\ \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} \end{bmatrix}$$

We will now specify the probability of winning in the 'left-better context' vs. the 'right-better context' depending on choice state. Here we will specify that, if in the 'left-better context', choosing the left machine (i.e., transitioning to the 'choose left machine' state) will lead to a win 80% of the time (row 3) and choosing the right machine will lead to a win 20% of the time, with inverse probabilities for a loss (row 2), and a probability of 0 of continuing to observe the 'start' observation (row 1) after making those choices:

$$\mathbf{A}\{\mathbf{2}\}(:,:,\mathbf{3}) = \begin{bmatrix} \mathbf{0} & \mathbf{0} \\ .\mathbf{2} & .\mathbf{8} \\ .\mathbf{8} & .\mathbf{2} \end{bmatrix}$$

We will then specify that the probabilities of winning are reversed if in the 'right-better context'.

$$\mathbf{A}\{\mathbf{2}\}(:,:,\mathbf{4}) = \begin{bmatrix} \mathbf{0} & \mathbf{0} \\ .\mathbf{8} & .\mathbf{2} \\ .\mathbf{2} & .\mathbf{8} \end{bmatrix}$$

Remember, the first column is the context where the left machine is better, and the second column is the context where the right machine is better. It is the third dimension that corresponds to behavioral states (in this case, behavioral state 3 and 4, corresponding to choosing the left vs. right machine).

Finally, for the third (choice observation) outcome modality, states simply map 1-to-1 to outcomes (across all other state combinations):

$$\mathbf{A}\{\mathbf{3}\}(:,:,\ \mathbf{1}) = \begin{bmatrix} \mathbf{1} & \mathbf{1} \\ \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} \end{bmatrix}$$

$$\mathbf{A}\{\mathbf{3}\}(:,:,\ \mathbf{2}) = \begin{bmatrix} \mathbf{0} & \mathbf{0} \\ \mathbf{1} & \mathbf{1} \\ \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} \end{bmatrix}$$

$$\mathbf{A}\{\mathbf{3}\}(:,:,\ \mathbf{3}) = \begin{bmatrix} \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} \\ \mathbf{1} & \mathbf{1} \\ \mathbf{0} & \mathbf{0} \end{bmatrix}$$

$$\mathbf{A}\{\mathbf{3}\}(:,:,\ \mathbf{4}) = \begin{bmatrix} \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} \\ \mathbf{1} & \mathbf{1} \end{bmatrix}$$

This simply indicates that the individual always knows what their choice was with complete certainty (rows top to bottom: ('start', 'asking for the hint', 'choosing the left machine', 'choosing the right machine' observations).

Now that we have the likelihood, the next step is to specify the (policy-dependent) state transition (**B**) matrices. The general structure for these matrices is:

$$\mathbf{B}\{\textbf{state factor}\}(\textbf{state at time } \boldsymbol{\tau}+\mathbf{1}, \textbf{state at time } \boldsymbol{\tau}, \textbf{action number}) = [\textbf{matrix}]$$

Because we have two state factors, we need two sets of matrices. The first matrix is for the context factor. In this case, because a context remains the same within each trial, this is simply an identity matrix that says **states at time $\boldsymbol{\tau}$ (columns)** remain the same **at $\boldsymbol{\tau}+\mathbf{1}$ (rows)**:

$$\mathbf{B}\{\mathbf{1}\}(:,:,\ \mathbf{1}) = \begin{bmatrix} \mathbf{1} & \mathbf{0} \\ \mathbf{0} & \mathbf{1} \end{bmatrix}$$

Columns from left to right and rows from top to bottom both correspond to the 'left-better' and 'right-better' states. There is only one 'action' (possible transition from each state) for this factor, so the third dimension is a 1 and has a length of 1; i.e., there is no B{1}(:, :, 2).

In contrast, the second state factor is choice state, where four different transitions (actions) are possible at each time step. In this case, each matrix below indicates that one could move from any state to the chosen state:

$$\mathbf{B}\{\mathbf{2}\}(:,:,\ \mathbf{1}) = \begin{bmatrix} \mathbf{1} & \mathbf{1} & \mathbf{1} & \mathbf{1} \\ \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} \end{bmatrix}$$

$$\mathbf{B}\{\mathbf{2}\}(:,:,\ \mathbf{2}) = \begin{bmatrix} \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} \\ \mathbf{1} & \mathbf{1} & \mathbf{1} & \mathbf{1} \\ \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} \end{bmatrix}$$

$$\mathbf{B}\{\mathbf{2}\}(:,:,\ \mathbf{3}) = \begin{bmatrix} \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} \\ \mathbf{1} & \mathbf{1} & \mathbf{1} & \mathbf{1} \\ \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} \end{bmatrix}$$

$$\mathbf{B}\{\mathbf{2}\}(:,:,\ \mathbf{4}) = \begin{bmatrix} \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} \\ \mathbf{1} & \mathbf{1} & \mathbf{1} & \mathbf{1} \end{bmatrix}$$

In other words, choice 1, B{2}(:, :, 1), entails moving to the 'start' state from any other state, choice 2, B{2}(:, :, 2), entails moving to the 'get hint' state from any other state, etc. The third dimension labels these as **action numbers 1, 2, 3, and 4**.

Next, we need to specify preferences over each set of outcomes (**C**), with one matrix per outcome modality. Here, **rows indicate observations** (same order as in the corresponding **A** matrices) and **columns indicate time points** in a trial from left to right. In other words:

$$\mathbf{C}\{\mathbf{outcome\ modality}\}(\mathbf{outcome}, \mathbf{time\ point}) = [\mathbf{matrix}]$$

In this case, the model has no direct preference for getting a hint (i.e., preferences for outcome modality 1: $C\{1\}$) or for observing the choice of a particular action (preferences for outcome modality 3: $C\{3\}$). So:

$$\mathbf{C}\{\mathbf{1}\} = \begin{bmatrix} \mathbf{0} & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \mathbf{0} \end{bmatrix}$$

$$\mathbf{C}\{\mathbf{3}\} = \begin{bmatrix} \mathbf{0} & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \mathbf{0} \end{bmatrix}$$

Here, columns from left to right indicate $\tau = 1, 2, 3$ in the trial. The model does have preferences for winning and losing (outcome modality 2), which are specified as follows:

$$\mathbf{C}\{\mathbf{2}\} = \begin{bmatrix} \mathbf{0} & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathbf{-1} & \mathbf{-1} \\ \mathbf{0} & \mathbf{4} & \mathbf{2} \end{bmatrix}$$

This indicates that, at the second and third time points, a win (row 3) has a value of 4 and 2, respectively. Remember, the value is less at the third time point (third column) because the individual wins less money if they instead choose to take the hint at the second time point (i.e., and thus continues to observe the 'start' outcome in row 1 for this outcome modality). The values of -1 in row 2 indicate a preference against observing a loss at time points 2 and 3. Note that we only initially specify the **C** matrix values in this form for convenience. In the code these preference distributions are first passed through a softmax function such that each column in the **C** matrix encodes a proper probability distribution that sums to 1, at which point a natural log is applied to each element of the matrix. This means that the values are transformed into log-probabilities as follows when used during simulations (i.e., less negative indicates more preferred):

$$\mathbf{ln}\boldsymbol{p}(\boldsymbol{o}) = \mathbf{C}\{\mathbf{2}\} = \begin{bmatrix} \mathbf{-1.1} & \mathbf{-4.0} & \mathbf{-2.1} \\ \mathbf{-1.1} & \mathbf{-5.0} & \mathbf{-3.2} \\ \mathbf{-1.1} & \mathbf{-0.02} & \mathbf{-0.2} \end{bmatrix}$$

Next, we need to specify allowable policies. There are three time points in a trial for this task, which means a policy will consist of two actions. If we want to include **'shallow' policies**, where the model only looks one step ahead, we need to specify a matrix U including one row, **one column for each allowable action**, and **a third dimension specifying each state factor**. Thus, the structure is:

$$\boldsymbol{U}(\mathbf{1}, \mathbf{action\ number}, \mathbf{state\ factor}) = [\mathbf{vector}]$$

In this case, we can include all actions:

$\boldsymbol{U}(:,:,\mathbf{1}) = [\mathbf{1\ 1\ 1\ 1}]$

$\boldsymbol{U}(:,:,\mathbf{2}) = [\mathbf{1\ 2\ 3\ 4}]$

Entries for $U(:,:,2)$ allow all four possible transitions (actions) between choice states (factor 2) at each time point. The entries for $U(:,:,1)$ are all ones because there is only one possible 'action' – that is, one transition (**B**) matrix – for state factor 1. There still needs to be four ones within $U(:,:,1)$ to match the number of actions in $U(:,:,2)$. In other words, each overall action option corresponds to the combined entries in a given column for both state factors. Although not shown in detail here, this also affords the possibility of *multidimensional* policies in more complex models. For example, if there were multiple possible actions (transition matrices) for two different state factors, one might specify that action 2 for state factor 1 can be chosen together with action 2 for state factor 2, but that action 2 for state factor 1 cannot be chosen together with action 3 for state factor 2 (simply by including a column that has entries of 2 for both state factors but no column that has an entry of 2 for state factor 1 and an entry of 3 for state factor 2).

If we instead want to include **'deep' policies**, where the simulated participant plans ahead until the end of the trial, this means that we need to specify **one column for each allowable policy (with each entry indicating an action number)** in a matrix **V**, with **one row per time point** and a **third dimension specifying each state factor**. Thus, the general structure is:

$\mathbf{V}(\mathbf{time\ point}, \mathbf{policy}, \mathbf{state\ factor}) = [\mathbf{matrix}]$

In this case, we might reasonably include five policies:

$$\mathbf{V}(:,:,\mathbf{1}) = \begin{bmatrix} \mathbf{1} & \mathbf{1} & \mathbf{1} & \mathbf{1} & \mathbf{1} \\ \mathbf{1} & \mathbf{1} & \mathbf{1} & \mathbf{1} & \mathbf{1} \end{bmatrix}$$

$$\mathbf{V}(:,:,\mathbf{2}) = \begin{bmatrix} \mathbf{1} & \mathbf{2} & \mathbf{2} & \mathbf{3} & \mathbf{4} \\ \mathbf{1} & \mathbf{3} & \mathbf{4} & \mathbf{1} & \mathbf{1} \end{bmatrix}$$

As with U, all policies here do not change state factor 1 at either time point (hence, all entries are ones; but they need not all be ones in a more complex model with multidimensional policies, as also described for U above). For state factor 2, we have included policies in which the model chooses to remain in the 'start state' (i.e., choosing action 1 twice; column 1), chooses to take the hint (action 2) and then select either of the slot machines (i.e., actions 3 or 4; columns 2-3), or decides to choose a slot machine right away (columns 4-5; note, these policies subsequently return to state 1, since it is not possible to win twice in one trial). We will use deep policies in the simulations below.

If so desired, one can also specify a prior over policies E to incorporate a bias or 'habit' to select some policies over others. This is simply a **column vector with one entry per policy** that encodes the probability of that policy. Here we will not include such a bias, which means that E will simply be a flat distribution over our 5 allowable policies in **V**:

$$\boldsymbol{E} = [\frac{\mathbf{1}}{\mathbf{7}} \ \frac{\mathbf{1}}{\mathbf{7}} \ \frac{\mathbf{1}}{\mathbf{7}} \ \frac{\mathbf{1}}{\mathbf{7}} \ \frac{\mathbf{1}}{\mathbf{7}} \ ]'$$

Finally, there are several scalar (single-value) parameters one can set. One is beta (β), which is the prior on the expected free energy precision term γ (which encodes the precision estimate for the expected free energy over policies). A low β value (around 1) indicates high expected precision, whereas higher values (e.g., 3, 5, 10) indicate lower expected precision. Higher β values will increase randomness in policy selection and also make policy selection more influenced by habits encoded in the E vector (for an example of these dynamics, see (Smith, Khalsa, & Paulus, 2019)). This follows from the fact that it implies less confidence that model beliefs will generate preferred outcomes (Hesp et al., 2020). Another is alpha (α), which is a standard 'inverse temperature' (or 'action precision') parameter that controls randomness (e.g., motor stochasticity) in action selection after a policy has been selected (higher values indicate less randomness; typical range is between around 1 – 32, but very high values can be chosen to remove choice stochasticity).

$$P(Action|\alpha) = \ \sigma(\alpha \times \ln P(Action|\pi))$$

Here, sigma (σ) indicates a softmax (normalized exponential) function that transforms the quantity on the right into a proper probability distribution that adds up to 1 (see **supplementary materials** for more detail). Both β and α must be positive numbers. Here we will make $\beta = 1$ and $\alpha = 32$, specifying reasonable amounts of indeterminacy in action selection.

3.2 Running and Plotting Simulations

We have now specified a generative model and are ready to run single-trial simulations. To do so in MATLAB, we will assign each of our variables to a structure called mdp (for Markov decision process). Concretely, this mean assigning $\mathbf{mdp.D} = \mathbf{D}$, $\mathbf{mdp.V} = \mathbf{V}$, $\mathbf{mdp.beta} = \mathbf{beta}$, and so forth for all the scalars, vectors, and matrices constructed above. We can then run this structure through the standard active inference estimation function **spm_MDP_VB_X.m** (available in the DEM toolbox of the most recent versions of SPM academic software: http://www.fil.ion.ucl.ac.uk/spm/). This just means entering:

$$\mathbf{MDP} = \mathbf{spm\_MDP\_VB\_X}(\mathbf{mdp})$$

However, because SPM software is often updated, we include a specific version for this tutorial. So here you should run:

$$\mathbf{MDP} = \mathbf{spm\_MDP\_VB\_X\_tutorial}(\mathbf{mdp})$$

This function will simulate behavior based on an POMDP structure (i.e., it runs the equations in **Table 2**), and the output MDP (capital letters) structure will contain the simulation results. As we have specified it here, it assumes the generative process and generative model are identical (see section on learning below where we remove this assumption). It will thus generate outcomes based on the generative process and simulate the subsequent inference and decision dynamics within the generative model when observing those outcomes. Because the above mentioned simulation script

is quite complex, we also direct readers interested in the details of how the belief updating scheme is implemented to the **Simplified_simulation_script.m** script included in the **supplementary material,** which is a stripped down (but heavily commented) version of the standard model inversion scheme used in **spm_MDP_VB_X**. For clarity, this additional tutorial script inverts the same generative model of the explore-exploit task introduced above.

Single-trial behavior can be plotted with some default plotting routines. The primary single-trial plotting routine available in SPM can be run in MATLAB by entering:

spm_figure('**GetWin**','**Figure 1**'); **clf**; **spm_MDP_VB_trial**(**MDP**); **subplot**(**3**,**2**,**3**)

This plotting routine can also take additional optional inputs:

spm_MDP_VB_trial(**MDP**, **Gf**, **Gg**).

Gf: state factors to plot.

Gg: outcome modalities to plot.

For example, **spm_MDP_VB_trial**(**MDP**, **1**:**2**, **2**:**3**) would plot the first two state factors and the second and third outcome modalities.

At this point, the reader is encouraged to set the variable $\boldsymbol{Sim}$ in the first section of the accompanying tutorial code (i.e., **Step_by_Step_AI_Guide.m**, line 51) to $\boldsymbol{Sim} = \mathbf{1}$ and then click **'Run'**, which will run the model and this plotting script. Before running this script, remember to make sure SPM12 is installed and that the 'DEM' folder within the SPM folder structure is added as a path in MATLAB (…spm12\toolbox\DEM).

Based on the current model specification, a representative plot of simulation results is shown in **Figure 7A**. This and similar plots are generated from specific output fields in the MDP structure (**Table 3** describes each output field). The two panels in the top-left of **Figure 7A** show posteriors over states *at the end of the trial* (i.e., the states the model believes it was in at each time point τ when at the last time point t). Here time goes from left to right, darker indicates higher probability, and the cyan dots denote the true states. Here, the model believes it was in the 'left-better context' and that it chose to take the hint and then chose the left slot machine. The top-right shows the action probabilities and true actions. Here the agent is highly confident that taking the hint and choosing left were the best choices (and cyan dots indicate that these were also the actual actions taken). The left-middle panel just shows the different possible two-step action-sequences specified in the model (from left to right). Note that lighter shades in this panel just indicate higher action numbers (e.g., action 1 is black, action 2 is dark gray, etc.). The right-middle panel shows the evolution of the posterior distribution over policies over time (from left to right). Here, it can be seen that at the second time point the model became highly confident in policy 2 (i.e., the 'take the hint and then choose the left slot machine' policy). The three panels in the bottom left show the outcomes (cyan dots) and preference distributions. The first 'hint' modality shows that the model received the hint at time point 2. The plot is gray because there is no preference for one observation over others. This

is also the case for the third 'observed action' plot, which simply confirms what the model chose. The second 'win/lose' modality shows that a win was observed at the third time point. The preference distribution here indicates the strong preference for the win at time points 2 and 3 (darker value), and a preference not to lose (lighter value). The 'null' (starting) outcome in the top row is an intermediate gray at time point 2 (no preference for or against this outcome); it becomes darker gray at the third time point because the value of the win at time point 3 was relatively less (i.e., \$2 vs. \$4) and so the overall distribution over outcomes at the third time point is less precise.

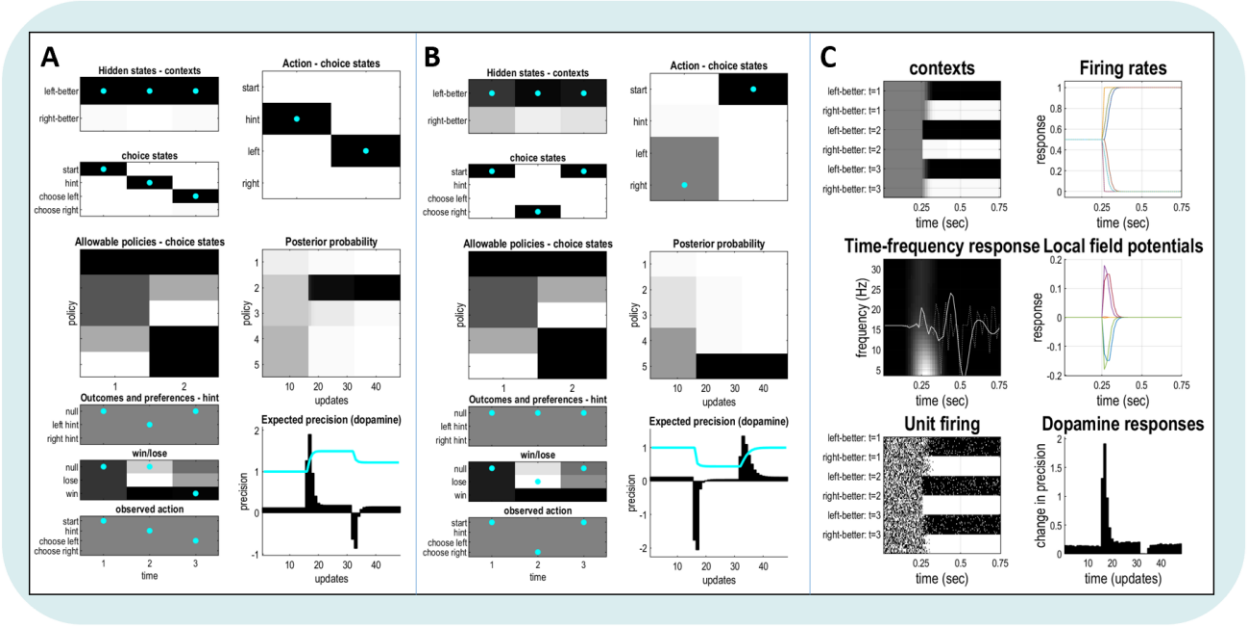

Figure 7. Example simulation plots that can be generated with the code included in this tutorial. A detailed walk-through is provided in the main text. **(A)** Example simulation of a risk-averse agent performing the explore-exploit task. The agent takes the hint, then chooses the left slot machine, and observes a win. **(B)** Example simulation of a risk-seeking agent performing the explore-exploit task. The agent foregoes the hint, then chooses the right slot machine, and observes a loss. **(C)** Example neuronal simulations based on the risk-averse agent in panel **(A)**. This illustrates the neuronal firing rates, local field potentials, and dopamine responses predicted by the neural process theory. Note that darker colors indicate higher probability values (for beliefs about states, outcomes, actions, and policies over time). For outcomes, darker values are more preferred. The 'allowable policies' plot simply displays the action sequences corresponding to each policy (darker indicates higher numbers, where each number denotes an available action). The dopamine response plots (lower right plots in each panel) correspond to updates in the expected precision of the EFE distribution over policies. In the upper and lower left plots ('contexts' and 'firing rates') of panel **C**, moving left to right along the x-axis each column corresponds to beliefs about context states at the time in which an observation was received (t), while rows from top to bottom on the y-axis correspond to the time point for which beliefs are updated (tau; τ). For example, the top-right quadrant corresponds to beliefs at time $t = 3$ about time $\tau = 1$ (note that, unfortunately, this standard SPM plotting routine inappropriately labels

each row with *t*s instead of τs). Firing rates (upper right) correspond to the magnitude of posteriors over each state (in this case, the states in the 'context' state factor), while local field potentials (middle right) correspond to their rate of change. See main text for interpretation of time frequency response plots and their motivation. These simulations can be reproduced by running the $\boldsymbol{Sim = 1}$ option in the supplementary **Step_by_Step_AI_Guide.m** code (although note that, because outcomes are sampled from probability distributions, results will not be identical each time).

Finally, the bottom-right plot displays predictions about dopamine responses in the neural process theory, which we have not yet discussed. This is based on changes in confidence about policy selection (i.e., the updates to the expected free energy precision parameter; bottom row of **Table 2**). The large jump in 'dopamine' reflects the fact that, at time point 2 in the trial, the agent received an observation about its own behavior (i.e., that it had 'chosen' to receive a hint; within outcome modality 3), which was inconsistent with all but two of the policies – thereby leading to a large update in the prior for expected free energy precision ($\boldsymbol{\beta}$, i.e., the inverse of γ). As a numerical example to help offer an intuition for how these updates operate, we can plug the VFE and EFE values that we get from the MDP structure at time point 2 into the policy distribution and $\boldsymbol{\beta}$ update equations. In these equations, γ is the expected free energy precision term, β is the initial prior for this precision at the start of a trial ($\gamma = 1/\beta$), and $\boldsymbol{\beta}$ is the posterior value for β that is continuously updated over time. For the sake of illustration, we can set γ, β, and $\boldsymbol{\beta}$ equal to one and specify the distributions over policies as follows[6]:

$$E = [1 \quad 1 \quad 1 \quad 1 \quad 1]^T$$

$$G \approx [12.505 \quad 9.51 \quad 12.5034 \quad 12.505 \quad 12.505]^T$$

$$F \approx [17.0207 \quad 1.7321 \quad 1.7321 \quad 17.0387 \quad 17.0387]^T$$

As can be seen here, the agent has no habit-like prior expectations over policies (i.e., the E distribution is flat), and the expected free energy over policies (G) favors policy 2 (i.e., entry 2 has the lowest value). The variational free energy after a new observation (F) provides precise evidence for policies 2 and 3 (values much closer to 0, indicating that the new observation is inconsistent with policies 1, 3, and 5). Given this setup, the prior and posterior distributions over policies (π_0 and $\boldsymbol{\pi}$, respectively), and the resulting difference vector, can be calculated as follows:

$$\pi_0 = \sigma(\ln E - \gamma G) = [0.0417 \quad 0.8332 \quad 0.0418 \quad 0.0417 \quad 0.0417]$$

$$\boldsymbol{\pi} = \sigma(\ln E - F - \gamma G) = [0 \quad .9523 \quad .0477 \quad 0 \quad 0]$$

$$\boldsymbol{\pi} - \pi_0 = [-.0417 \quad .1191 \quad .0.006 \quad -.0417 \quad -.0417]$$

The update in the expected free energy precision will then be:

$$\dot{\boldsymbol{\beta}} = \beta + (\boldsymbol{\pi} - \pi_0) \cdot (-G) - \boldsymbol{\beta} = .3567$$

[6] Note that the **spm_MDP_VB_X.m** script (and the tutorial version here) works with negative free energies, and so these *F* and *G* values are made negative in the MDP output structure in MATLAB.

$$\boldsymbol{\beta} = \boldsymbol{\beta} - \dot{\boldsymbol{\beta}} = 1 - .3567 = .6433$$

$$\boldsymbol{\gamma} = \frac{1}{\boldsymbol{\beta}} = 1.55$$

Note that, while we have here shown an example of a single round of updating, there are many rounds of updating to convergence for each new observation in a trial (16 iterations to converge on a stable posterior $\boldsymbol{\gamma}$ value in the code for each time point). Notice that the increase in the probability of policy 2 and 3 between the prior and posterior over policies is driven by F, which scores the evidence afforded each policy given current observations. Policy 2 and 3 better minimize F and are therefore more plausible. Taking the dot product between the vector encoding the difference between the prior and posterior over policies ($\boldsymbol{\pi} - \pi_0$) and the vector G is equivalent to scaling each element of the difference vector by the associated G value and then summing the results, which in this case creates a positive update. This is because the difference vector is pointing in roughly the same direction as the (negative) G vector. This is apparent in that G initially indicated the highest probability for policy 2, and F also provided evidence for policy 2. As a result, the $\boldsymbol{\beta}$ update then increases the impact of G on the posterior over policies because the agent is more confident in its beliefs about G. In contrast, if the agent instead chose a 'risky' policy (i.e., did not first ask for a hint), and subsequently observed a loss, there would be a large negative $\boldsymbol{\beta}$ update. This is because there would be a large difference between the posterior over policies and prior over policies, and the difference vector would be pointing in a different direction to the (negative) EFE vector (see **Figure 7B**) – decreasing the impact of G on the posterior over policies. For a derivation of these update equations, see Appendix in (Sales et al., 2019).

Note that the cyan line in the dopamine plot corresponds to the stable expected free energy precision value (with a hypothesized link to tonic dopamine levels), as opposed to the rate of change (in black; with a hypothesized link to phasic dopamine responses). To help the reader gain a better intuition for the dynamics of $\boldsymbol{\beta}$ updates, we have provided supplementary code (**EFE_Precision_Updating.m**), which allows the reader to specify the number of policies, values for the vectors E, F, and G, and prior values for β, and then simulate these updates. Note that, unlike the example shown above, this code also divides the update value ($\dot{\boldsymbol{\beta}}$) by a standard scaling parameter of 2, which reduces the magnitude of each update (i.e., where smaller updates can promote more stable convergence). For instance, in the example of a single update shown above, this would instead lead to $\boldsymbol{\gamma} = 1.24$. **Figure 8** also illustrates a helpful geometric interpretation of the factors that determine the direction of $\boldsymbol{\beta}$ updates. Namely, when the difference vector ($\boldsymbol{\pi} - \pi_0$) and the $-G$ vector point in a similar direction (i.e., an angle of less than 90° apart), the dot product of the two will result in an increase in $\boldsymbol{\gamma}$. The fact that these vectors point in the same direction is a way to visualize how new observations (through F) have provided evidence supporting the reliability of G, and therefore increase its precision weighting. In contrast, when the angle separating these vectors is greater than 90° apart this suggests that G is less reliable; its precision ($\boldsymbol{\gamma}$) is therefore reduced and it contributes less to the posterior distribution over policies ($\boldsymbol{\pi}$).

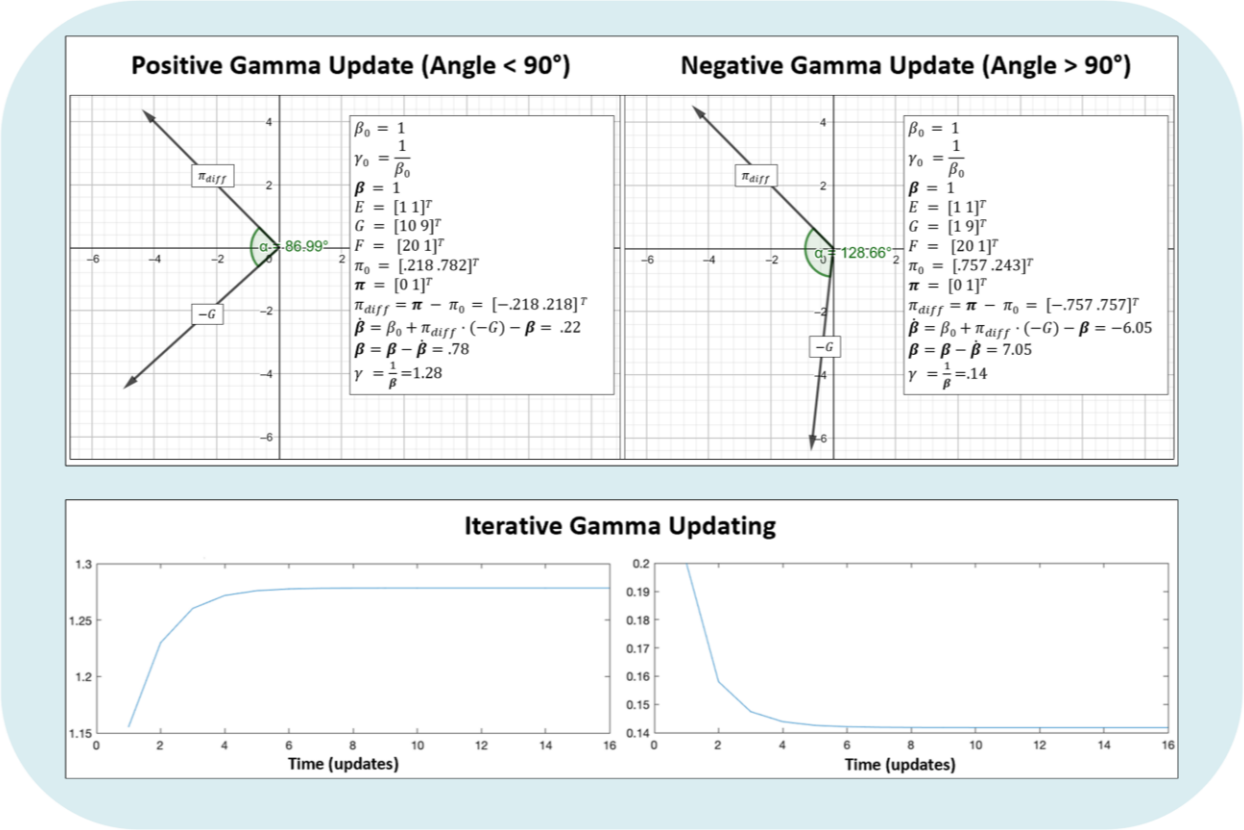

Figure 8. Illustration of a geometric interpretation of the factors contributing to updates in expected free energy precision estimates ($\boldsymbol{\gamma}$). In these examples we include two policies and specify the values of priors over policies (E), expected free energy over policies (G), and the variational free energy following a new observation (F). Based on these, a prior and posterior over policies (π_0 and $\boldsymbol{\pi}$) are computed (calculations shown in each of the top panels). Then updates are computed for the prior on expected free energy precision (from β_0 to $\boldsymbol{\beta}$), resulting in an update in $\boldsymbol{\gamma}$ – based on the dot product between the difference vector for prior and posterior policy beliefs ($\boldsymbol{\pi_{Diff}}$) and the $-G$ vector. The plots in the top panels show cases where $\boldsymbol{\gamma}$ is positively updated (left) and negatively updated (right). In the case on the left, the two vectors ($\boldsymbol{\pi_{Diff}}$ and $-G$) point in a similar direction (i.e., less than 90° apart), which represents a way to visualize how new observations (through F) provide evidence for the reliability of G (leading to an increase in its precision weighting $\boldsymbol{\gamma}$). In the case on the right, the vectors are greater than 90° apart, providing evidence against the reliability (leading to a decrease in its precision weighting $\boldsymbol{\gamma}$). Note that, for reasons of clarity, the actual endpoints of the vectors shown here are not the actual values of $\boldsymbol{\pi_{Diff}}$ and $-G$; they instead correspond to scaled values of these vectors, which makes them similar in length and more clearly illustrates the angle separating them. The bottom panels show iterative updating of $\boldsymbol{\gamma}$ (i.e., via updates in $\boldsymbol{\beta}$; 16 updates, as is done per time point [observation] in a trial in the **supplementary code** and in the standard routines) until a stable posterior estimate is reached (i.e., similar to prediction error minimization dynamics for the state and outcome predictions errors described earlier). It is these

values, and their rate of change, that are associated with dopamine in the neural process theory discussed in Section 4. These simulations can be reproduced using the **EFE_Precision_Updating.m** code provided in **supplementary materials**.

Table 3. Output fields for spm_MDP_VB_X_tutorial.m simulation script

MDP Field	Model Element	Structure	Description
MDP.F	Negative variational free energy of each policy over time.	Rows = policies. Columns = time points.	Negative variational free energy of each policy at each time point in the trial. For example, if there are 2 policies and 6 time points there will be a 2x6 matrix containing the negative variational free energy of each policy at each point in the trial.
MDP.G	Negative expected free energy of each policy over time.	Rows = policies. Columns = time points.	Negative expected free energy of each policy at each time point in the trial. For example, if there are 2 policies and 6 time points there will be a 2x6 matrix containing the negative expected free energy of each policy at each point in the trial.
MDP.H	Negative total variational free energy over time.	Columns = time points.	Total negative variational free energy averaged across states and policies at each time point. For example, if there are 8 time points there will be a 1x8 row vector containing the total negative free energy at each time point.

MDP.Fa	Negative free energy of parameter 'a' (if learning A matrix). There are also analogous fields if learning other matrices/vectors (e.g., MDP.Fd for learning the parameters of the D vector, etc.).	Columns = number of outcome modalities or number of hidden state factors.	KL divergence between the parameters of the matrix/vector that is being learned at the beginning of each trial and at the end of each trial. Each column in the vector represents either an outcome modality (in the case of the A matrix) or a hidden state factor) in the case of the B matrix and D vector).
MDP.O	Outcome vectors	Rows = outcome modalities. Columns = time points.	Vectors (one per cell) specifying the outcomes for each modality at each time point. Observed outcomes are encoded as 1s, with 0s otherwise.
MDP.P	Probability of emitting action	Rows = one per controllable state factor. Columns = actions. Third dimension = time point.	The probability of emitting each particular action, expressed as a softmax function of a vector containing the probability of each action summed over each policy. For example, assume that there are two possible actions, with a posterior over policies of [.4 .4 .2], with policy 1 and 2 leading to action 1, and policy 3 leading to action 2. The probability of action 1 and 2 is therefore [.8 .2]. This vector is then passed through another softmax function controlled by the temperature parameter alpha, which by default is extremely large (alpha = 512). This leads to the

			deterministic selection of the action with the highest probability.
MDP.Q	Posteriors over states under each policy at the end of the trial.	1 cell per state factor. Rows = states. Columns = tau. Third dimension = policy number.	Posterior probability of each state conditioned on each policy at the end of the trial after successive rounds of updating at each time point.
MDP.R	Posteriors over policies.	Rows = policies. Columns = time points.	Posterior over policies at each time point.
MDP.X	Overall posteriors over states at the end of the trial. These are Bayesian model averages of the posteriors over states under each policy.	1 cell per state factor. Rows = states. Columns = time points.	This means averaging the posteriors after weighting each state by the posterior probability of the associated policy.
MDP.un	Neuronal encoding of policies	1 cell per policy dimension. Rows = policies. Columns = iterations of variational Bayes/message passing (16 per time point). For example, 16 iterations, and 8 time points gives a vector with 128 columns).	Simulated neuronal encoding of the posterior probability of each policy at each iteration of message passing.
MDP.vn	Neuronal encoding of state prediction errors.	1 cell per state factor. Rows = iterations of variational Bayes/message passing (16 per time point). Columns = states.	Bayesian model average of prediction error averaged over each policy at each message passing iteration.

		Third Dimension: time point the belief is *about* (τ). Fourth Dimension: time point the belief is *at* (t).	
MDP.xn	Neuronal encoding of hidden states.	1 cell per state factor. Rows = iterations of variational Bayes/message passing (16 per time point). Columns = states. Third Dimension: time point the belief is *about* (τ). Fourth Dimension: time point the belief is *at* (t).	Bayesian model average of the normalized firing rate averaged over each policy at each message passing iteration.
MDP.wn	Neuronal encoding of precision (tonic).	Rows = iterations of variational Bayes/message passing (16 per time point). For example, if there were two time points in a trial this would be 1 column with 32 rows.	Variational updates for the posterior value of beta ($\boldsymbol{\beta}$), which reflect the expected precision of the expected free energy over policies at each iteration of variational Bayes.
MDP.dn	Neuronal encoding of dopamine responses (phasic) reflecting the rate of change of beta ($\dot{\boldsymbol{\beta}}$).	Rows = iterations of variational Bayes/message passing (16 per time point). For example, if there were two time points in a trial this would be 1 column with 32 rows.	This variable reflects the rate of change in the expected precision of expected free energy over policies ($\dot{\boldsymbol{\beta}}$) at each iteration of variational Bayes over successive time points.
MDP.rt	Simulated reaction times.	Columns = time points.	Computation time (i.e., time to convergence) for each round of message passing and action selection.

As mentioned earlier, however, optimal information-seeking (in the sense of maximizing preferred outcomes in the long-run) depends on having the right balance of reward value and information value. To illustrate this, **Figure 7B** shows simulations in which the magnitude of the preference distribution for a win has been multiplied by 2: $\boldsymbol{C}\{\mathbf{2}\}(\mathbf{3},:) = [\mathbf{0\ 8\ 4}]$**.** As can be seen there, the model instead decided to make a guess right away about which machine will win (in this case, choosing right) and unfortunately observed a loss (bottom left, middle sub-panel). As can be seen in the upper right, its confidence in the left vs. right action is equal (equally gray over each). As can be seen in the upper left, the model's posterior over states shows high confidence that it was in fact in the 'left-better' context at every time point, because (retrospectively) this must have been the case if it lost after choosing the machine on the right.

4. Neural Process Theory

4.1 Assumptions and Common Depictions

In many active inference papers, you will see neural process theory figures depicting a series of update equations combined with columns of 'ball' neurons and example synaptic connections labeled with model elements (as in **Figure 9** here). This is an example of one *possible* neural implementation of active inference, and it should only be taken as illustrative of the general biological plausibility of the theory.

To make this more interpretable, we will here walk the reader through the depiction step-by-step. In this neural network, each column of neurons (in this case, 3 columns) represents beliefs and prediction errors *about* each point in time (from left to right, indicated by subscripts for $\tau = 1,2,3$). With each new observation, beliefs about all time points (i.e., about the past, present, and future) are updated, corresponding to changes in neural activation across all neurons. The upward arrows from observations (purple nodes at the bottom) to layer 3 (denoted by $\varepsilon_{\pi,\tau}$ for state prediction errors) are depicted as conveying excitatory (red) observation signals (e.g., sensory input) to granular cells in each cortical column, where these observations can differ at each time point. The receiving (pink) state prediction error neurons calculate their prediction errors by combining observation signals with prediction signals from the state representations in the cyan neurons of layer 2 (supragranular neurons denoted by $s_{\pi,\tau}$ for state representations). Note that excitatory (red) downward signals from these neurons to layer 3 are conveyed both forward (from the $\tau = 1$, left neurons) and backward (from the $\tau = 3$, right neurons) – indicating both prospective and retrospective predictive influences on state representations *about* a time point. In contrast, inhibitory (blue) signals are conveyed by these state representation neurons to layer three neurons for the current time point, leading to minimization of prediction error when predictions from state representations match observation signals.

Note next that each of these state and state prediction error representations are calculated in parallel for each policy (denoted by one example neural column in front of another). The top (red) layer 1 neurons, however, do not have another set of neurons behind them. This is because they do a Bayesian model average as an overall best guess about states. They specifically do this by taking state representations for each policy, multiplying them by the probability of that policy, and then averaging

them to get a final posterior over states. This is accomplished by the policy representation (π) neuron on the left (meant to depict a subcortical neural population), the signals from which multiply (via the green modulatory connection) the excitatory (red) signals from the state representations for each policy in layer 2. The policy representation (π) neuron(s) are in turn inhibited by the expected free energy (G; i.e., greater expected free energy reduces the probability of a policy), where the influence of G on π is modulated by the expected free energy precision term (γ), depicted here as being conveyed by subcortical dopamine neurons. G neuron activity is increased by outcome prediction errors (ζ) in layer 5, multiplied (green connections) by the probability of those outcomes (cyan neurons in layer 4) – where the outcome prediction errors reflect the difference between preferred outcomes and predicted outcomes (from layer 4; downward excitatory connections between layers 4 and 5), and where those predicted outcomes are in turn based on state representations in layer 2 (excitatory red connections from layer 2 to layer 4 ad 5).

The policy representation (π) neuron(s) then entail actions (u) at each time point. Based on this description, we assume the reader should be able to follow the equations on the left to identify each associated connection in the network on the right. Note that this figure only shows example connections, assuming two policies and three time points. However, the basic idea is that, if each excitatory connection corresponds to addition, each inhibitory connection corresponds to subtraction, and each modulatory connection corresponds to multiplication, then each of the update equations on the left (in the 'belief updating', 'actions selection', and 'precision' boxes; see further below for learning) can be implemented in a straightforward manner within a relatively simple neural network. It is worth mentioning that these update equations are not always presented in identical form in such figures across the literature. However, these variations are either algebraically equivalent or have been presented with or without certain elements (e.g., with or without learning, with or without expected free energy precision, etc.), depending on the goals of the paper.

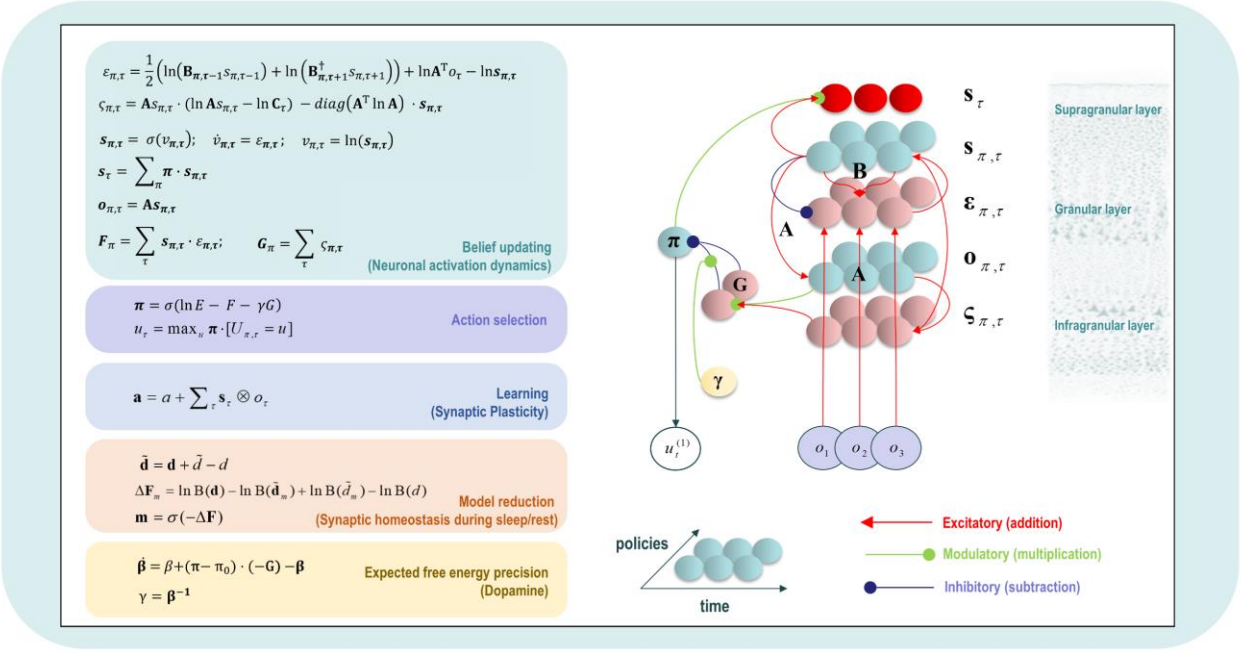

Figure 9. Common depiction of the neural process theory associated with active inference. This includes the update equations on the left and an example neural network that could implement them on the right (only exemplar synaptic connections are shown to avoid visual clutter). See main text for an in-depth walk-through. As briefly discussed in the text, the 'Model reduction' equations on the left correspond to a theory of the function of sleep associated with active inference (in this example, model reduction with respect to the number of hidden states in D based on the values of its concentration parameters d). In these equations, B indicates a beta function, and $\mathbf{m}$ is the posterior probability of different possible models with differing complexities. Here, the posterior distribution over initial states ($\mathbf{d}$) is used to assess the difference in the evidence ($\mathbf{\Delta F}$) it provides for the number of hidden states in one's current model and other possible models characterized by fewer hidden states. Prior concentration parameters are shown in italics, posteriors in bold, and those priors and posteriors associated with the reduced model are equipped with a tilde (**~**). As stated in the text, the details of this aspect of active inference is beyond the scope of this tutorial, but we leave it in the figure to point the interested reader to relevant literature on this topic.

We also briefly note the equations in the 'Model reduction' box on the left side of **Figure 9**, which we do not discuss in detail, as they are beyond the scope of this tutorial. These equations are reproduced from Smith, Schwartenbeck, Parr, & Friston (2020), and we leave them in the figure here only to point the interested reader to other literature describing this aspect of the process theory in more detail (Bucci & Grasso, 2017; Friston, Lin, et al., 2017; Hobson & Friston, 2012; Hobson, Hong, & Friston, 2014; Smith, Schwartenbeck, Parr, & Friston, 2020; Tononi & Cirelli, 2014). In short, this literature suggests that during sleep (and waking rest) the brain can also minimize *VFE* by finding simpler models (with fewer parameters) to account for its previous experience. This can occur in part via a 'synaptic downscaling' process – known to occur during sleep – in which synaptic connection strength changes (that have accrued during recent learning) are subsequently attenuated, which can be helpful in removing any synaptic changes driven by noise (e.g., uninformative coincidences in the presence of multiple stimuli). Although we do not cover it here, we note that the **spm_MDP_VB_X.m** script (and our tutorial version) does have an additional '**BMR**' option that can be turned on, which will implement Bayesian model reduction by calling a further SPM script written to simulate this proposed function of sleep (**spm_MDP_VB_sleep.m**).

Now that we have introduced the structure of the neural process theory, we will now present the resulting neuronal dynamics. In addition to behavior, the **spm_MDP_VB_X_tutorial.m** function also generates neural response simulations based on these dynamics. In this implementation, the firing rates of the neuronal populations representing states (whose location in the brain would depend on the task) encode the various probability distributions in the model (e.g., a higher firing rate in one population would encode a higher probability of the 'left-better context' vs. the 'right-better context' in the explore-exploit task model we constructed above). Changes in firing rates are driven by minimizing the state and outcome prediction errors presented earlier. Simulated electrophysiological responses are based on the rates of change (derivatives) in firing rates during prediction-error minimization (interpreted as local field potentials [LFPs] or event-related potentials [ERPs] depending on the context). Recall again that beliefs are maintained *about* each time point *at* each time point. As such, separate neural populations are postulated to encode beliefs about states at each time point (which are updated with each new observation over time).

Next, synaptic inputs with different strengths are postulated to carry the conditional probabilities encoded in each of the matrices described above. This means that, for example, activity levels in neuronal populations encoding probabilities over states are updated by ascending (state prediction error) signals, based on synaptic weights encoding the amount of evidence that each possible observation provides for each possible hidden state (i.e., entries within the **A** matrices). Changes in these synaptic strengths occur during learning. Although not yet discussed, learning corresponds to updating the entries in the matrices and/or vectors that define the generative model (e.g., changing the probabilities in the **A** or **B** matrices, or in the D or E vectors). These updates occur after each trial, depending on the pattern of observations, belief updates, and/or policies chosen on that trial (this will be covered further below). The important point here, however, is that the value of each entry in particular matrices or vectors in a model can be thought of as corresponding to the strength of a synaptic connection between two neurons, and that such connections can be updated over repeated trials (i.e., over a slower timescale than perception).

An available SPM function for plotting single-trial neural simulations can be run by inputting the following into MATLAB:

$$\mathbf{spm\_figure}('\mathbf{GetWin}','\mathbf{Figure\ 2}');\ \mathbf{clf};\ \mathbf{spm\_MDP\_VB\_LFP}(\mathbf{MDP});\ \mathbf{subplot}(\mathbf{3},\mathbf{2},\mathbf{3})$$

Setting the variable $\boldsymbol{Sim}$ in the accompanying tutorial code (i.e., **Step_by_Step_AI_Guide.m**, line 51) to $\boldsymbol{Sim} = \mathbf{1}$ will also generate simulation plots using this function.

Based on the current model specification, a representative plot of simulation results is shown in **Figure 7C**, based on the single trial depicted in **Figure 7A**. The *top-left* panel depicts the belief updates *at* each time point t (columns) *about* each time point τ (rows). As before, darker indicates higher probability. In this case, at the first time point (column 1), the model is fully uncertain about the state at each time point. At the second time point (column 2), when the model receives the hint, it becomes highly confident that it was, currently is, and will be in the 'left-better' state (i.e., based on its transition beliefs that this state does not change within a trial). These beliefs remain true at time point 3 (upon observing the expected win). The *top-right* panel depicts these belief updates (i.e., changes in beliefs over time about the state each time point; 3 states by 3 time points = 9 probability distributions in total). These belief updates are depicted as traces of changes in neural firing rates, with 2 firing rates per distribution (i.e., encoding the probability of the left- vs. right-better context), resulting in 18 firing rate traces in total (note that, due to overlap, not all 18 traces are visible in this example). The *bottom-left* plot depicts this same information, but here displayed in terms of a simulated raster plot (i.e., one tick per action potential per neuron in a simulated population). The *middle-right* panel depicts predicted local field potentials (or event-related potentials), which reflect the rate of change in the simulated firing rates. The *middle-left* panel depicts the neural responses associated with context state beliefs before (dotted line) and after (solid line) filtering at 4 Hz, superimposed on a time-frequency decomposition of the local field potential (averaged over all simulated neurons). This type of plot has been used in previous work to explain how/why simulated depolarization in specific frequency ranges may coincide with specific stimulus-induced neural responses (Friston, FitzGerald, et al., 2017). The *bottom-right* panel depicts simulated dopamine

responses (as also depicted in a slightly different way in the previous plotting routine). These correspond to updates in the β parameter (see last row of **Table 2**) with each new observation.

This plotting function also has several options as additional function entries and outputs as follows:

$[\mathbf{u}, \mathbf{v}] = \mathbf{spm\_MDP\_VB\_LFP}(\mathbf{MDP}, \mathbf{UNITS}, \mathbf{FACTOR}, \mathbf{SPECTRAL})$.

UNITS: a matrix with 2 rows and one or more columns. The first row indicates which hidden state(s) to plot over time for the specified state factor. The second row specifies the time point being represented. For example: $\boldsymbol{UNITS} = [\mathbf{1\ 2\ 1\ 2};\ \mathbf{1\ 1\ 3\ 3}]$ would plot firing rates for the first two hidden states of the selected state factor over time with regard to beliefs about time points 1 and 3. By default, all units are selected.

FACTOR: a single number denoting which state factor to plot (default = 1).

SPECTRAL: either a 0 or 1 (default = 0). If one, the *top-left* plot is replaced by a plot of the power of neural responses in different frequency ranges.

The optional **outputs u and v** correspond to the vectors of simulated local field potentials and firing rates, respectively (for selected units). As was mentioned above, the local field potentials correspond to the temporal derivative of the firing rates, while the firing rates reflect the magnitude of posterior beliefs over each state at each epoch of gradient descent.

5. Modelling Learning

5.1 Technical Introduction to Dirichlet Priors (optional)

For the reader without a strong mathematical background, this section may be challenging. We encourage all readers to attempt it, but complete understanding of this section will not be required to follow subsequent sections. The non-interested reader can safely skip to the non-technical description below.

Learning in active inference is formulated in terms of a Dirichlet-categorical model. Specifically, Bayesian inference is performed using a categorical distribution (which was discussed earlier) as the likelihood, and a Dirichlet distribution as the prior. The Dirichlet distribution is a distribution defined over a vector of values that sit on the interval $[0, 1]$, and sum to one. That is, the values of the vector have the same properties as a probability distribution. As such, the Dirichlet distribution is often described as a *distribution over a distribution*. In this case, it can be used to encode beliefs about model parameters (e.g., confidence in parameters in the likelihood or transition matrices of a POMDP).

The Dirichlet distribution is used as the prior over the parameters of the categorical distribution because it is the conjugate prior for the categorical distribution. This means that if we multiply a categorical distribution by a Dirichlet distribution, and then normalize to obtain the posterior distribution over the parameters of the categorical distribution, we end up with another Dirichlet distribution – allowing it to be used as a prior in the next round of inference. This allows active

inference agents to sequentially update their beliefs about model parameters as they receive new observations. The Dirichlet distribution, denoted $Dir(\theta|\alpha)$, is defined as follows:

$$p(\boldsymbol{\theta}|\boldsymbol{\alpha}) = Dir(\boldsymbol{\theta}|\boldsymbol{\alpha}) = \frac{\Gamma(\sum_{k=1}^{K} \alpha_k)}{\prod_{k=1}^{K} \Gamma(\alpha_k)} \prod_{k=1}^{K} \theta_k^{\alpha_k - 1}$$

Where $\frac{\Gamma(\sum_{k=1}^{K} \alpha_k)}{\prod_{k=1}^{K} \Gamma(\alpha_k)}$ is a normalization constant that ensures the distribution sums to one, and Γ denotes the gamma function (for a brief introduction to the gamma function see **supplementary materials**). The variable $\boldsymbol{\theta} = (\theta_1, \dots, \theta_K)$ is a vector of length K containing the parameters of a categorical distribution, and $\boldsymbol{\alpha} = (\alpha_1, \dots, \alpha_K)$ is the set of **concentration parameters** of the Dirichlet distribution, which satisfy the condition that $\alpha_i > 0$. The gamma function is used in the normalization constant to account for the combinatorics of drawing a random variable from a categorical distribution. That is, it counts the number of ways in which we can place the variable θ in one of K mutually exclusive states. Similarly, the categorical distribution is defined as:

$$p(\boldsymbol{x}|\boldsymbol{\theta}) = Cat(\boldsymbol{x}|\boldsymbol{\theta}) = \frac{1}{x_1! \, x_2! \dots x_K!} \prod_{k=1}^{K} \theta_k^{x_k}$$

Where $\boldsymbol{x} = (x_1, \dots, x_K)$ is a categorical variable that occupies one of K mutually exclusive states (e.g., $\boldsymbol{x} = [0 \quad 1 \quad 0]^T$). Here $\boldsymbol{\theta} = (\theta_1, \dots, \theta_K)$ are the parameters of the distribution and satisfy the conditions $\theta_k \geq 0$, and $\sum_k \theta_k = 1$. The term $\frac{1}{x_1! x_2! \dots x_K!}$ is the normalisation constant.

If we multiply the Dirichlet and categorical distributions to arrive at the posterior distribution over the parameters $\boldsymbol{\theta} = (\theta_1, \dots, \theta_K)$ of the categorical distribution, we obtain the following (ignoring the normalization constant for the sake of brevity):

$$p(\boldsymbol{\theta}|\boldsymbol{x}, \boldsymbol{\alpha}) = Dir(\boldsymbol{\theta}|\boldsymbol{x}, \boldsymbol{\alpha} + \boldsymbol{x}) \propto \prod_{k=1}^{K} \theta_k^{\alpha_k + x_k - 1}$$

Notice that this has exactly the same form as the prior defined above, except that we have added a 'count' (i.e., x_k) of 1 for the observed variable and a 'count' of 0 is added to the concentration parameters of the distribution that correspond to the non-observed variables. It is the concentration parameters of the Dirichlet distributions in the POMDP structure that are updated during learning. The exact way they are updated depends on the model element in question (i.e., **A** or **B** matrix, or D vector) which we discuss in more intuitive terms below. But, as an initial example, the resulting updates for an example D vector become:

$$p(D) = Dir(d)$$

$$d = p(s_{\tau=1}) = [d_1 \quad d_2]^T$$

$$d_{trial} = \omega \times d_{trial-1} + \eta \times \boldsymbol{s}_{\tau=1}$$

Here, the concentration parameters for D are denoted by lowercase d, and d_1, d_2 are the individual parameters in the vector to be updated. The eta (η) parameter is a **learning rate** (scalar from 0-1), which controls how much posteriors over initial states ($\boldsymbol{s}_{\tau=1}$) on each trial update d on the subsequent trial. A higher value for η indicates faster learning, such that the agent will become confident in beliefs more quickly (which can lead to a quicker switch from information-seeking to reward-seeking behavior). The omega (ω) parameter is a **forgetting rate** (scalar from 0-1), which controls how strongly recent experience is weighted relative to experience in the more distant past. A higher value for ω indicates less forgetting, such that recent and remote past experience are weighted equally. A strong amount of forgetting (i.e., a low value for ω) is appropriate if one believes the statistics of the environment are unstable and frequently change. For a more detailed introduction to the Dirichlet-categorical model, see (Tu, 2014).

5.2 Non-Technical Continuation on Dirichlet Priors

Although the form of the Dirichlet distribution and the associated update equations can seem complex (readers who skipped the previous section may glance briefly above), the resulting learning process turns out to be quite intuitive. Essentially, it just involves **adding counts** to a matrix or vector based on posterior beliefs, where larger numbers of counts indicate higher confidence. For those who skipped the technical section, counts are also modulated (multiplied) by a **learning rate** (η) that takes values between 0 and 1. For example, if you start out with an initial state prior of $d\{1\} = [0.5\ 0.5]'$ on the first trial, and your posterior belief at the end of the trial was that you were in state 1 (with probability = 1), then, assuming the learning rate $\eta = 1$, your prior on the second trial would become $d\{1\} = [1.5\ 0.5]'$. If this happened 3 more times, then it would become $d\{1\} = [4.5\ 0.5]'$. In cases of uncertainty, you instead add *proportions* of counts. For example, if you start out with an initial state prior of $d\{1\} = [1\ 1]'$ on the first trial, and your posterior belief at the end of the trial was $\boldsymbol{s} = [0.7\ 0.3]'$, then your prior would be updated on the second trial to be $d\{1\} = [1.7\ 1.3]'$. During within-trial inference, these distributions are put through a softmax function so that they retain their same shape but again add up to 1. However, larger numbers indicate greater confidence in the shape of the distribution. This can be seen by comparing $d\{1\} = [1\ 1]'$ to $d\{1\} = [50\ 50]'$. While both distributions have the same (in this case flat) shape, it would take *many more* (new) observations to meaningfully change the shape of the second distribution compared to the first. For example, after one further trial with a precise posterior over state 1, the resulting shape of the distribution $d\{1\} = [2\ 1]'$ has changed quite a bit more than $d\{1\} = [51\ 50]'$. This is an important aspect of active learning because it means that the initial (prior) counts determine how 'open' an agent is to new experience. Typically, in a novel environment or task, the initial counts are set to very low values – so that experience has a substantial effect on an agent's prior beliefs. This makes inference and planning more context-sensitive, as opposed to an agent with high initial counts who is 'stuck in its ways' and would require much more evidence to 'change its mind'. As mentioned above, the agent also has a learning rate η that controls how quickly it gets 'stuck in its ways' during learning (this also influences how quickly the agent ceases to select information-seeking policies; see below for more details). For example, if $\eta = 0.5$, then an update from $d\{1\} = [1\ 1]'$ after inferring state 1 would not lead to $d\{1\} = [2\ 1]'$ as shown above. Instead, it would be $d\{1\} = 1 \times [1\ 1]' + 0.5 \times [1\ 0]' = [1.5\ 1]'$ (see equation for learning rate at the end of the previous section). Thus, counts (and hence confidence) will increase more slowly after each trial.

As introduced in the previous technical section, counts are also modulated (multiplied) by a **forgetting rate** (ω) that takes values between 0 and 1. This parameter controls how strongly recent experience is able to "over-write" what one has learned in the more distant past. A value of $\omega = 1$ indicates no forgetting (i.e., recent experience is unable to over-write what has been learned previously), while values less than 1 allow increasing levels of forgetting (essentially, with each new observation the agent becomes less confident in what it has previously learned). This is important because, as counts increase during learning, an agent's beliefs can become rigid and resistant to change, which is suboptimal in changing environments. Higher counts also reduce information-seeking, because the agent is highly confident in its beliefs (described in more detail below), which further hinders the opportunity to learn that (and how) the environment has changed. As such, if an agent believes that the environment is volatile (e.g., that the probabilities of rewards under each policy can change over time), then it is appropriate to adopt a low value for ω. In other words, a low value for ω can be understood as encoding an agent's prior belief that the contingencies in the world are unstable. It is worth noting that inferences about changes in the environment (and about the volatility of the environment) could also be implemented in a more principled manner in a hierarchical model (for one example of a model with dynamically updated beliefs about environmental volatility, see the Hierarchical Gaussian Filter; (Mathys et al., 2014)). However, including the simpler forgetting rate parameter described here could be sufficient for modelling task behavior in many cases.

To get an intuition for how this works, glance back at the equation for learning at the end of the previous section and then consider a case where $d\{1\} = [50\ 50]'$ and $\eta = 1$. If $\omega = 1$, and the agent infers it is in state 2, then the update will be $d\{1\} = 1 \times [50\ 50]' + 1 \times [0\ 1]' = [50\ 51]'$. In contrast, if $\omega = 0.1$, then the update will be $d\{1\} = 0.1 \times [50\ 50]' + 1 \times [0\ 1]' = [5\ 6]'$. In this latter case, the agent therefore becomes much less confident in its prior beliefs, and the shape of the posterior (Dirichlet) distribution is changed to a greater degree at the end of that trial.

As a slightly more complex example of learning, the resulting updates for an example **A** matrix become:

$$p(\mathbf{A}) = Dir(\boldsymbol{a})$$

$$\boldsymbol{a} = p(o|s) = \begin{bmatrix} a_1 & a_2 \\ a_3 & a_4 \\ a_5 & a_6 \end{bmatrix}$$

$$\boldsymbol{a}_{trial} = \omega \times \boldsymbol{a}_{trial-1} + \eta \times \sum_{\tau} o_\tau \otimes \boldsymbol{s}_\tau$$

Here, the concentration parameters for **A** are denoted by lowercase $\boldsymbol{a}$, and $a_1 .., a_6$ are the individual parameters in the matrix to be updated. The $\otimes$ symbol indicates the Kronecker (i.e., outer) product. This again just involves accumulating (proportions of) counts, modulated by a learning rate and a forgetting rate. But in this case, what is being counted are coincidences between states and observations. For example, assume you have a posterior over states of $\boldsymbol{s} = [1\ 0]$ and you made the observation associated with row 1, $o = [1\ 0\ 0]^T$. Because you believed you were in state 1 when you

observed outcome 1, this indicates that their association in a should increase. That is, a count should be added to a_1 (i.e., the intersection between state 1 and outcome 1) before the subsequent trial. If you instead have a posterior over states of $\boldsymbol{s} = [.7\ .3]$ and you made the observation associated with row 2, $o = [0\ 1\ 0]^T$, this indicates that their association in $\boldsymbol{a}$ should increase proportionally. That is, updates of $a_3 + .7$ and $a_4 + .3$ should occur before the next trial. Analogous update rules apply to the other model parameters ($\mathbf{B}, \mathbf{C}, E$). This general type of 'coincidence detection' learning is analogous to Hebbian synaptic plasticity, where neurons with coincident firing rates increase their synaptic connection strengths (T. H. Brown et al., 2010). As mentioned above, the neural process theory associated with active inference proposes that each concentration parameter can therefore be associated with the strength of a synaptic connection.

Another important change when learning is incorporated is that the expected free energy gains an extra term, depending on which parameter is being learned. This is because learning is also based on minimizing expected free energy. For example, if learning $\mathbf{A}$, the equation for expected free energy becomes:

$$G_\pi = D_{KL}[q(o_\tau|\pi)||p(o_\tau)] + E_{q(s_\tau|\pi)}[\mathrm{H}[p(o_\tau|s_\tau)] - E_{p(o_\tau|s_\tau)q(s_\tau|\pi)}[D_{KL}[q(\mathbf{A}|o_\tau, s_\tau)||q(\mathbf{A})]]$$

$$\approx \sum_\tau \left(\mathbf{A}s_{\pi,\tau} \cdot \left(\ln \mathbf{A}s_{\pi,\tau} - \ln \mathbf{C}_\tau\right) - diag(\mathbf{A}^T \ln \mathbf{A}) \cdot s_{\pi,\tau} - \mathbf{A}s_{\pi,\tau} \cdot \mathbf{W}s_{\pi,\tau}\right)$$

$$\mathbf{W} := \frac{1}{2}(\boldsymbol{a}^{\odot(-1)} - \boldsymbol{a}_{sums}^{\odot(-1)})$$

Note that the $:=$ symbol just means that two things are defined to be equivalent; and the $\odot$ symbol indicates the element-wise power (i.e., separately raising each element in a matrix to the power of some number). The term $\boldsymbol{a}_{sums}$ is a matrix of the same size as $\boldsymbol{a}$ where each entry within a column corresponds to the sum of the values of the associated column in $\boldsymbol{a}$. For example, if $\boldsymbol{a} = \begin{bmatrix} .25 & 1 \\ .75 & 3 \end{bmatrix}$, then $\boldsymbol{a}_{sums} = \begin{bmatrix} .25 + .75 & 1 + 3 \\ .25 + .75 & 1 + 3 \end{bmatrix} = \begin{bmatrix} 1 & 4 \\ 1 & 4 \end{bmatrix}$.

Although this updated equation for EFE may appear complex, it simply indicates that, by adding the new 'novelty' term (i.e., the third [final] term on the right-hand side of the equation: $\mathbf{A}s_{\pi,\tau} \cdot \mathbf{W}s_{\pi,\tau}$), minimizing expected free energy will now also drive information-seeking about parameter values in the $\mathbf{A}$ matrix (i.e., as opposed to simply seeking out information about states). In other words, the agent will also seek out observations to increase confidence in its beliefs about $p(o_\tau|s_\tau)$. Although we do not show them explicitly here, similar terms can also be added if the agent is learning any of the other matrices or vectors in the model (e.g., learning the transition probabilities in $\mathbf{B}$).

In more technical detail, the first equation shows the expression for EFE introduced above, but now with an additional 'novelty' term ($E_{p(o_\tau|s_\tau)q(s_\tau|\pi)}[D_{KL}\left[q(\mathbf{A}|o_\tau, s_\tau)\right||q(\mathbf{A})]\right]$). As novelty is a positive value (and subtracted from the total value), to minimize EFE agents must maximize novelty by seeking out state-observation pairings that maximize the difference in concentration parameters between posterior and prior distributions over $\mathbf{A}$. That is, by maximizing the novelty term agents

move the value of EFE closer to zero. This quantifies the drive or epistemic affordance of finding out 'what would happen if I do that?'.

The size of the term $\mathbf{A}s_{\pi,\tau} \cdot \mathbf{W}s_{\pi,\tau}$ is inversely related to concentration parameter values. When the concentration parameters have large values, this term will have a small value, and when the concentration parameters have small values, this term will have a large value. This term is called 'novelty' because it drives a specific type of information-seeking. In this case, G_π will be minimized by finding the policy that is expected to change the concentration parameters as much as possible. In other words, policies are driven to increase confidence in beliefs about the likelihood function. It follows that when the concentration parameter values are high, the model will become primarily reward-seeking, as it will be highly confident in its beliefs. Analogous dynamics occur when updating concentration parameters for other model parameters.

To make this more concrete, we show a worked example of the novelty term for two **A** matrices. One with small concentration parameter values (i.e., low confidence in beliefs about the outcomes generated by hidden states), and the other with large concentration parameter values (i.e., high confidence in beliefs about the outcomes generated by hidden states).

Small concentration parameter values (low confidence):

$$\boldsymbol{a} = \begin{bmatrix} .25 & 1 \\ .75 & 1 \end{bmatrix};\ \mathbf{A} = \sigma(\boldsymbol{a}) = \begin{bmatrix} .25 & .5 \\ .75 & .5 \end{bmatrix};\ s_{\pi,\tau} = \begin{bmatrix} .9 \\ .1 \end{bmatrix};$$

$$\boldsymbol{a}_{sums} = \begin{bmatrix} .25+.75 & 1+1 \\ .25+.75 & 1+1 \end{bmatrix} = \begin{bmatrix} 1 & 2 \\ 1 & 2 \end{bmatrix};$$

$$\mathbf{A}s_{\pi,\tau} = \begin{bmatrix} .275 \\ .725 \end{bmatrix}$$

$$\mathbf{W} := \frac{1}{2}(\boldsymbol{a}^{\odot(-1)} - \boldsymbol{a}_{sums}^{\odot(-1)})$$

$$\mathbf{W} = \frac{1}{2}\left(\begin{bmatrix} .25^{-1} & 1^{-1} \\ .75^{-1} & 1^{-1} \end{bmatrix} - \begin{bmatrix} 1^{-1} & 2^{-1} \\ 1^{-1} & 2^{-1} \end{bmatrix}\right) = \frac{1}{2}\left(\begin{bmatrix} 4 & 1 \\ 1.3333 & 1 \end{bmatrix} - \begin{bmatrix} 1 & .5 \\ 1 & .5 \end{bmatrix}\right) = \frac{1}{2}\left(\begin{bmatrix} 3 & .5 \\ .3333 & .5 \end{bmatrix}\right)$$

$$= \begin{bmatrix} 1.5 & .25 \\ .1667 & .25 \end{bmatrix}$$

$$\mathbf{W}s_{\pi,\tau} = \begin{bmatrix} 1.5 & .25 \\ .167 & .25 \end{bmatrix}\begin{bmatrix} .9 \\ .1 \end{bmatrix} = \begin{bmatrix} 1.375 \\ .175 \end{bmatrix}$$

$$Novelty = \mathbf{A}s_{\pi,\tau} \cdot \mathbf{W}s_{\pi,\tau} = \begin{bmatrix} .275 \\ .725 \end{bmatrix} \cdot \begin{bmatrix} 1.375 \\ .175 \end{bmatrix} = .505$$

Large concentration parameter values (high confidence):

$$\boldsymbol{a} = \begin{bmatrix} 25 & 100 \\ 75 & 100 \end{bmatrix};\ \mathbf{A} = \sigma(\boldsymbol{a}) = \begin{bmatrix} .25 & .5 \\ .75 & .5 \end{bmatrix};\ s_{\pi,\tau} = \begin{bmatrix} .9 \\ .1 \end{bmatrix};$$

$$\boldsymbol{a}_{sums} = \begin{bmatrix} 25+75 & 100+100 \\ 25+75 & 100+100 \end{bmatrix} = \begin{bmatrix} 100 & 200 \\ 100 & 200 \end{bmatrix};$$

$$\mathbf{A}s_{\pi,\tau} = \begin{bmatrix} .275 \\ .725 \end{bmatrix}$$

$$\mathbf{W} := \frac{1}{2}(\boldsymbol{a}^{\odot(-1)} - \boldsymbol{a}_{sums}^{\odot(-1)})$$

$$\mathbf{W} = \frac{1}{2}\left(\begin{bmatrix} 25^{-1} & 100^{-1} \\ 75^{-1} & 100^{-1} \end{bmatrix} - \begin{bmatrix} 100^{-1} & 200^{-1} \\ 100^{-1} & 200^{-1} \end{bmatrix}\right) = \frac{1}{2}\left(\begin{bmatrix} .04 & .01 \\ .0133 & .01 \end{bmatrix} - \begin{bmatrix} .01 & .005 \\ .01 & .005 \end{bmatrix}\right)$$

$$= \tfrac{1}{2}\left(\begin{bmatrix} .03 & .005 \\ .0033 & .005 \end{bmatrix}\right) = \begin{bmatrix} .015 & .0025 \\ .0017 & .0025 \end{bmatrix}$$

$$\mathbf{W}s_{\pi,\tau} = \begin{bmatrix} .015 & .0025 \\ .00167 & .0025 \end{bmatrix}\begin{bmatrix} .9 \\ .1 \end{bmatrix} = \begin{bmatrix} .01375 \\ .00175 \end{bmatrix}$$

$$Novelty = \mathbf{A}s_{\pi,\tau} \cdot \mathbf{W}s_{\pi,\tau} = \begin{bmatrix} .275 \\ .725 \end{bmatrix} \cdot \begin{bmatrix} .01375 \\ .00175 \end{bmatrix} = .00505$$

In both examples, the policy assigns high probability to occupying state 1 (p = .9). The normalized shape of the distributions for each column in **A** is also the same in both examples. However, the novelty term is larger in the first example where the associated Dirichlet prior $\boldsymbol{a}$ has smaller concentration parameter values (which when subtracted from the total will lead to a lower EFE). This means the agent will learn more (i.e., change its beliefs more) when moving to states where it is less confident in its beliefs (encoded as smaller concentration parameter values). To get a more intuitive sense for these computations, you can reproduce these results and adjust the concentration parameter values in the supplementary script **EFE_learning_novelty_term.m**.

5.3 Simulating Learning

To simulate learning in the explore-exploit model we specified above, we only need a few additions to the code. First, a lowercase version of the to-be-learned model element must be created. For example, to enable learning within the **A** matrix, one must specify an **mdp.a** with the same dimensions as **mdp.A**. The same goes for other parameters (**mdp.b**, **mdp.d**, etc.). Typically, the initial concentration parameters would be set to low-confidence (i.e., low magnitude), flat distributions before learning begins; for example: $\boldsymbol{d}\{\mathbf{1}\} = [.\mathbf{25}\ .\mathbf{25}]'$. Note here that the generative process continues to correspond to the capital-letter matrices (i.e., which will generate the patterns of observations), while the lowercase-letter matrices are now the generative model. Next, we can specify a learning rate and a forgetting rate by setting **mdp.eta** and **mdp.omega** equal to values between 0 and 1. Finally, we need to replicate the **mdp** structure to include many trials; for example, using code such as:

$$\boldsymbol{N\_Trials} = \mathbf{30};$$

$$[\boldsymbol{mdp}(\mathbf{1}{:}\,\boldsymbol{N\_Trials})] \;=\; \boldsymbol{deal}(\boldsymbol{mdp});$$

Then, one can simply run the **mdp** structure through the **spm_MDP_VB_X_tutorial.m** function we have provided as before. Here, we will simulate two different versions of the task. In the first version there are 30 trials, and the better slot machine is the same for all trials (left machine). We will allow the simulated agent to learn prior expectations about which context is more likely (i.e., whether the left or right machine tends to lead to wins more often). To enable this type of learning, we will include **mdp.d**. As shown in the **Step_by_Step_AI_Guide.m** code, we specify low confidence in initial state priors, $\boldsymbol{d}\{\mathbf{1}\} = [.\mathbf{25}\;.\mathbf{25}]'$. We also set the learning rate to $\mathbf{mdp.eta} = .\mathbf{5}$ and the forgetting rate to $\mathbf{mdp.omega} = \mathbf{1}$ (i.e., no forgetting). We then define a "risk-seeking" ($\boldsymbol{RS}$) parameter that corresponds to how strong the preference is to win the higher amount of money in the **C** matrix:

$$\mathbf{C}\{\mathbf{2}\} = \begin{bmatrix} \mathbf{0} & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & -\mathbf{1} & -\mathbf{1} \\ \mathbf{0} & \boldsymbol{RS} & \dfrac{\boldsymbol{RS}}{\mathbf{2}} \end{bmatrix}$$

In our first simulation, we set $\boldsymbol{RS} = \mathbf{3}$ and in our second we set $\boldsymbol{RS} = \mathbf{4}$. We expect the agent will be less information-seeking, and more risk-seeking, in the latter case. As can be seen in **Figure 10** (*top-left*), the $\boldsymbol{RS} = \mathbf{3}$ agent chooses to take the hint on the first several trials, and slowly begins to forego the hint on later trials (with some choice stochasticity). Unexpected losses, shown in the panel just below, also cause the agent to return to 'playing it safe' and again asking for the hint. In contrast, the $\boldsymbol{RS} = \mathbf{4}$ agent chooses to take the hint only once (**Figure 10**, *top-right*) and then takes that as sufficient evidence that the left machine must be the better one (and continues to choose this one throughout, despite occasional losses). The lower panels in the figure show simulated ERPs, simulated dopamine responses, and how beliefs change over time about which context is more likely (darker = higher probability). When the $\boldsymbol{Sim}$ variable is set to $\boldsymbol{Sim} = \mathbf{2}$ in the **Step_by_Step_AI_Guide.m** code, you can reproduce these simulations (and adjust the $\boldsymbol{RS}$ parameter to other possible values).

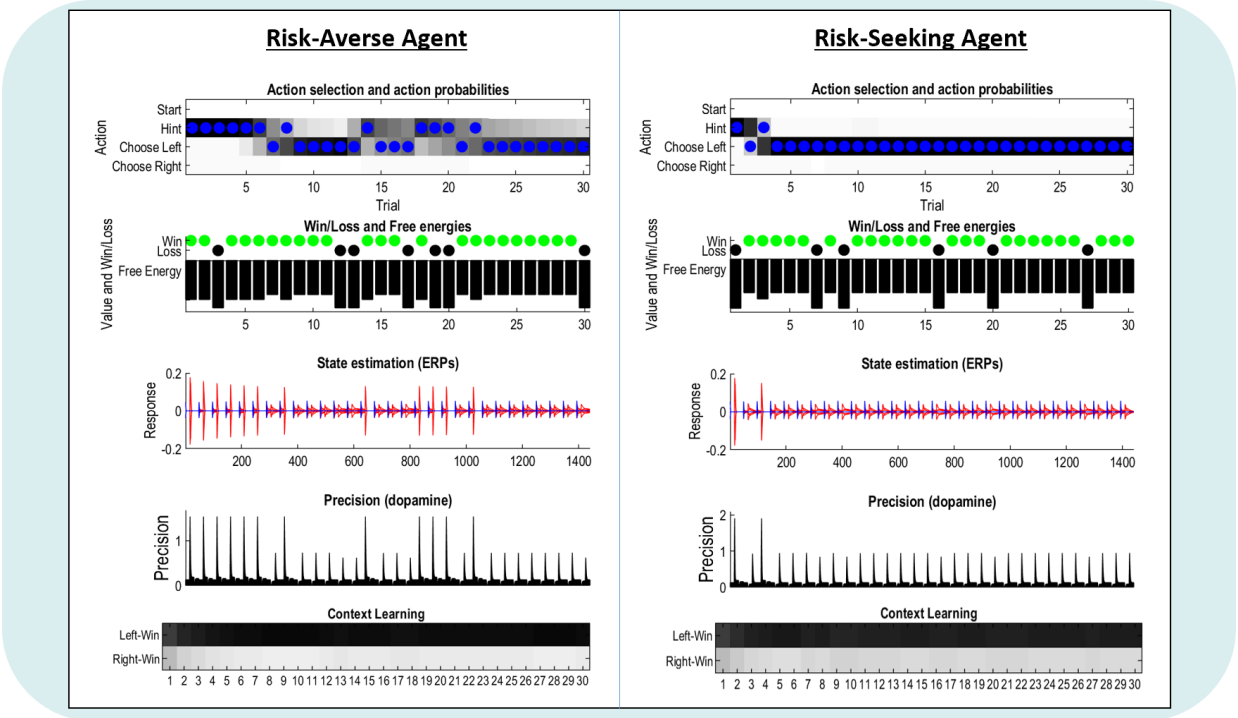

Figure 10. Simulated learning on the explore-exploit task and predicted neuronal responses. See main text for more details. (*Left*) Example simulation of a risk-averse agent learning from repeated trials of the explore-exploit task. The top panel shows that the agent slowly gains confidence that the left machine will always be better, and in choosing that option directly, after taking the hint several times (darker = stronger belief, blue circle indicates chosen action). This agent also often returns to taking the hint after unexpected losses. (*Right*) Example simulation of a risk-seeking agent learning from repeated trials of the explore-exploit task. The top panel shows that the agent only required seeing the hint one time before attempting to pick the correct option directly (and win more money). Note that both agents also show some stochasticity in choice. Wins/losses, free energies, simulated neuronal responses, and changes in prior beliefs about context over time are shown in the lower panels. These simulations can be reproduced by running the $\boldsymbol{Sim = 2}$ option in the supplementary **Step_by_Step_AI_Guide.m** code (although note that, because outcomes are sampled from probability distributions, results will not be identical each time).

Next, we simulate a reversal learning paradigm. Here, there are 32 trials in total. Unbeknownst to the agent, in the first 8 trials the left machine will be better, but in the rest of the trials the right machine will be better. Again, we examine an agent with $\boldsymbol{RS = 3}$ and $\boldsymbol{RS = 4}$. As shown in **Figure 11**, the $\boldsymbol{RS = 3}$ agent chose to take the hint on all trials in this simulation. In contrast, the $\boldsymbol{RS = 4}$ agent quickly locked on to the left machine, but it then returned to taking the hint after the reversal. After several trials of again choosing the hint, it becomes confident in directly choosing the right machine in the final trials. When the $\boldsymbol{Sim}$ variable is set to $\boldsymbol{Sim = 3}$ in the **Step_by_Step_AI_Guide.m** code, you can reproduce these simulations (and adjust the $\boldsymbol{RS}$ parameter to other possible values).

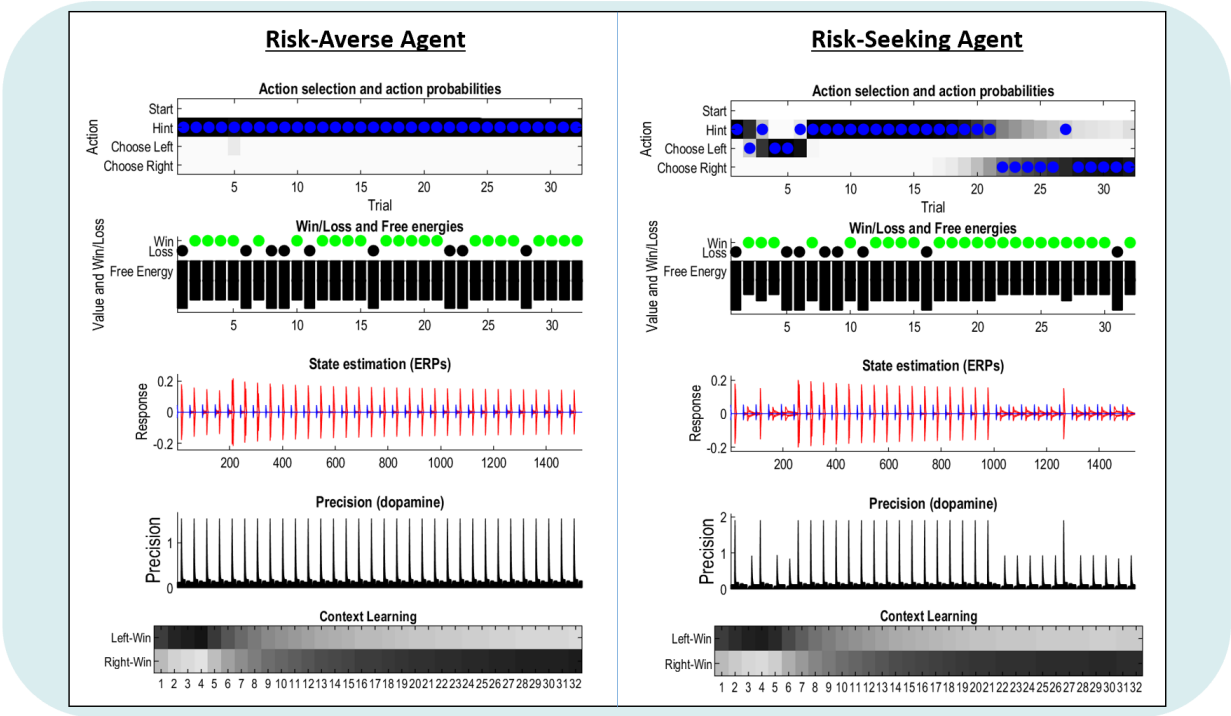

Figure 11. Simulated reversal learning on the explore-exploit task and predicted neuronal responses. Here, the better machine was on the left for the first 8 trials and then on the right for all subsequent trials (without the agent expecting this). See main text for more details. (*Left*) Example simulation of a risk-averse agent, who always chose to take the hint (shown in the top panel; darker = stronger belief, blue circle indicates chosen action). (*Right*) Example simulation of a risk-seeking agent. The top panel shows that the agent quickly became confident that the left machine was better, but then decided to take the hint for many trials after the reversal, before becoming confident in choosing the right machine without the hint. Wins/losses, free energies, simulated neuronal responses, and changes in prior beliefs about context over time are shown in the lower panels. These simulations can be reproduced by running the $\boldsymbol{Sim = 3}$ option in the supplementary **Step_by_Step_AI_Guide.m** code (although note that, because outcomes are sampled from probability distributions, results will not be identical each time).

Each of these example simulations is meant primarily to give readers a sense of how to work with these types of simulations. But there are many other parameters that could be manipulated. In the accompanying MATLAB code, the reader can easily re-run these simulations while changing the learning rate, forgetting rate, action precision, or any other parameters in the model. We encourage the reader to do so to get a sense of the unique influence of different parameters on task behavior. For examples of papers that model learning using active inference, see (Friston, FitzGerald, et al., 2016; Smith, Parr, & Friston, 2019; Smith, Schwartenbeck, Parr, et al., 2020; Smith, Schwartenbeck, Stewart, et al., 2020).

6. Building Hierarchical Models

6.1 Hierarchical Model Structure

Now that we have a clear idea of how to specify a generative model of a behavioral task, and how to interpret the relevant outputs, we will now extend this foundation to build a hierarchical or 'deep temporal' model; for examples, see (Friston et al., 2018; Parr & Friston, 2017b; Smith, Lane, Parr, & Friston, 2019; Whyte & Smith, 2020). The steps are quite similar to what we've already covered, because this primarily just involves building two models, and then placing one below the other. The further step is figuring out how to link the two models together. This is because, in hierarchical models, the states at the lower level are treated as the observations made by the higher level (see **Figure 12**).

Specifically, the posteriors over initial states in the lower-level model (i.e., at the end of the first lower-level trial) are used as single observations in the higher-level model (i.e., at single time points within a higher-level trial). In turn, the priors over initial states (D vector) in the lower-level model (i.e., at the start of a lower-level trial) are also based on the predicted observations generated by hidden states in the higher-level model (i.e., at each time point within a higher-level trial). This means that the higher-level **A** matrix (likelihood mapping) mediates the ascending and descending messages between hierarchical levels. This structure also entails that the higher-level model must operate at a slower timescale than the lower-level model, because each observation in the higher-level model (i.e., each time point in a higher-level trial) corresponds to the results of (i.e., posterior beliefs after) a complete trial in the lower-level model. For example, if there are four time points in a higher-level trial, this means there will need to be a sequence of four lower-level trials (where each lower-level trial could itself have several time points). This is why such models are often called deep temporal models.

This type of model architecture is essential for capturing perceptual phenomena with nested dynamics, or where objects must be recognized before regularities in the behavior of those objects can be detected. For example, to perceive a baseball flying in a leftward direction, a lower-level model would first need to infer the baseball's identity and position (one inferred position per lower-level trial), and a higher-level model would then need to accumulate evidence for a leftward trajectory of motion based on how the baseball's inferred position changes across several lower-level trials. As another example, to recognize a melody, a lower-level model would be needed to infer the presence of each note, and a higher-level model would then be needed to accumulate evidence for a specific melody, based on a specific sequence of inferred notes over time. A further intuitive example is reading, where the first level infers single words, while the second level infers the narrative entailed by the sequences of words (Friston et al., 2018). Note, as soon as we start to use deep or hierarchical generative models, we are essentially relaxing the Markovian assumption by introducing a separation of temporal scales to produce what are known as semi-Markovian models. These are essential for inferring narratives, language, or any deeply structured sequence of state transitions.

Aside from these examples, the hierarchical POMDP setup is quite flexible and can be used to model a wide range of temporally structured phenomena. For example, a policy space could be included at either level alone, or both levels, depending on the target phenomenon to be modelled (e.g., verbal report at a higher level vs. reflexive behavior at a lower level). One could also specify several time

points in each lower-level trial, such that higher-level states generate sequences or trajectories of state transitions at the lower level (i.e., within-trial). In previous work, hierarchical models have been used to model working memory, reading, visual consciousness, and emotional awareness, among other phenomena (Friston, Parr, et al., 2017; Friston et al., 2018; Hesp et al., 2020; Parr & Friston, 2017b, 2018b; Smith, Lane, et al., 2019; Whyte & Smith, 2020). Hierarchical POMDPs also afford further opportunities for simulating neuronal processes. To date, simulations associated with the faster and slower timescales of belief updating have been shown to reproduce an impressive number of task-based electrophysiological findings. For example, empirically observed patterns of event-related potentials (ERPs) associated with specific cognitive and perceptual processes, such as the P300 and mismatch negativity (MMN), emerge naturally in simulations of different experimental paradigms, which supports the face validity of both the model structure and the neural process theory (e.g., (Friston, FitzGerald, et al., 2017; Parr & Friston, 2017b; Whyte & Smith, 2020)).

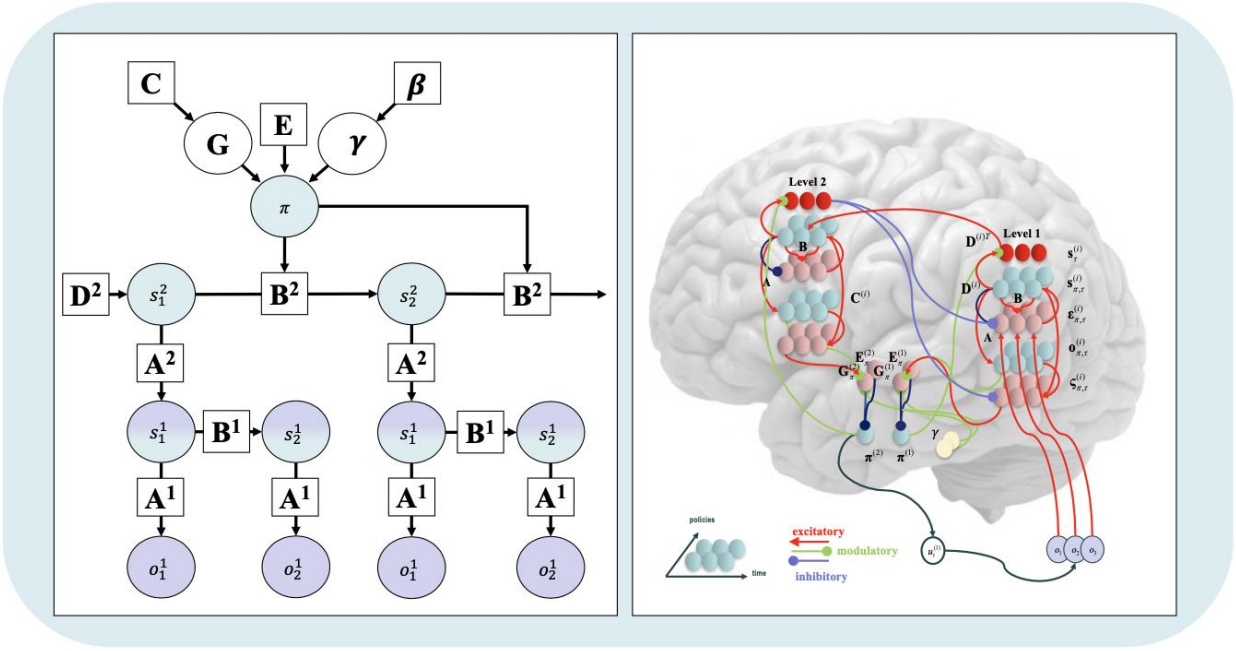

Figure 12. (*Left*) Bayesian network depiction of a 2-level POMDP. Observations depend on hidden states at the first level. In turn, hidden states at the first level depend hidden states at the second level. Specifically, first-level hidden states function as observations for the second level. Equivalently, first-level hidden states are the outcomes generated by hidden states at the second level. In the example shown here, the first level has two state transitions per trial. This entails that the first level has two state transitions for every one state transition at the second level. Thus, beliefs at the second level evolve over a slower timescale. (*Right*) Example neural network implementing the hierarchical POMDP shown on the left.

6.2 Building a Hierarchical Model

As a concrete, empirically relevant example, we will now demonstrate how one could build a hierarchical model to simulate a simplified version of the auditory mismatch "local-global" paradigm introduced in (Bekinschtein et al., 2009). In this paradigm each trial consists of a sequence of five tones (with either low or high frequency), the first four have the same frequency, and the fifth tone either conforms to the predicted pattern (e.g., low-low-low-low-low; *local standard*) or violates the predicted pattern by presenting a higher or lower frequency tone (low-low-low-low-high; *local deviation*). During electroencephalography (EEG), local deviations elicit a mismatch negativity component in ERPs (i.e., a negative component obtained by subtracting "local standard" trials from "local deviation" trials) which appears after approximately 130ms. Importantly, the sequence of local standard and local deviation trials establishes a global pattern that can itself be confirmed (*global standard*) or violated (*global deviation*). For example, several local standard trials in a row followed by an unexpected local deviation trial. Global deviations are known to elicit a P300 ERP component (a positive component that appears after approximately 300ms). Unlike other auditory mismatch paradigms, this design also allows local and global violation responses to be dissociated. That is, the factorial design leads to four conditions *local standard + global standard*, *local standard + global deviation*, *local deviation + global standard*, and *local deviation + global deviation*. For brevity, here we only simulate two of the four possible conditions, *local standard + global deviation*, and *local deviation + global standard*. In our simulated version of the task, we presented an active inference agent with sequences of 4 tones, where each tone could be low or high, in an analogous manner to the empirical task. We then had the agent report whether the last stimulus on each trial was the same as, or different from, the established pattern.

At this point, the reader is encouraged to open the accompanying MATLAB script and follow along in parallel (**Step_by_Step_Hierarchical_Model.m**). As with the previous model, we will start by setting up the sets of hidden state factors at the first level, which in terms of MATLAB code, are defined in terms of priors over initial states.

$$\boldsymbol{D}\{\mathbf{1}\} = [\mathbf{1}\ \mathbf{1}]'$$

$$\boldsymbol{d} = \boldsymbol{D}$$

The specification for $\boldsymbol{D}\{\mathbf{1}\}$ means that there is an equal probability of a high and a low tone (left and right entries, respectively). Note that because the columns of all the vectors and matrices are normalized, $\boldsymbol{D}\{\mathbf{1}\} = [\mathbf{1}\ \mathbf{1}]'$ is equivalent to $\boldsymbol{D}\{\mathbf{1}\} = [.\,\mathbf{5}\,.\,\mathbf{5}]'$. As the simulation involves learning, we also need to separate the generative process from the generative model by including the lower-case d for the generative model. Here we simply set $\boldsymbol{d} = \boldsymbol{D}$, as the agent will also start out with the belief that a high and a low tone are equally probable. However, including d will allow the agent to accumulate concentration parameters (changing the shape of its initial state priors) over trials based on patterns in its observations.

Next, we must specify the likelihood mappings for the first level in the **A** matrix**.**

$$\mathbf{A}\{\mathbf{1}\} = \begin{bmatrix} \mathbf{1} & \mathbf{0} \\ \mathbf{0} & \mathbf{1} \end{bmatrix}$$

This is simply an identity matrix indicating that the tone states correspond 1-to-1 with tone observations (columns [left to right]: high tone, low tone states; rows [top to bottom]: high tone, low tone observations). However, in the generative model we may want to introduce some noise into tone perception. One convenient way to do this is to first specify:

$$\boldsymbol{a} = \mathbf{A}$$

Then, we can use a softmax function to control the expected precision of the state-observation mapping with $precision$ parameter:

$$\boldsymbol{precision} = \mathbf{2};$$

$$\boldsymbol{a}\{\mathbf{1}\} = \boldsymbol{spm\_softmax}(\boldsymbol{precision} * \boldsymbol{log}(\mathbf{A}\{\mathbf{1}\} + \boldsymbol{exp}(-\mathbf{4})));$$

Note that the $\boldsymbol{exp}(-\mathbf{4})$ is simply a very small number added to $\mathbf{A}\{\mathbf{1}\}$ to prevent the possibility of $\boldsymbol{log}(\mathbf{0})$, which is undefined (while -4 is a reasonable value, other values could be chosen). Depending on the value of the $\boldsymbol{precision}$ parameter (higher = more precise), this will result in a likelihood mapping that specifies different amounts of sensory noise. For example:

$$\mathbf{a}\{\mathbf{1}\} = \begin{bmatrix} .\mathbf{92} & .\mathbf{08} \\ .\mathbf{08} & .\mathbf{92} \end{bmatrix}$$

Note that, for clarity, this example shows a lower precision than what results from setting $\boldsymbol{precision} = \mathbf{2}$ in the accompanying tutorial code.

As we are mainly interested in simulating the learning of prior expectations (D vector), we also multiply the generative model parameter for **a** by 100 (an arbitrary large number) to effectively prevent learning in this parameter. This is because we want the level of sensory noise to remain consistent across trials.

Next, we can specify transition probabilities in the **B** matrix as identity matrices, as tones do not change within a lower-level trial.

$$\mathbf{B}\{\mathbf{1}\} = \begin{bmatrix} \mathbf{1} & \mathbf{0} \\ \mathbf{0} & \mathbf{1} \end{bmatrix}$$

Here columns (left to right) are high tone and low tone states at time τ, while rows (top to bottom) are high tone and low tone states at time $\tau + 1$. Here there is no need to separate the generative process from the generative model, so we do not specify a separate **b** matrix.

We do not include preferences or policies at this level, so we now simply assign each variable to the **mdp** structure. For convenience when later linking this to the higher-level model below, we will denote this structure with an '**_1**' as follows:

$$\mathbf{mdp\_1}.\mathbf{D} = \mathbf{D}$$

$$\mathbf{mdp\_1}.\mathbf{d} = \mathbf{d}$$

$\mathbf{mdp\_1.A = A}$

$\mathbf{mdp\_1.a = a}$

$\mathbf{mdp\_1.B = B}$

We then set **MDP**_1 = **mdp_1** and clear **mdp_1**.

Now we move on to specifying the second-level model. To keep the variables separate we will denote each model variable with a '**_2**' for this level.

Here, we will include one hidden state for each possible *sequence* of tones:

$\boldsymbol{D}\mathbf{\_2\{1\} = [1\ 1\ 1\ 1]'}$

This indicates that initially there is an equal probability of each tone sequence (left to right: 'all high', 'all low', 'high-low', 'low-high').

Here we must also include a second hidden state factor that encodes beliefs about the time point within a trial (e.g., 'first tone', 'second tone', etc.). This includes (from left to right) time points for 4 tones, a delay period, and then a reporting period:

$\boldsymbol{D}\mathbf{\_2\{2\} = [1\ 0\ 0\ 0\ 0\ 0]'}$

This indicates that the agent always starts in the 'time 1' state.

We also include a reporting state factor, corresponding to the agent either not yet reporting (left entry), reporting 'same tone' (middle entry), and reporting 'different tone' (right entry) at the end of the trial:

$\boldsymbol{D}\mathbf{\_2\{3\} = [1\ 0\ 0]'}$

This indicates that the agent always starts in a state of not yet having made a report.

Finally, we allow the agent to build up prior beliefs over time with repeated trials. In this case, because the agent's beliefs initially match the generative process, we can simply set:

$\boldsymbol{d}\mathbf{\_2 = }\boldsymbol{D}\mathbf{\_2}$

Next, we must specify the likelihood mappings for the second level in the **A** matrix**.** Because *time in trial* is a state factor, this becomes somewhat more complex. Specifically, we are now required to specify the type of tone *at each time point* that is expected *under each sequence*. To do so, we can specify the matrices as follows. For convenience, we can first specify:

$\boldsymbol{for\ i\ =\ 1{:}6}$

$\quad\boldsymbol{for\ j\ =\ 1{:}3}$

$$\mathbf{A\_2}\{\mathbf{1}\}(:,:,\boldsymbol{i},\boldsymbol{j}) = \begin{bmatrix} \mathbf{1} & \mathbf{0} & \mathbf{1} & \mathbf{0} \\ \mathbf{0} & \mathbf{1} & \mathbf{0} & \mathbf{1} \end{bmatrix}$$

$\boldsymbol{end}$

$\boldsymbol{end}$

This says that for the first six time points ($i = 1{:}6$), and for all three choice states ($j = 1{:}3$), the first and third sequence states (i.e., 'all high tones' and 'high tones followed by low tone'; columns 1 and 3) are associated with the 'high tone' observation (top row), whereas the second and fourth sequence states (i.e., 'all low tones' and 'low tones followed by high tone'; columns 2 and 4) are associated with the 'low tone' observation (bottom row). Then, we can adjust this so that the deviation sequences ('low tones followed by high tone' and 'high tones followed by low tone'; columns 3 and 4) are associated with the opposite tone mapping at the fourth time point ($i = 4$):

$\boldsymbol{for\ i = 4}$

$\boldsymbol{for\ j = 1{:}3}$

$$\mathbf{A\_2}\{\mathbf{1}\}(:,:,\boldsymbol{i},\boldsymbol{j}) = \begin{bmatrix} \mathbf{1} & \mathbf{0} & \mathbf{0} & \mathbf{1} \\ \mathbf{0} & \mathbf{1} & \mathbf{1} & \mathbf{0} \end{bmatrix}$$

$\boldsymbol{end}$

$\boldsymbol{end}$

The second outcome modality at the higher level (which does not correspond to a lower-level state) is feedback about whether a chosen report was correct or incorrect. Here, we need to specify that the agent will observe 'correct' feedback (at the final time point) in cases where its report matches the appropriate sequence (row 3), and 'incorrect' feedback otherwise (row 2). To do this, we can initially specify that no feedback ('null'; row 1) will be observed across all time points:

$\boldsymbol{for\ i = 1{:}6}$

$\boldsymbol{for\ j = 1{:}3}$

$$\mathbf{A\_2}\{\mathbf{2}\}(:,:,\boldsymbol{i},\boldsymbol{j}) = \begin{bmatrix} \mathbf{1} & \mathbf{1} & \mathbf{1} & \mathbf{1} \\ \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} \end{bmatrix}$$

$\boldsymbol{end}$

$\boldsymbol{end}$

The we can specify that at time point $i = 6$, if the agent reports 'same' ($j = 2$), then it will receive correct feedback when it is one of the first two (standard) sequences and incorrect for second two (deviant) sequences:

$\boldsymbol{for\ i\ =\ 6}$

$\quad\boldsymbol{for\ j\ =\ 2}$

$$\mathbf{A\_2}\{\mathbf{2}\}(:,:,\boldsymbol{i},\boldsymbol{j}) = \begin{bmatrix}\mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0}\\ \mathbf{0} & \mathbf{0} & \mathbf{1} & \mathbf{1}\\ \mathbf{1} & \mathbf{1} & \mathbf{0} & \mathbf{0}\end{bmatrix}$$

$\quad\boldsymbol{end}$

$\boldsymbol{end}$

The we specify the opposite mapping if the agent reports 'different' ($j\ =\ 3$):

$\boldsymbol{for\ i\ =\ 6}$

$\quad\boldsymbol{for\ j\ =\ 3}$

$$\mathbf{A\_2}\{\mathbf{2}\}(:,:,\boldsymbol{i},\boldsymbol{j}) = \begin{bmatrix}\mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0}\\ \mathbf{1} & \mathbf{1} & \mathbf{0} & \mathbf{0}\\ \mathbf{0} & \mathbf{0} & \mathbf{1} & \mathbf{1}\end{bmatrix}$$

$\quad\boldsymbol{end}$

$\boldsymbol{end}$

As with the first-level model, to control the precision of the mapping between first- and second-level states in the generative model, we can use a $precision\_2$ parameter. To do so, we can specify:

$\boldsymbol{a\_2 = A\_2}$

$\boldsymbol{precision\_2 = 2}$;

$\boldsymbol{a\_2\{1\}\ =\ spm\_softmax(precision\_2 * log(A\_2\{1\} + exp(-4)));}$

This results in a minor amount of noise in the messages passed between levels. As with the first level, we also multiply this parameter by 100 to prevent learning.

Next, we must specify the transition matrices for the second level. In this case, the sequence type is stable within a trial, so this should be an identity matrix (columns: states at time τ, rows: states at $\tau + 1$)

$$\mathbf{B\_2}\{\mathbf{1}\} = \begin{bmatrix} \mathbf{1} & \mathbf{0} & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathbf{1} & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \mathbf{1} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{1} \end{bmatrix}$$

Time in trial should progress forward (e.g., 'Time 1' should 'transition to 'Time 2', and so forth. As such:

$$\mathbf{B\_2}\{\mathbf{2}\} = \begin{bmatrix} \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} \\ \mathbf{1} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathbf{1} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \mathbf{1} & \mathbf{0} & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{1} & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{1} & \mathbf{1} \end{bmatrix}$$

Finally, report states are under control of the agent. In this case, there are three actions:

$$\mathbf{B\_2}\{\mathbf{3}\}(:,:,\mathbf{1}) = \begin{bmatrix} \mathbf{1} & \mathbf{1} & \mathbf{1} \\ \mathbf{0} & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \mathbf{0} \end{bmatrix}$$

$$\mathbf{B\_2}\{\mathbf{3}\}(:,:,\mathbf{2}) = \begin{bmatrix} \mathbf{0} & \mathbf{0} & \mathbf{0} \\ \mathbf{1} & \mathbf{1} & \mathbf{1} \\ \mathbf{0} & \mathbf{0} & \mathbf{0} \end{bmatrix}$$

$$\mathbf{B\_2}\{\mathbf{3}\}(:,:,\mathbf{3}) = \begin{bmatrix} \mathbf{0} & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \mathbf{0} \\ \mathbf{1} & \mathbf{1} & \mathbf{1} \end{bmatrix}$$

These three matrices (from 1-3 in dimension 3) correspond to the actions of moving (from any state) to the 'no report' state, 'report same' state, and 'report different' state, respectively. Next, we must specify the allowable sequences of actions (i.e., policies). In this case, we include two policies (two columns) and one row per time point. There are no actions for the first state factor, so (number = action, column = policy, row = time point):

$$\mathbf{V\_2}(:,:,\mathbf{1}) = \begin{bmatrix} \mathbf{1} & \mathbf{1} \\ \mathbf{1} & \mathbf{1} \\ \mathbf{1} & \mathbf{1} \\ \mathbf{1} & \mathbf{1} \\ \mathbf{1} & \mathbf{1} \end{bmatrix}$$

There are also no actions for the second state factor:

$$\mathbf{V\_2}(:,:,\mathbf{2}) = \begin{bmatrix} \mathbf{1} & \mathbf{1} \\ \mathbf{1} & \mathbf{1} \\ \mathbf{1} & \mathbf{1} \\ \mathbf{1} & \mathbf{1} \\ \mathbf{1} & \mathbf{1} \end{bmatrix}$$

For the third state factor, the agent must wait until the last time point and then either select the 'report same' or 'report different' actions:

$$\mathbf{V\_2}(:,:,\mathbf{3}) = \begin{bmatrix} \mathbf{1} & \mathbf{1} \\ \mathbf{1} & \mathbf{1} \\ \mathbf{1} & \mathbf{1} \\ \mathbf{1} & \mathbf{1} \\ \mathbf{2} & \mathbf{3} \end{bmatrix}$$

Lastly, we must provide the agent with preferences that will motivate accurate reporting. For the first outcome modality (tones), the agent has no preferences (columns = time point, rows [top to bottom] = high tone, low tone observation):

$$\mathbf{C\_2}(:,:,\mathbf{1}) = \begin{bmatrix} \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} \end{bmatrix}$$

For the second outcome modality (accuracy feedback), the agent prefers to receive 'correct' feedback at the last time point (column 6, row 3) and finds 'incorrect' feedback to be aversive at the last time point (column 6, row 2):

$$\mathbf{C\_2}(:,:,\mathbf{2}) = \begin{bmatrix} \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & -\mathbf{1} \\ \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{1} \end{bmatrix}$$

Having now specified the second-level model, we will place each of these matrices into its own mdp structure:

$$\mathbf{mdp.D} = \mathbf{D\_2}$$

$$\mathbf{mdp.d} = \mathbf{d\_2}$$

$$\mathbf{mdp.A} = \mathbf{A\_2}$$

$$\mathbf{mdp.a} = \mathbf{a\_2}$$

$$\mathbf{mdp.B2} = \mathbf{B\_2}$$

$$\mathbf{mdp.C} = \mathbf{C\_2}$$

$$\mathbf{mdp.V} = \mathbf{V\_2}$$

We then need to connect the lower-level model with the higher-level model as follows:

$$\mathbf{mdp.MDP} = \mathbf{MDP\_1}$$

Next, we need to provide a matrix specifying which outcome modalities at the second level (columns) corresponds to which state factors at the lower level (rows) within a 'link' field. Here, the first outcome at the second level ('tones') corresponds to the first state factor at the first level:

$$\mathbf{mdp.link} = [\mathbf{1\ 0}]$$

Lastly, we need to set the value of the ERP 'reset' or 'decay' parameter $\mathbf{mdp.erp}$, which at the start of every epoch of gradient descent is used to reset the posterior over states by dividing the posterior by the value of the parameter (i.e., higher values = more resetting). Setting the value of the parameter is entirely up to the discretion of the modeler, depending on assumptions about the particular neurocognitive process under study. In empirical work, this parameter would be fit to observed ERP responses. In the experimental paradigm we simulate here, the tones are played to the participant in quick succession, so we assume that the posterior at each time step carries over and does not decay between presentations. As such, we set $\mathbf{mdp.erp} = \mathbf{1}$. If, however, we were trying to model a task with longer time periods between updating (e.g., a subject navigating a maze), some degree of decay could be appropriate ($\mathbf{mdp.erp} = \mathbf{4}$ is the default value in the standard simulation script).

As before, we can now run this structure through the standard routine to generate simulated behavioral and neuronal responses of an example trial.

$$\mathbf{MDP} = \mathbf{spm\_MDP\_VB\_X\_tutorial}(\mathbf{mdp});$$

To simulate our two conditions of interest, *local standard + global deviation*, and *local deviation + global standard*, we will simulate 10 sequential trials for each condition. For brevity, we will not describe the code that implements this here. Instead, we direct readers to the **Step_by_Step_Hierarchical_Model.m** script, which has detailed comments describing each of the necessary steps. For both conditions, the first nine trials consist of three high tones and a fourth low tone. On the tenth trial of the *local deviation + global standard* condition, the trial again consists of three high tones and a fourth low tone. On the tenth trial of *local standard + global deviation* condition, the tenth trial instead consists of four high tones, thereby violating the global regularity. **Figure 13** shows plots of second-level belief updating and policy selection, analogous to the single-level model plots shown in **Figure 7**. The model performed at ceiling with 100% accuracy when classifying the last stimulus in the sequence as same or different.

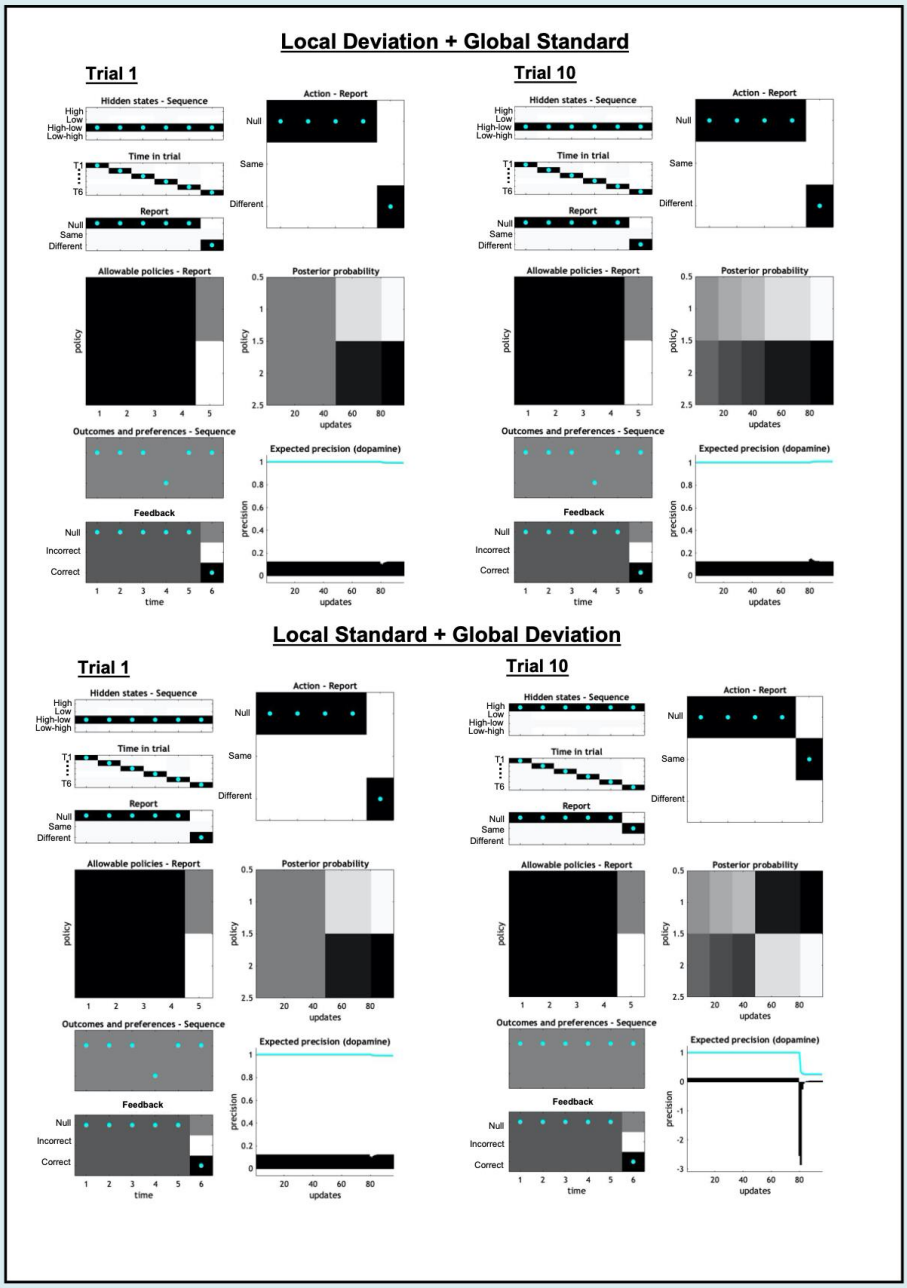

Figure 13. Simulation results from the hierarchical POMDP (analogous to the single-level model plots show in **Figure 7**). The three panels in the *top-left* of each plot show posteriors over states at the end of the trial. That is, the states the model believes it was in at each time point τ at the last time point t. Here, time goes from left to right, darker indicates higher probability, and the cyan dots denote the true states. The *top-right* panels in each plot show the action probabilities and true actions. The *left-middle* panel in each plot just shows the different possible action-sequences/policies possible in the specified model (encoded numerically from left to right). A darker color indicates a higher number.

The *right-middle* panel in each plot shows the progression of posterior beliefs in each policy over time (from left to right, darker = higher confidence). Policies (rows) line up with the action sequences in the plot on the *middle-left*. The two panels in the bottom left of each plot display the outcomes in cyan dots and the agent's preference for each outcome, where darker colors indicate a greater preference (i.e., higher prior probability). Lastly, the bottom-right plot displays predictions about dopamine responses based on the neuronal process theory (i.e., encoding changes in the precision of the distribution over policies with each new observation). In terms of behavior, notice that all models selected the correct actions and received "correct" feedback, as indicated in the lower outcome plot. These simulations can be reproduced using the **Step_by_Step_Hierarchical_Model.m** script included as supplementary code. Note that, due to random sampling, results may not be identical each time.

The next step is to visualize the resulting simulations in order to make empirical predictions about behavior and neuronal responses.

6.3 Plotting Hierarchical Models

To visualize the resulting belief updating and simulated neuronal responses for this hierarchical model, we have provided a modified version of the plotting script provided in the standard routines:

MDP = **spm_MDP_VB_ERP_tutorial**(**MDP**);

This plotting script shows the simulated firing rate associated with belief updating at each level. It also shows the simulated ERPs (summed first derivative of the firing rates) that would be expected to be generated during the simulated task. **Figure 14** shows the first and tenth trial from each condition. Notice that, on the first trial of both the *local standard + global deviation* condition and the *local deviation + global standard* condition, the high firing rates at the first level reflect a high posterior confidence in the tone each time it is presented. In contrast, after hearing the first tone the firing rate is evenly spread between both the "high" and "high-low" hidden states at the second level, reflecting the agent's uncertainty about which type of sequence is being presented (i.e., since both sequences predict "high" tones for the first three time steps). Note, however, that the fourth tone on the first trial generates an increased firing rate for one of the hidden states, and the firing rate drastically decreases for the other state (depending on whether a "high" or "low" tone was presented). By the tenth trial, the agent has a high prior expectation that it will experience a "high-low" sequence, reflected in the high firing rate for this second-level hidden state from early time points. In the *local deviation + global standard* condition, this expectation is confirmed, leading to little change in second-level firing rates. Importantly, however, in the *local standard + global deviation* condition, this expectation is violated, because the agent is presented with four high tones. This creates a rapid switch in posterior confidence from the "high-low" hidden state to the "high" hidden state at the fourth time step, generating a very strong and rapid shift in second-level beliefs.

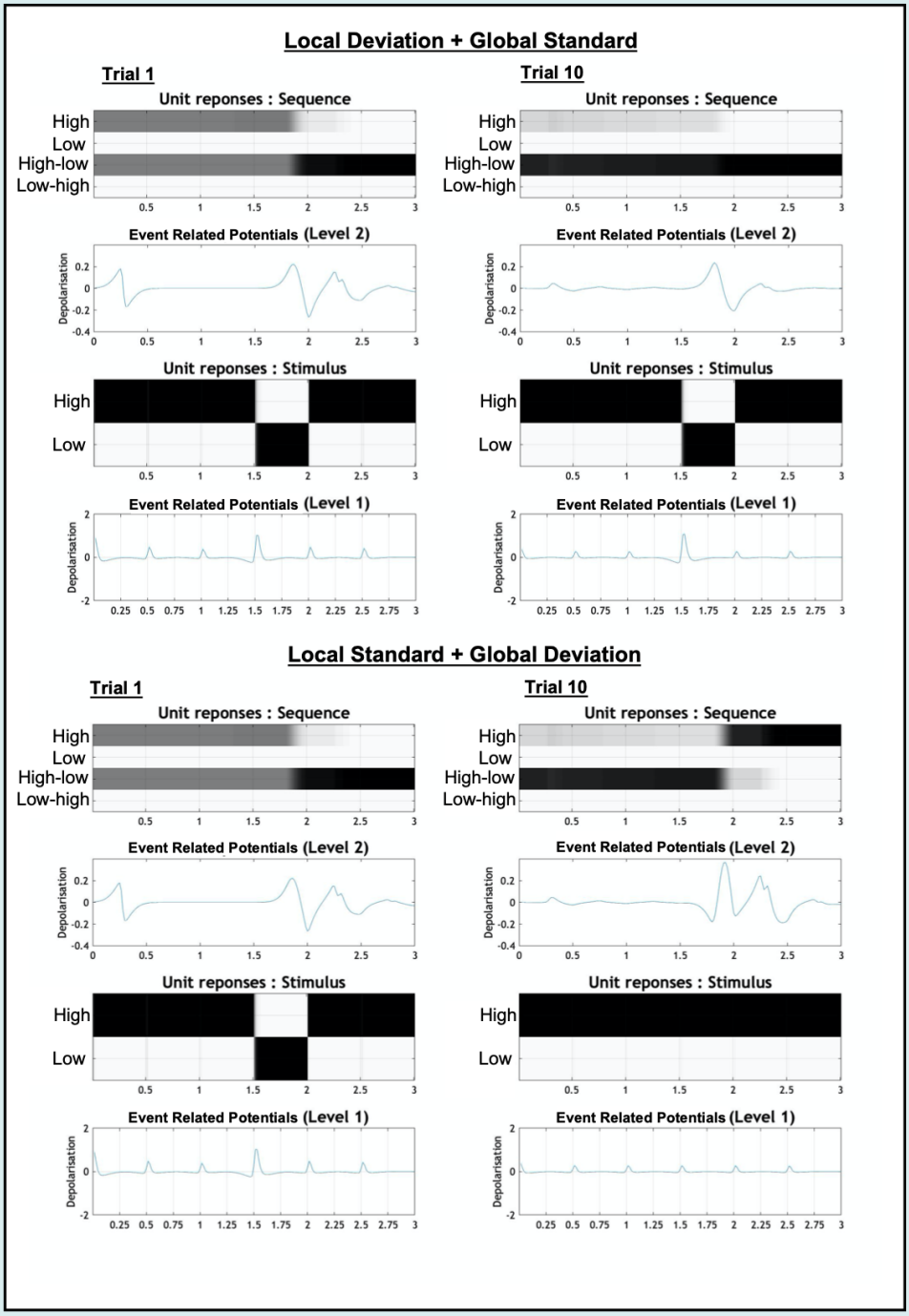

Figure 14. ERPs and firing rates extracted from the hierarchical POMDP model of the oddball paradigm described in the text. The unit response plots show the posterior probability over states ($s_{\pi,\tau}$) at each level of the model (as usual, darker colors = higher posterior = higher firing rate). As described in the neuronal process theory section, normalized firing rates are generated by passing the depolarization variable $v_{\pi,\tau} = \ln(s_{\pi,\tau})$ through a softmax function $s_{\pi,\tau} = \sigma(v_{\pi,\tau})$. The ERP plots

show the rate of change (first derivative) of posterior beliefs over states summed over all states at each level of the model (analogous to the aggregate signal measured at the level of the scalp by EEG). Note that each increment of 0.5 along the x-axis corresponds to a trial at the lower level, and to a time point at the higher level (i.e., 6 time points in a higher-level trial). For a detailed description of each plot and its meaning in relation to the task, see the main text. These simulations can be reproduced using the **Step_by_Step_Hierarchical_Model.m** script included as supplementary code. Note that, due to random sampling, results may not be identical each time.

This brings us to simulated ERPs, which reflect the rate at which posterior beliefs change within each epoch of belief updating, summed over hidden states at each level of the model (i.e., similar to the aggregate signal measured by EEG). At the first level, the repeated presentation of high tones generates small ERPs, whereas deviations from this pattern at the fourth time step (local deviation) generate a larger amplitude mismatch response because posterior beliefs about the tone change rapidly. Importantly, when the local deviation response is subtracted from the local standard response, we can reproduce the classic mismatch negativity effect (MMN; see **Figure 15**). At the second level, the repeated occurrence of three high tones and one low tone for the first nine trials creates a strong prior expectation (through the increase in concentration parameters in the *D* vector) for the "high-low" sequence state. When the model is unexpectedly presented with four high tones on the tenth trial, this expectation violation generates a rapid change in beliefs and a correspondingly large second-level ERP resembling the P300 (see **Figures 14 and 15**). Finally, **Figure 15** shows custom-made ERP plots, which isolate the contrasts of interest. Again, for the sake of brevity, we will not describe the code that generates this plot in the main text, but direct interested readers to the **Step_by_Step_Hierarchical_Model.m** script for more details.

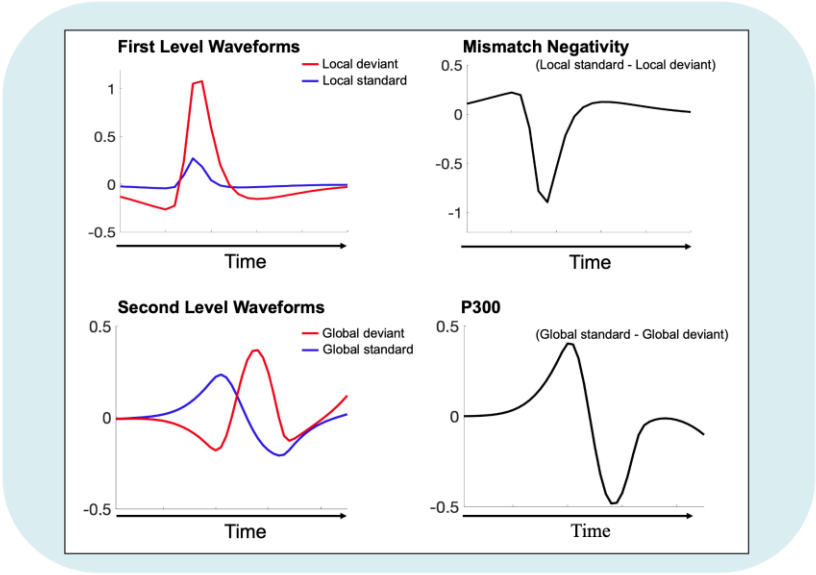

Figure 15. Custom ERP plots generated from the simulated oddball paradigm. On the left we have the "raw" first- and second-level ERP waveforms centered on the fourth time step of the tenth trial

from each condition. The right side of the figure shows how the subtraction of the deviant trials from the standard trials reproduces both the MMN and P300. At both the first and second level, "deviant" ERPs have a substantially larger amplitude than "standard" ERPs. Note that, although the relative differences in timing between simulated ERPs (and the relative differences in timing between ERPs at different levels of the model) are meaningful, the units we ascribe to time are (usually) somewhat arbitrary. So, for clarity, we have not included any units of time on the x-axis. These simulations can be reproduced using the **Step_by_Step_Hierarchical_Model.m** script included as supplementary code. Note that, due to random sampling, results may not be identical each time.

7. Fitting Models to Behavior

So far, we have focused on reproducing sentient behavior from first (variational) principles and establishing the face validity of this approach using canonical examples from the decision-making and electrophysiological literature. In this section, we turn to the use of these models to explain the behavior of experimental subjects. We will see that this basically affords the opportunity to quantify the prior beliefs that each subject brings to the table, when completing some empirical task. For previous empirical studies that have applied active inference models to human behavior during task performance, see (Mirza, Adams, Mathys, & Friston, 2018; Schwartenbeck, FitzGerald, Mathys, Dolan, & Friston, 2015; Smith et al., 2021; Smith, Kuplicki, et al., 2020; Smith, Schwartenbeck, Stewart, et al., 2020).

Empirical applications of active inference require fitting a task model to participant behavior. When fitting a model to behavior one would like to find the parameters that maximize the posterior probability of a model given that behavior, $p(model|participant\ behavior)$. Assuming a flat prior belief over models, this posterior is proportional to the likelihood term, $p(participant\ behavior|model)$. Thus, many fitting approaches (estimation algorithms) try to find the parameters that maximize this likelihood – referred to as *maximum likelihood estimation* (MLE). However, in some cases one might also have reason to expect that certain models are more likely than others a priori. In this case, one can also incorporate an informative prior belief over models, $p(model)$, and maximize the posterior as opposed to the likelihood. This is referred to as *maximum a posteriori estimation* (MAP). Regardless of the specific approach, the goal of any fitting procedure is to find the set of parameters that would best reproduce the actual behavior of a participant (with the highest probability) when running simulations using a model (while in some cases also incorporating any prior knowledge one might have).

To do this one needs to feed the trial-by-trial observations made by participants (e.g., cues, wins/losses, etc.) into the model and look at the actions predicted by the model (i.e., posterior probabilities over actions). One can then compare these predictions to the actions a participant chose. Under some sets of parameter values (e.g., prior expectations, precisions, etc.), the model's predictions may not match behavior well (i.e., the probability of a participant's actions under the model may be low). However, by searching through different possible combinations of parameter values, the best combination can be found for a given participant. Note, however, that this will only

be the best combination possible *for that model*. This does not mean that the model has high explanatory power (e.g., the best parameter combination for one model might lead to an average action probability of 0.4, while another model might reach 0.7, etc.). This is why it is important to compare the explanatory power of a few different models.

Throughout this section, we encourage the reader to follow along in the companion MATLAB script (**Step_by_Step_AI_Guide.m**). This can be found in **supplementary materials**, as well as at: https://github.com/rssmith33/Active-Inference-Tutorial-Scripts. At the top of this script, if you set $\boldsymbol{Sim = 4}$ it will perform parameter estimation on a single set of simulated behavioral data from the explore-exploit task model (see the top panel of **Figure 16** for example outputs). If you set $\boldsymbol{Sim = 5}$ it will simulate behavioral data from two models (one with two parameters and one with three parameters; see below) for a few synthetic participants and then perform model comparison. It will also assess how well the estimated parameter values match with the true parameter values used to generate the simulated task behavior (i.e., it will output correlation matrices and associated p-values). This is important to check (as described below in relation to parameter recoverability) to ensure that parameters can be estimated reliably (this is sometimes referred to as **model identifiability**). All the code for performing these steps is included and described in the script comments. All examples here will be based on the explore-exploit task model, when simulating behavior during the case of reversal learning described above. We will estimate the alpha parameter (α) encoding action precision (i.e., inverse temperature) as well as a risk-seeking parameter (described below). For model comparison, we will also estimate learning rate (η) in a second model.

As mentioned above, finding the optimal combination of parameter values to account for a participant's behavior requires a model fitting (parameter estimation) procedure. There are many different procedures (estimation algorithms) that are available, each with their strengths and limitations. The simplest approach is called a grid search, where each possible combination (within some specified range of values) is tried one-by-one, and then the one that best reproduces behavior is chosen. However, this approach is limited to fairly simple models with a small number of parameters, and it can also lead to overfitting (i.e., finding parameters that reproduce random aspects of behavior). Thus, a number of other algorithms have been developed. In this section, we will focus on one estimation technique in particular called variational Bayes (Friston, Mattout, Trujillo-Barreto, Ashburner, & Penny, 2007), which is based on exactly the same free energy minimization ideas described above. We will go over a set of concrete steps that can be taken to use it for model fitting. As we are focusing on the explore-exploit task model, it may help to glance back at the model/task description in earlier sections before moving on.

The first step is to place trial-by trial-observations and participant actions into an mdp structure. In the explore-exploit task model, for each trial this would mean specifying the observations ($mdp.o$) a participant made for the 3 outcome modalities at the 3 time points in each trial, and then specifying the 2 actions taken ($mdp.u$). For example, if on trial #1 the participant chose to take the hint and then chose the left machine:

$$mdp(1).u = \begin{bmatrix} 1 & 1 \\ 2 & 3 \end{bmatrix}$$

$$mdp(1).o = \begin{bmatrix} 1 & 2 & 1 \\ 1 & 1 & 3 \\ 1 & 2 & 3 \end{bmatrix}$$

Here, time goes from left to right. For $mdp.u$, actions for each state factor correspond to each row from top to bottom. For $mdp.o$, time also goes from left to right and rows correspond to outcome modalities. The parenthetical "(1)" for the mdp denotes trial #1. Here, for $mdp.u$, remember there is only one "action" for the first state factor (i.e., the participant does not have control of which machine will lead to a win). So, the top row is simply a 1 for both time points. For the second state factor (choice), the participant first chose the hint (action #2) and then chose the left machine (action #3). For $mdp.o$, the first row indicates "null" (observation 1), "left hint" (observation #2), and then back to "null". The second row indicates two "null" observations followed by a "win" (observation #3). The last row simply corresponds to the choices ("start", "take hint", "choose left"). Observations and behavior for each trial need to be inserted in this same way (e.g., $mdp(2)$, $mdp(3)$, etc. until the final trial number).

The second step is to choose the model parameters you want to estimate and which you want to hold fixed. (Ultimately, one may want to try estimating different models and/or different numbers of parameters and then compare them to find which best accounts for behavior, as we describe further below). In this example, we will first consider estimating two parameters: learning rate ($\boldsymbol{\eta}$) and "risk-seeking" ($\boldsymbol{RS}$). The latter corresponds to how strong the preference is to win money in the **C** matrix:

$$\mathbf{C}\{\mathbf{2}\} = \begin{bmatrix} \mathbf{0} & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathbf{-1} & \mathbf{-1} \\ \mathbf{0} & \boldsymbol{RS} & \dfrac{\boldsymbol{RS}}{\mathbf{2}} \end{bmatrix}$$

As reviewed above, learning rate scales the size of the 'count' that is added to the (Dirichlet) concentration parameters after a new observation. The $\boldsymbol{RS}$ parameter controls the explore-exploit trade-off. In the simulations shown in **Figure 10** and **Figure 11**, we saw that, as $\boldsymbol{RS}$ values go up, the probability that a participant will take the hint goes down. That is, they will tend to 'risk it' and simply guess left or right in hopes of winning the larger amount of money.

When using variational Bayes, the next step is to choose prior expectations for each parameter – which include both a prior mean and a prior variance (these can be set within the supplementary "**Estimate_parameters.m**" script, as described further below). A simple way of thinking about the variational Bayes algorithm is that it starts from these prior values and slowly moves away from them to find the best combination, while also balancing the associated cost of increased model complexity that is incurred when the parameters move too far away from prior values. More specifically, variational Bayes is accomplished by performing a gradient descent on VFE (i.e., the same process active inference models use to accomplish perception, learning, and action selection). In this case, the gradient descent starts with the chosen prior values, and then evaluates the log-probability of a participant's actions (e.g., sequence of choices). This likelihood is evaluated using the posterior beliefs about action that the subject would have had, given the priors in question and the outcomes

they observed. The scheme then evaluates neighboring parameter values and continues in the direction of increasing likelihood, until a combination is found with no neighboring values that improve the fit.

However, because we seek a VFE minimum, which includes both complexity and accuracy terms, the algorithm will not simply maximize model accuracy. It will also try to minimize how much estimates move from the (*objective*) prior values when trying to identify a participant's (i.e., *subjective*) prior beliefs (i.e., parameter estimates). In other words, parameter estimates for a given subject will represent values that accurately reproduce a that subject's behavior while moving as little as possible from the prior values chosen by the experimenter. Importantly, complexity minimization depends on the prior variance that you choose. Setting a small prior variance will lend strong weight to complexity minimization; in contrast, setting a large variance will lead to a stronger weight being placed on maximizing accuracy. In practice, one may want to test out different prior values and assess the consistency in the parameter estimates they produce. If estimates for all participants tend to move far in one direction, on also might consider choosing an objective prior that appears closer to the range of values where posterior estimates tend to be found. For example, if you initially set a prior value of $\boldsymbol{RS} = \mathbf{1}$ and notice that estimates for all participants tend to move up to between 3 and 5, one could consider instead choosing a prior of $\boldsymbol{RS} = \mathbf{4}$ (as it appears to be a better prior for the group as a whole; this may also help to better identify individual differences by reducing complexity costs, because parameters would not need to move as far from prior values). Note that, although we do not go into detail here, there are also 'empirical Bayes' methods that could be used to estimate group-level priors and then re-estimate individual-level parameters based on those group-level priors (Carlin & Louis, 1998; Friston, Litvak, et al., 2016).

Note a subtlety of the above scheme, in the sense that we are using variational Bayes to estimate the parameters that underwrite variational inference. In other words, there are two generative models in play: first, a **subjective model**, which is the POMDP we assume the subject is using to plan their responses. Second, there is an **objective model** that the experimenter specifies in terms of the parameters of the subjective model. A key conceptual point here is that the parameters of the subjective model can always be interpreted as priors; either explicitly – or implicitly in terms of the structure or form of the subjective model. This is important because of something called the complete class theorem (L. D. Brown, 1981; Wald, 1947). The complete class theorem says that for any pair of reward functions (i.e., preferences) and choice behavior, there exists some prior beliefs that render the choices Bayes optimal. This means that any behavior can be described, under ideal Bayesian assumptions, in terms of some prior beliefs. It is these that provide a theoretically complete characterization of any given subject, in any given experimental paradigm.

After choosing a final set of objective priors, and obtaining parameter estimates for each participant, it is also important to confirm what is called *parameter recoverability* (also sometimes referred to as testing whether the model is *identifiable* or *invertible*). What this means is that you need to be sure that, if simulated behavior were generated from a model with a given set of parameter values, that the estimation algorithm would provide reasonably precise estimates that approach the parameter values that generated the behavior. For example, assume a given participant's estimates were $\boldsymbol{RS} = \mathbf{3.2}$ and $\boldsymbol{\alpha} = \mathbf{4}$ (e.g., with estimation priors of $\boldsymbol{RS} = \mathbf{4}$ and $\boldsymbol{\alpha} = \mathbf{2}$). Assessing recoverability would

require putting these estimates into the task model, generating simulated behavior, and feeding that behavior into the fitting algorithm. If the algorithm returns estimates that moved in the right direction from the prior values and approach the values that generated the data (e.g., $\boldsymbol{RS} = \mathbf{3.3}$ and $\boldsymbol{\alpha} = \mathbf{3.5}$), then we can be more confident that the participant's estimates are capturing something meaningful and reliable about their decision process. If the algorithm returned estimates that were farther away and moved in the wrong direction from the prior values (e.g., $\boldsymbol{RS} = \mathbf{4.9}$ and $\boldsymbol{\alpha} = \mathbf{1.3}$), this suggests that the parameter estimates for that participant may not be reliable (e.g., a different combination of parameter values might reproduce their behavior equally well). Thus, before interpreting parameter estimates – and using them in subsequent group analyses – it is important to confirm that the parameters used to generate the data and estimated parameters are correlated within the range of parameter values found within participants.

It is important to keep in mind, however, that the products of Bayesian model inversion (here, parameter estimation) will generally speaking never be the same as the parameters used to generate data. This is because there is usually a simpler way of generating any given data that the model inversion will identify. In other words, in a certain sense there are no 'true' parameters – only the best explanation in the sense of Occam's principle. Occasionally, people repeatedly generate synthetic data using estimated parameters until the estimates converge. One can then be assured that the data are being generated in the simplest way possible – and can meaningfully compare the parameter estimates with the parameters used to generate the data.

If the $\boldsymbol{Sim}$ variable is set to $\boldsymbol{Sim} = \mathbf{5}$ when you run the **Step_by_Step_AI_Guide.m** code, simulated behavior will be generated under several parameter values for the explore-exploit task model and it will run correlations between the estimated parameter values and the parameter values that actually generated the simulated behavior. In a representative example simulation, recoverability appeared high for $\boldsymbol{RS}$ (r = .95) and $\boldsymbol{\alpha}$ (r = .81). In a model that also included learning rate ($\boldsymbol{\eta}$), this parameter appeared recoverable as well (r = .75).

Before assessing parameter recoverability (i.e., model identifiability), however, it is important to first understand more about the steps for fitting. If you set $\boldsymbol{Sim} = \mathbf{4}$ in the **Step_by_Step_AI_Guide.m** script, it will generate simulated explore-exploit task behavior for a single participant and feed it into a 'DCM' (for Dynamic Causal Modeling) data structure in the appropriate format for model fitting. It then calls the supplementary **Estimate_parameters.m** script we have provided, which takes the DCM structure as input, runs the variational Bayes routine in SPM12 (**spm_nlsi_Newton.m**), and calculates the log-likelihood – that is, the sum of the log-probabilities of chosen actions under the model. In our script, the DCM structure includes the following:

DCM.MDP (**i.e., the generative model**)

DCM.U (**participant observations**)

DCM.Y (**participant actions**)

DCM.field (**which specifies the parameters that will be fit**)

Within the **Estimate_parameters.m** file, you will see locations (starting on line 47) where the script searches for the to-be-fit parameter names. Here, you can enter the prior means and variances for each parameter. Note that some parameters (e.g., learning rate) need to be between 0 – 1. For this reason, they are here transformed into logit-space so that the estimation routine does not assess values outside of that range. Similarly, parameters that can only take on positive values can be transformed into log-space to preclude negative values. For illustration, we have assumed that reward-seeking can only take positive values and is therefore log-transformed. Farther down in the script (starting on line 133), these values are re-transformed out of logit- or log-space when they are fed into the model and output as final estimates. Starting on line 188, the log-likelihood function loops through each trial, takes the probability of a participant's action in the model (stored in **MDP**. **P;** see **Table 3**), log-transforms it, and then sums the log-probabilities. The closer this value is to 0, the better the model fits the behavior.

When you run the **Step_by_Step_AI_Guide**. **m** script (which calls the others automatically) with $\boldsymbol{Sim = 4}$, a display will appear (as in **Figure 16**) that shows how the free energy changes over iterations (see **Figure 16**, *top panel*). Note that these values increase because they are negative and will approach 0 as model fit improves. If these values steadily increase, this suggests that fitting is converging onto a reliable estimate. If they instead fluctuate up and down inconsistently, this could suggest one or more problems that need to be addressed before model estimates are considered reliable (e.g., poor choice of priors, problems with parameter values remaining in valid ranges, estimates getting stuck in locally [but not globally] optimal values). More generally, a failure to converge is usually read as a useful diagnostic that the priors are somehow mis-specified. In other words, if model inversion does not converge gracefully within a few tens of iterations, you may want to think about whether your priors are appropriate – or whether you are trying to fit too many parameters to rather sparse data.

After convergence to the best estimates, the output **DCM** structure will contain additional fields. The following are relevant to our current use:

DCM. **Ep**: **posterior mean estimates** (**i**. **e**. , **expectation**) **for each parameter**.

DCM. **Cp**: **posterior covariance matrix**. **This includes posterior parameter variances**

on the diagonal, **and parameter covariances during estimation**.

DCM. **F**: **the final free energy value of the best fit model**.

The free energy values for each participant will later be used for model comparison. The covariance values (i.e., the off-diagonals in **DCM**. **Cp**) should be checked to make sure they are not too high (e.g., > .8), or it would suggest they were not independently estimated and that each estimate may not carry unique/reliable information. This is the same kind of check you would apply to an experimental design – to ensure explanatory variables are orthogonal and the model parameters can be estimated efficiently.

When testing hypotheses about which of several models best explains behavior, one must fit each model and compare how well they fit behavior. For Bayesian model comparison using variational Bayes, this means comparing the free energies at the group level. One common approach is to use a random effects model, which can be done using the **spm_BMS.m** function (available within SPM12). This function takes as input a matrix containing the free energies for each participant for each model (one column per model). For the details of this implementation of Bayesian model selection, see (Rigoux, Stephan, Friston, & Daunizeau, 2014; Stephan, Penny, Daunizeau, Moran, & Friston, 2009). An important output of this function is the **protected exceedance probability** (***pxp***) of each model. In this case, the model with the highest ***pxp*** has the most evidence. Often there will be a clear winner, with $\boldsymbol{pxp} = \mathbf{1}$ for a single model and 0 for the others. In cases, where there is no clear winner (e.g., [.48 .52]), it will be important to note as this may reflect insufficient evidence for a "best" model. If parameters in both models are recoverable (i.e., both models are identifiable), one may wish to consider parameters for both models in further between-subject analyses (e.g., parameters in one model may have higher explanatory value for specific theoretical questions). Setting $\boldsymbol{Sim} = \mathbf{5}$ in the **Step_by_Step_AI_Guide.m** script will generate/estimate behavior for a few simulated participants for models with and without the learning rate parameter and then do Bayesian model comparison (this will take several minutes). This will output the ***pxp*** for each model, as well as the model probabilities and a few other diagnostic outputs we will not cover in detail here; for further information, see (Rigoux et al., 2014; Stephan et al., 2009) as well as the documentation within the **spm_BMS.m** function. As one example, you could input the free energies for the 2-parameter (**F_2_params**) and 3-parameter (**F_3_params**) models as follows:

[alpha,exp_r,xp,pxp,bor] = spm_BMS([F_2_params F_3_params])

If the output were $\boldsymbol{pxp} = [\mathbf{1}\ \mathbf{0}]$, this would mean that the probability that the 2-parameter model is a better fit than the 3-parameter model is equal to 1. In a representative example simulation, the $\boldsymbol{pxp} = [.\mathbf{37}\ .\mathbf{63}]$, weakly favoring the 3-parameter model.

Once parameters for a winning, identifiable model have been estimated, one simple approach for group-level analysis would be to analyze the posterior parameter means across participants using standard frequentist analyses. However, one advantage of using variational Bayes is that it also provides information about posterior parameter variances for each individual (as opposed to only the posterior means). This allows for between-subject Bayesian analyses that take the variances into account. One approach that takes the output **DCM** structures for each participant as input is parametric empirical Bayes (**PEB**), described in detail in (Friston, Litvak, et al., 2016; Zeidman et al., 2019). PEB uses a general linear model and effectively down-weights the contribution of individual subject parameter estimates when those estimates have larger posterior variances (i.e., those with greater uncertainty). PEB can be run using the **spm_dcm_peb.m** and **spm_dcm_peb_bmc.m** functions, as well as the **spm_dcm_peb_review.m** function to inspect results. The first and second functions estimate and compare group models, respectively. In short, they allow a test of whether there is evidence for a model that does include group differences or whether there is more evidence for a model that does not include those differences. When including covariates (e.g., age), it also allows comparing the evidence for or against a relationship between parameters and those covariates.

Example code to implement such empirical Bayesian (random effects) analyses is also included in the accompanying MATLAB code **Step_by_Step_AI_Guide.m**. The code shows examples about (with commented descriptions of) how to set up inputs to PEB. In our example, the code sets several parameters to default values (for more information on these, see (Friston, Litvak, et al., 2016; Zeidman et al., 2019)) and then inputs a matrix ($\boldsymbol{M}.\boldsymbol{X}$) for a general linear model, with a column for the mean, a column separating participants into two groups (here, with low vs. high learning rates), and a column with randomly generated participant age values). This will allow us to assess the evidence for models including effects of group and/or age (and the strength of these effects).

To run PEB, set $\boldsymbol{Sim} = \mathbf{5}$ and set $\boldsymbol{PEB} = \mathbf{1}$. This may take some time, as it will first generate and estimate parameters for several (six) simulated participants before feeding them into the PEB scripts (note that the script will save the outputs of $\boldsymbol{Sim} = \mathbf{5}$ so that this does not to be repeated each time you want to practice using PEB). In beginning of *section 10* of the script, you can also specify whether you want to use PEB on the 2-parameter or 3-parameter model. When complete, some output plots and the PEB results viewer window will appear. Example outputs are shown in **Figure 16**, *bottom panel*). The figure legend describes how to interpret these outputs.

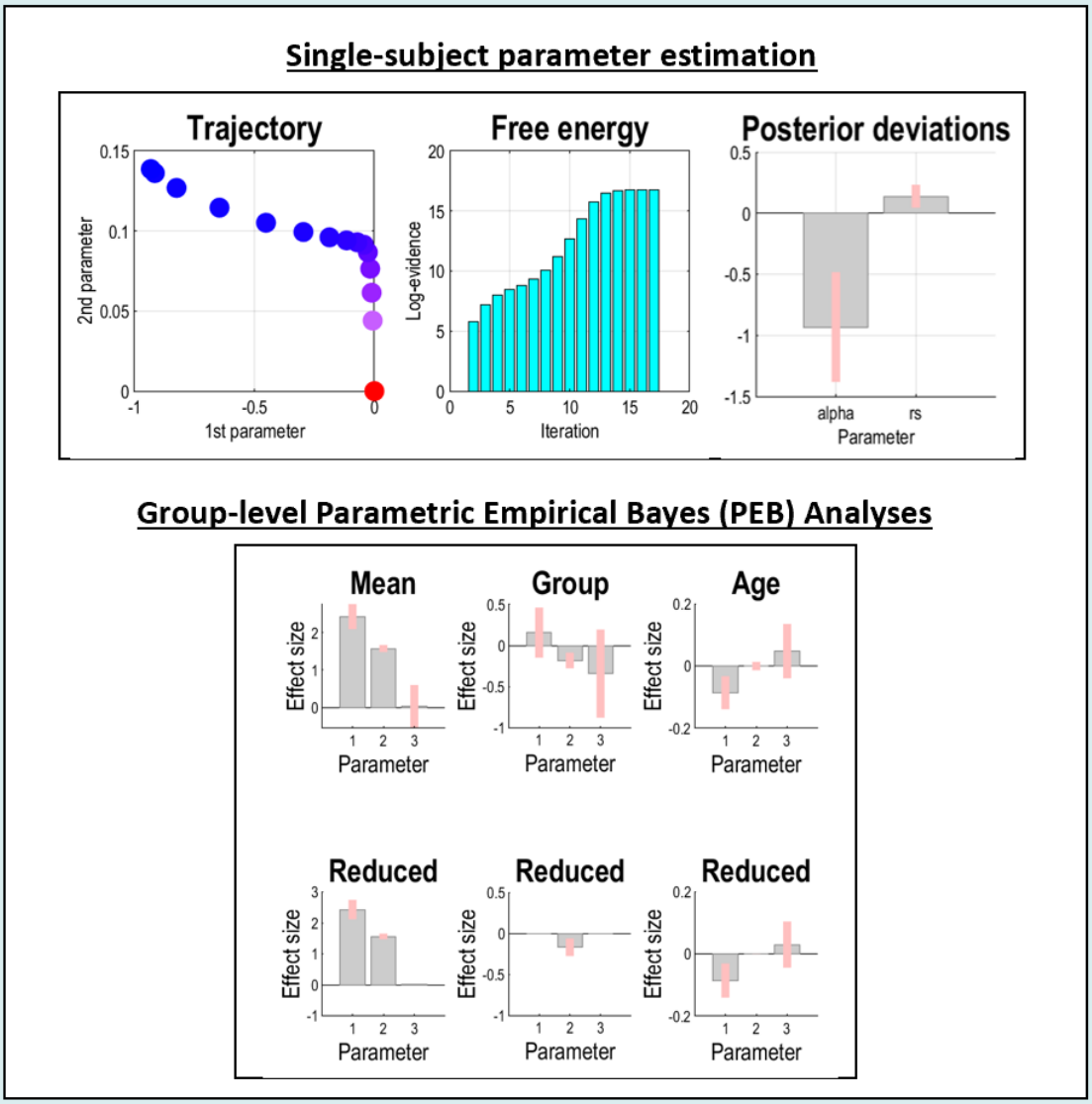

Figure 16. (*Top*) Output of single-subject parameter estimate scripts provided with this tutorial. This example included estimation of two parameters: alpha (action precision) and RS (risk-seeking). The left plot shows the trajectory of parameter values as fitting progressed (from red to blue). The middle plot shows how the log-probabilities of the model (given participant data) increased during estimation (via gradient descent on free energy). The right plot depicts how parameters moved from their prior to posterior values (and their posterior variances). Note that, for clarity, prior values for all parameters in these plots are given a common reference value of 0, such that posterior values are shown as deviations from those priors. For example, the prior values for alpha and RS were 16 and 5, respectively. The plots therefore show that posteriors for alpha deviated to a value below 16 and that those for RS deviated to a value above 5. These results can be reproduced by running the $\boldsymbol{Sim = 4}$ option in the supplementary **Step_by_Step_AI_Guide.m** code. (*Bottom*) Output of group-level parametric empirical Bayes' (PEB) analyses (scripts provided in supplementary code). This example

included estimation of three parameters for six simulated participants, where parameters 1, 2, and 3 correspond to alpha (action precision), RS (risk-seeking), and eta (learning rate), respectively. The top row (left to right) shows the evidence (for each parameter) for differences from 0, differences between two groups (simulated as having different risk-seeking values, i.e., parameter 2), and relationships to (arbitrarily specified) age values in models assuming effects for all parameters. The bottom row shows analogous results for reduced models that have greater evidence than the full models (here, no group difference in alpha [parameter 1] or learning rate [parameter 3], and some effects of age on alpha and learning rate). Recall that these results are based on a very small sample of only 6 simulated participants, which will generally be unreliable. They are for example only, and the identified relationships with the arbitrary 'age' values should not be taken seriously. Note that we have omitted some additional plots that are also generated. This is because these are the automatic output of scripts originally designed for dynamic causal modelling in neuroimaging, and not all of them are useful in the present context. These results can be reproduced by running the $\boldsymbol{Sim = 5}$ option in the supplementary **Step_by_Step_AI_Guide.m** code, while also setting the option $\boldsymbol{PEB = 1}$ (note that, because outcomes are sampled from probability distributions, results will not be identical each time).

8. Concluding Remarks

This concludes the tutorial. However, we have provided 'pencil and paper' exercises in **supplementary materials** (as well as solutions to check your work; see **Pencil_and_paper_exercise_solutions.m** code). In our experience, doing a few practice problems of this sort, and working with the code, are the best way to gain useful intuitions about the dynamics of these models and how they can be tailored for specific studies. We note that, while we have strived to be comprehensive, there are many new directions in active inference (and associated functionality in the standard SPM routines) incorporating, for example, multi-agent interactions, deep parametric models (e.g., 2nd-order parameters on habits, preferences, precisions, etc.), mixed models that link POMDPs to continuous motor control processes, among others (e.g., see (Friston, Parr, et al., 2017; Hesp et al., 2020)). We hope that working through the materials provided here will offer a "launching point" that will provide the reader with a sufficient foundation to independently extend their work with advances in the field as they emerge.

Acknowledgments: The authors would like to thank Thomas Parr, Giuseppe Pagnoni, and Conor Heins for offering useful comments and suggestions during preparation of the manuscript. The authors would also like to acknowledge Samuel Taylor for additional suggestions that improved the clarity of the manuscript.

Conflict of Interest: The authors have no conflicts of interest to disclose.

Funding: R.S. is supported by the William K. Warren Foundation and the Stewart G. Wolf Fellowship. C.W. is supported by the University of Cambridge Harding Distinguished Postgraduate Scholars Programme.

Supplementary Materials

1. Pencil and Paper Exercises

The purpose of this supplementary section is to provide a set of exercises that can be worked through using pencil and paper, with the aim of building an intuition for how active inference models operate. The first exercise is a simple example of static perception. The second example extends this by modelling how states (and the observations they generate) change across time, which is an example of a hidden Markov model (HMM). HMMs are the perceptual component of partially observable Markov decision processes (POMDPs). We have not included an example of policy selection, as performing the calculations for even two policies involves too many computations to reasonably expect readers to perform them by hand.

To help the reader build a conceptual bridge between the update equations and their implementation in code, we have also included MATLAB code (**Pencil_and_paper_exercise_solutions.m**) that has solutions to each of the exercises shown below. Finally, readers should note that, to ensure that the exercises can be solved by hand, we have excluded a key aspect of active inference models as they are usually implemented. Namely, instead of inferring the posterior over hidden states using gradient descent, we use only a single round of marginal message passing. For readers seeking to understand how message passing and policy selection operate in the model inversion procedure implemented in **spm_MDP_VB_X.m**, please see the stand-alone MATLAB script **Simplified_simulation_script.m** that is also provided, which is a stripped down, but thoroughly commented, version of the model inversion scheme used in **spm_MDP_VB_X.m**.

Exercises

Static Perception

For this first example we will keep things as simple as possible.

Example 1

Update Equation

$$\mathbf{s} = \sigma(\ln D + \ln \mathbf{A}^T \mathrm{o})$$

Generative Model and Observation

$$D = \begin{bmatrix} .5 \\ .5 \end{bmatrix};\ \mathbf{A} = \begin{bmatrix} .9 & .3 \\ .1 & .7 \end{bmatrix};\ \mathrm{o} = \begin{bmatrix} 1 \\ 0 \end{bmatrix}$$

Model Inversion

$$\mathbf{s} = \sigma\left(\ln \begin{bmatrix} .5 \\ .5 \end{bmatrix} + \ln \begin{bmatrix} .9 & .1 \\ .3 & .7 \end{bmatrix} \begin{bmatrix} 1 \\ 0 \end{bmatrix}\right) = \sigma\left(\ln \begin{bmatrix} .5 \\ .5 \end{bmatrix} + \ln \begin{bmatrix} .9 \\ .3 \end{bmatrix}\right)$$

$$= \sigma\left(\ln\begin{bmatrix}.5 \times .9\\ .5 \times .3\end{bmatrix}\right) = \sigma\left(\ln\begin{bmatrix}.45\\ .15\end{bmatrix}\right)$$

$$= \begin{bmatrix}\frac{e^{\ln(.45)}}{e^{\ln(.45)} + e^{\ln(.15)}}\\ \frac{e^{\ln(.15)}}{e^{\ln(.45)} + e^{\ln(.15)}}\end{bmatrix} = \begin{bmatrix}\frac{.45}{.45 + .15}\\ \frac{.15}{.45 + .15}\end{bmatrix} = \begin{bmatrix}.75\\ .25\end{bmatrix}$$

Exercise 1

Based on the update equation in example 1 and the observation listed below, invert the following generative model:

$$D = \begin{bmatrix}.75\\ .25\end{bmatrix};\ \mathbf{A} = \begin{bmatrix}.8 & .2\\ .2 & .8\end{bmatrix};\ \mathrm{o} = \begin{bmatrix}1\\ 0\end{bmatrix}$$

Dynamic Perception

In this second example we move from a static environment to a dynamic one in which hidden states, and the observations generated by these states, change across time. In addition to including a **B** matrix that encodes the transition probabilities, we must initialize the approximate posteriors for each tau (τ) before starting model inversion (recall, tau references a time *about* which one has beliefs, not a time *at* which one updates beliefs with a new observation). It is also important to note that, because the log of zero is not defined, the model inversion procedure implemented in the code adds a very small number ($e^{-16} = 0.00000011253$) to all inputs which turns the log of zero into the log of a very small number. Since we expect readers to do this exercise by hand, we approximate this by adding 0.01 to the input of all logs.

Example 2

Update Equations

$$\boldsymbol{s_{\pi,\tau=1}} = \sigma\left(\frac{1}{2}\left(\ln \mathbf{D} + \ln \mathbf{B}^{\dagger}_{\boldsymbol{\pi,\tau}} s_{\pi,\tau+1}\right) + \ln \mathbf{A}^{\boldsymbol{T}}\, o_{\tau}\right)$$

$$\boldsymbol{s_{\pi,1<\tau<T}} = \sigma\left(\frac{1}{2}\left(\ln \mathbf{B}_{\pi,\tau-1}\, \mathrm{s}_{\pi,\tau-1} + \ln \mathbf{B}^{\dagger}_{\boldsymbol{\pi,\tau}} s_{\pi,\tau+1}\right) + \ln \mathbf{A}^{\boldsymbol{T}}\, o_{\tau}\right)$$

$$\boldsymbol{s_{\pi,\tau=T}} = \sigma\left(\frac{1}{2}\left(\ln \mathbf{B}_{\pi,\tau-1}\, \mathrm{s}_{\pi,\tau-1}\right) + \ln \mathbf{A}^{\boldsymbol{T}}\, o_{\tau}\right)$$

*Recall that $\mathbf{B}^{\dagger}$ denotes the transpose of **B** with normalized columns (i.e., columns that sum to 1). Also note that because the example below only includes 2 time points, only the first and third equations will apply.

Generative Model and Observations

$D = \begin{bmatrix} .75 \\ .25 \end{bmatrix}$; $\mathbf{A} = \begin{bmatrix} .8 & .2 \\ .2 & .8 \end{bmatrix}$; $\mathbf{B} = \begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix}$; $o_{\tau=1} = \begin{bmatrix} 1 \\ 0 \end{bmatrix}$; $o_{\tau=2} = \begin{bmatrix} 0 \\ 1 \end{bmatrix}$

Initialize Approximate Posteriors

$\mathbf{s}_{\tau=1} = \begin{bmatrix} .5 \\ .5 \end{bmatrix}$; $\mathbf{s}_{\tau=2} = \begin{bmatrix} .5 \\ .5 \end{bmatrix}$

Model Inversion: Time Step 1

$$\mathbf{s}_{\tau=1} = \sigma\left(\tfrac{1}{2}\ln\begin{bmatrix} .75 \\ .25 \end{bmatrix} + \tfrac{1}{2}\ln\begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix}\begin{bmatrix} .5 \\ .5 \end{bmatrix} + \ln\begin{bmatrix} .8 & .2 \\ .2 & .8 \end{bmatrix}\begin{bmatrix} 1 \\ 0 \end{bmatrix}\right)$$

$$= \sigma\left(\begin{bmatrix} -.1372 \\ -.6735 \end{bmatrix} + \begin{bmatrix} -.3367 \\ -.3367 \end{bmatrix} + \begin{bmatrix} -.2107 \\ -1.5606 \end{bmatrix}\right)$$

$$= \sigma\left(\begin{bmatrix} -.6846 \\ -2.5709 \end{bmatrix}\right)$$

$$= \begin{bmatrix} .8683 \\ .1317 \end{bmatrix}$$

$$\mathbf{s}_{\tau=2} = \sigma\left(\frac{1}{2}\ln\begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix}\begin{bmatrix} .8683 \\ .1317 \end{bmatrix} + \ln\begin{bmatrix} .8 & .2 \\ .2 & .8 \end{bmatrix}\begin{bmatrix} 0 \\ 0 \end{bmatrix}\right)$$

$$= \sigma\left(\begin{bmatrix} -.9771 \\ -.0649 \end{bmatrix} + \begin{bmatrix} -4.6052 \\ -4.6052 \end{bmatrix}\right)$$

$$= \sigma\left(\begin{bmatrix} -5.5823 \\ -4.6700 \end{bmatrix}\right)$$

$$= \begin{bmatrix} .2865 \\ .7135 \end{bmatrix}$$

Model Inversion: Time Step 2

$$\mathbf{s}_{\tau=1} = \sigma\left(\frac{1}{2}\ln\begin{bmatrix} .75 \\ .25 \end{bmatrix} + \frac{1}{2}\ln\begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix}\begin{bmatrix} .2865 \\ .7135 \end{bmatrix} + \ln\begin{bmatrix} .8 & .2 \\ .2 & .8 \end{bmatrix}\begin{bmatrix} 1 \\ 0 \end{bmatrix}\right)$$

$$= \sigma\left(\begin{bmatrix} -.1372 \\ -.6735 \end{bmatrix} + \begin{bmatrix} -.1619 \\ -.6078 \end{bmatrix} + \begin{bmatrix} -.2107 \\ -1.5606 \end{bmatrix}\right)$$

$$= \sigma\left(\begin{bmatrix} -.5098 \\ -2.8420 \end{bmatrix}\right)$$

$$= \begin{bmatrix} .9115 \\ .0885 \end{bmatrix}$$

$$\mathbf{s}_{\tau=2} = \sigma\left(\tfrac{1}{2}\ln\begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix}\begin{bmatrix} .9115 \\ .0885 \end{bmatrix} + \ln\begin{bmatrix} .8 & .2 \\ .2 & .8 \end{bmatrix}\begin{bmatrix} 0 \\ 1 \end{bmatrix}\right)$$

$$= \sigma\left(\begin{bmatrix}-1.1589\\ -.0409\end{bmatrix} + \begin{bmatrix}-1.5606\\ -0.2107\end{bmatrix}\right)$$

$$= \sigma\left(\begin{bmatrix}-2.7195\\ -0.2516\end{bmatrix}\right)$$

$$= \begin{bmatrix}.0781\\ .9219\end{bmatrix}$$

Exercise 2

Using the equations presented in example 2, and the observations listed below, invert the following generative model. Note again that because this example only includes 2 timepoints, only the first and third equations will apply.

$D = \begin{bmatrix}.5\\ .5\end{bmatrix}$; $\mathbf{A} = \begin{bmatrix}.9 & .1\\ .1 & .9\end{bmatrix}$; $\mathbf{B} = \begin{bmatrix}1 & 0\\ 0 & 1\end{bmatrix}$; $o_{\tau=1} = \begin{bmatrix}1\\ 0\end{bmatrix}$; $o_{\tau=2} = \begin{bmatrix}1\\ 0\end{bmatrix}$

Answers

Answer: Exercise 1

$$\mathbf{s} = \sigma\left(\ln\begin{bmatrix}.75\\ .25\end{bmatrix} + \ln\begin{bmatrix}.8 & .2\\ .2 & .8\end{bmatrix}\begin{bmatrix}1\\ 0\end{bmatrix}\right) = \sigma\left(\ln\begin{bmatrix}.75\\ .25\end{bmatrix} + \ln\begin{bmatrix}.8\\ .2\end{bmatrix}\right)$$

$$= \sigma\left(\ln\begin{bmatrix}.75 \times .8\\ .25 \times .2\end{bmatrix}\right) = \sigma\left(\ln\begin{bmatrix}.6\\ .05\end{bmatrix}\right)$$

$$= \begin{bmatrix}\frac{e^{\ln(.6)}}{e^{\ln(.6)} + e^{\ln(.05)}}\\ \frac{e^{\ln(.05)}}{e^{\ln(.6)} + e^{\ln(.05)}}\end{bmatrix} = \begin{bmatrix}\frac{.6}{.6 + .05}\\ \frac{.05}{.6 + .05}\end{bmatrix} = \begin{bmatrix}.9231\\ .0769\end{bmatrix}$$

Answer: Exercise 2

Model Inversion: Time Step 1

$$\mathbf{s}_{\tau=1} = \sigma\left(\tfrac{1}{2}\ln\begin{bmatrix}.5\\ .5\end{bmatrix} + \tfrac{1}{2}\ln\begin{bmatrix}1 & 0\\ 0 & 1\end{bmatrix}\begin{bmatrix}.5\\ .5\end{bmatrix} + \ln\begin{bmatrix}.9 & .1\\ .1 & .9\end{bmatrix}\begin{bmatrix}1\\ 0\end{bmatrix}\right)$$

$$= \sigma\left(\begin{bmatrix}-.3367\\ -.3367\end{bmatrix} + \begin{bmatrix}-.3367\\ -.3367\end{bmatrix} + \begin{bmatrix}-.0943\\ -2.2073\end{bmatrix}\right)$$

$$= \sigma\left(\begin{bmatrix}-.7677\\ -2.8806\end{bmatrix}\right)$$

$$= \begin{bmatrix}.8922\\ .1078\end{bmatrix}$$

$$\mathbf{s}_{\tau=2} = \sigma\left(\tfrac{1}{2}\ln\begin{bmatrix}1 & 0\\ 0 & 1\end{bmatrix}\begin{bmatrix}.8922\\ .1078\end{bmatrix} + \ln\begin{bmatrix}.9 & .1\\ .1 & .9\end{bmatrix}\begin{bmatrix}0\\ 0\end{bmatrix}\right)$$

$$= \sigma\left(\begin{bmatrix}-0.0515\\-1.0692\end{bmatrix} + \begin{bmatrix}-4.6052\\-4.6052\end{bmatrix}\right)$$

$$= \sigma\left(\begin{bmatrix}-4.6567\\-5.6744\end{bmatrix}\right)$$

$$= \begin{bmatrix}.7345\\.2655\end{bmatrix}$$

Model Inversion: Time Step 2

$$\mathbf{s}_{\tau=1} = \sigma\left(\frac{1}{2}\ln\begin{bmatrix}.5\\.5\end{bmatrix} + \frac{1}{2}\ln\begin{bmatrix}1 & 0\\0 & 1\end{bmatrix}\begin{bmatrix}.7345\\.2655\end{bmatrix} + \ln\begin{bmatrix}.9 & .1\\.1 & .9\end{bmatrix}\begin{bmatrix}1\\0\end{bmatrix}\right)$$

$$= \sigma\left(\begin{bmatrix}-.3367\\-.3367\end{bmatrix} + \begin{bmatrix}-.1475\\-.6446\end{bmatrix} + \begin{bmatrix}-.0943\\-2.2073\end{bmatrix}\right)$$

$$= \sigma\left(\begin{bmatrix}-.5785\\-3.1886\end{bmatrix}\right)$$

$$= \begin{bmatrix}.9315\\.0685\end{bmatrix}$$

$$\mathbf{s}_{\tau=2} = \sigma\left(\tfrac{1}{2}\ln\begin{bmatrix}1 & 0\\0 & 1\end{bmatrix}\begin{bmatrix}.9315\\.0685\end{bmatrix} + \ln\begin{bmatrix}.9 & .1\\.1 & .9\end{bmatrix}\begin{bmatrix}1\\0\end{bmatrix}\right)$$

$$= \sigma\left(\begin{bmatrix}-0.0301\\-1.2724\end{bmatrix} + \begin{bmatrix}-0.943\\-2.2073\end{bmatrix}\right)$$

$$= \sigma\left(\begin{bmatrix}-0.1244\\-3.4797\end{bmatrix}\right)$$

$$= \begin{bmatrix}.9663\\.0337\end{bmatrix}$$

2. Appendix: additional mathematical details

Introduction to Variational Inference

The typical goal of Bayesian inference is to find the posterior distribution $p(s|o)$ – that is, to infer how the states of the world (s) have changed based on new observations (o). However, this requires one to calculate the marginal likelihood $p(o)$, which often involves intractable sums (or integrals in the continuous case). The key idea behind variational inference is to convert this inference problem into an optimization problem. To do so, instead of evaluating the marginal likelihood, we optimize an auxiliary distribution $q(s)$ (sometimes called the recognition distribution, or variational posterior) to approximate the true posterior $p(s|o)$. This is done by using the KL divergence as a measure of the relative difference (in the information-theoretic unit *nats* – the natural log equivalent of bits) between the two distributions:

$$D_{KL}(q(s)||p(s|o)) = \sum_s q(s) \ln \frac{q(s)}{p(s|o)} \tag{A.1}$$

The KL divergence sums over the states of the two distributions so the output is always greater than or equal to zero. When the recognition distribution and the true posterior match, the KL divergence is zero (i.e., when $q(s) = p(s|o)$, $D_{KL}(q(s)||p(s|o)) = 0$). Although, as we do not know the true posterior distribution, this sum also cannot be evaluated. Crucially, however, working from the definition of conditional probabilities, $p(s|o) = \frac{p(o,s)}{p(o)}$, we can introduce a quantity that can be evaluated directly (noting that $= \frac{1}{\mathrm{x}} / \frac{1}{\mathrm{y}} = \frac{\mathrm{y}}{\mathrm{x}}$):

$$\begin{aligned} D_{KL}(q(s)||p(s|o)) &= \sum_s q(s) \ln \frac{q(s)}{\frac{p(o,s)}{p(o)}} \\ &= \sum_s q(s) \ln \frac{q(s)p(o)}{p(o,s)} \\ &= \sum_s q(s) \ln \frac{q(s)}{p(o,s)} + \ln (p(o)) \end{aligned} \tag{A.2}$$

Equation A.2 substitutes the alternate definition of the posterior distribution into equation A.1. With some minor rearrangement, we see that the KL divergence between our approximate posterior and the true posterior is now equal to the KL divergence between our approximate posterior and $p(o, s)$ – which can be viewed as a generative model of how states of the world generate observations – plus the log probability of observations (i.e., the log of the marginal likelihood). This is the critical move. Because we are free to specify the generative model, and $q(s)$ is the variable we seek to optimize

(and can thus be initially set to an arbitrary value; see below), we have access to both the quantities we need to compute the KL divergence. We now introduce a new quantity, variational free energy (VFE), denoted F, and define it in terms of this KL divergence: $F \equiv \sum_s q(s) \ln \frac{q(s)}{p(o,s)}$. The value of the approximate posterior $q(s)$ that minimizes VFE will then be the $q(s)$ that best approximates the true posterior distribution.

$$D_{KL}(q(s)||p(s|o)) = F + \ln(p(o)) \tag{A.3}$$

Equation A.3 rewrites line 3 of equation A.2, but substitutes in VFE as an explicit variable. From this vantage point, we see that when VFE is minimized the KL divergence between the approximate posterior and the true posterior is also minimized, meaning that the approximate posterior is close to the true posterior. Hence, minimizing VFE allows a tractable means of performing approximate Bayesian inference. One way to do this is using gradient descent. That is, one can start $q(s)$ at an arbitrary value and then test neighboring values to find the one that reduces VFE most. Then one can move to that value, search neighboring values again (etc.), and repeat this process until a value for $q(s)$ is found for which no neighboring values further reduce VFE.

Expected Free Energy

Active inference reverses the usual logic of action selection. Instead of asking "which sequence of actions will bring about my preferred outcomes?", it formally asks, "given the assumption that I achieve my preferred outcomes, what course of action am I most likely to pursue?". Within active inference, the answer to this question is the policy (π; i.e., action sequence) that best minimizes a quantity termed expected free energy (EFE). Here we show the most common decompositions of EFE that appear in the active inference literature and describe the workings and the intuition behind each decomposition. EFE is defined in terms of the expected difference between the log of the generative model $p(o,s|\pi)$ and the log of the approximate posterior given a choice of policy $q(s|\pi)$. Equation A.4 shows the decomposition of the EFE of each policy (G_π) into terms often referred to as *epistemic* and *pragmatic* value (intuitively, expected information gain and reward probability under each policy, respectively). It also shows another common decomposition into terms referred to as *risk* and *ambiguity* (similarly corresponding to expected reward and uncertainty minimization under each policy). Note below that, because EFE is calculated with respect to expected outcomes that (by definition) have not yet occurred, observations enter the expectation operator E_q as random variables.

$$\begin{aligned} G_\pi &= \mathrm{E}_{q(o,s|\pi)}[\ln q(s|\pi) - \ln p(o,s|\pi)] \\ &= \mathrm{E}_{q(o,s|\pi)}[\ln q(s|\pi) - \ln p(s|o,\pi)] - \mathrm{E}_{q(o|\pi)}[\ln p(o|\pi)] \\ &\approx \mathrm{E}_{q(o,s|\pi)}[\ln q(s|\pi) - \ln q(s|o,\pi)] - \mathrm{E}_{q(o|\pi)}[\ln p(o)] \end{aligned} \tag{A.4}$$

$$= -E_{q(o,s|\pi)}[\ln q(s|o,\pi) - \ln q(s|\pi)] - E_{q(o|\pi)}[\ln p(o)]$$

$$= \mathrm{E}_{q(o,s|\pi)}[\ln q(o|\pi) - \ln p(o|s,\pi)] - \mathrm{E}_{q(o|\pi)}[\ln p(o)]$$

$$= D_{KL}(q(o|\pi)||p(o)) + E_{q(s|\pi)}[H[p(o|s)]]$$

The second line uses the product rule of probability, $p(o,s|\pi) = p(s|o,\pi)p(o|\pi)$ to rearrange EFE into the epistemic and pragmatic value terms described in the main text. Note, however, that the dependence on π in the second (pragmatic value) term is removed in active inference so that this term can be used to encode preferred observations. In the third line, the pragmatic value term is therefore specified so as not to depend on policies (i.e., becoming $E_{q(o|\pi)}[\ln p(o)]$) so that this prior can be explicitly used to encode prior preferences that are independent of choice of policy. Line 3 also replaces the true posterior ($\ln p(s|o,\pi)$) with an approximate posterior ($\ln q(s|o,\pi)$). Line 4 offers a clearer intuition for epistemic value by flipping the terms inside the first expectation so that it becomes prefixed with a negative sign (i.e., $p(x)[\ln p(x) - \ln q(x)] = -p(x)[\ln q(x) - \ln p(x)]$). Because the epistemic value term is now subtracted from the total, it is clear that to minimize EFE overall an agent must maximize the value of this term by selecting policies that take it into states that maximize the difference between $\ln q(s|o,\pi)$ and $\ln q(s|\pi)$. In other words, the agent is driven to seek out observations that reduce uncertainty about hidden states (i.e., maximize the change from prior to posterior beliefs after a new observation). For example, if you are in a dark room, then the mapping between hidden states and observations is entirely ambiguous. The best way to minimize uncertainty is to turn a light on. This is equivalent to choosing policies that maximize the mutual information between states and observations.

Moving from the expression in line 3, line 5 uses Bayes rule in the denominator $\frac{q(s|\pi)}{q(s|o,\pi)} = \frac{q(s|\pi)q(o|\pi)}{p(o|s,\pi)q(s|\pi)} = \frac{q(o|\pi)}{p(o|s,\pi)}$ to express the same epistemic imperative, but with the conditional probabilities flipped. Note here that, although algebraically $q(s|o,\pi) = \frac{q(o|s,\pi)q(s|\pi)}{q(o|\pi)}$, in this case $q(o|s,\pi)$ and $p(o|s,\pi)$ refer to the same distribution. The epistemic value terms in lines 3, 4, and 5 are formally equivalent since they each express the *mutual information* between hidden states and observations. First noting that $H[p(x)]$ denotes the entropy of a distribution $p(x)$, where $H[p(x)] = -\sum_x p(x)\ln p(x) = -E_{p(x)}[\ln p(x)]$, mutual information can be written as $I(x,y) = H[p(x)] - H[p(x|y)] = H[p(y)] - H[p(y|x)]$. This quantity $I(x,y)$ is symmetric and scores the reduction in uncertainty (entropy) about the value of a variable x afforded by knowledge of another variable y. If two variables are independent, mutual information is zero. Finally, line 6 expresses EFE in the form shown in the main text as *risk* plus *ambiguity*. The first term, risk, is the KL divergence between the observations expected under a policy and prior preferences. The second term, ambiguity, scores the uncertainty in the likelihood mapping between states and observations. Minimizing EFE thus requires agents to select policies that minimize the difference between expected observations and preferred observations (e.g., seeking warmth when it is cold, maximizing reward, etc.) and to take actions that reduce uncertainty (i.e., ambiguity) about the mapping between hidden states and observations (i.e., another way of expressing the drive to maximize information gain). The work to

move between the fifth and the sixth line is somewhat convoluted, so equation A.5 shows it step by step.

$$
\begin{aligned}
G_\pi &= \sum_{o,s} p(o|s)q(s|\pi)(\ln\frac{q(o|\pi)}{p(o|s,\pi)}) - \sum_{o,s} p(o|s)q(s|\pi)\ln p(o) \\
&= \sum_{\mathrm{o,s}} p(o|s)q(s|\pi)(\ln\frac{q(o|\pi)}{p(o)p(o|s,\pi)}) \\
&= \sum_{\mathrm{o,s}} q(o,s|\pi)(\ln\frac{q(o|\pi)}{p(o)}) - \sum_{\mathrm{s}} q(s|\pi)\sum_{\mathrm{o}} p(o|s)\ln p(o|s,\pi) \\
&= \sum_{\mathrm{o}} q(o|\pi)(\ln\frac{q(o|\pi)}{p(o)}) - \sum_{\mathrm{s}} q(s|\pi)\sum_{\mathrm{o}} p(o|s)\ln p(o|s,\pi) \\
&= \sum_{\mathrm{o}} q(o|\pi)(\ln\frac{q(o|\pi)}{p(o)}) + \sum_{\mathrm{s}} \mathrm{q(s|\pi)H}[\,p(o|s)] \\
&= \mathrm{D_{KL}}(q(o|\pi)||p(o)) + \mathrm{E}_{q(\mathrm{S}|\pi)}[\mathrm{H}[p(o|s)]]
\end{aligned}
\qquad \text{A.5}
$$

Line 1 of equation A.5 rewrites line 5 of equation A.4, but makes the summations implied by the expectation operators explicit. Line 2 moves $p(o)$ back into the same expectation. Line 3 expresses the expectation in the first term in terms of an approximate joint distribution $p(o|s)q(s|\pi) = q(o,s|\pi)$, and in the second term separates out $p(o|s,\pi)$. In the first term of line 4 we evaluate the summation over states in the joint distribution $\sum_{\mathrm{o,s}} q(o,s|\pi) = \sum_{\mathrm{o}} q(o|\pi)$, leaving the fraction inside the log untouched as it does not depend on states. In the second term of line 4 we drop the dependency on policies since the likelihood mapping is constant across choices of policy. In line 5 this then allows us to express $-\sum_{\mathrm{o}} p(o|s)\ln p(o|s)$ in terms of entropy $H[p(o|s)]$, and rewrite $\sum_o q(o|\pi) ln \frac{q(o|\pi)}{p(o)}$ in terms of the KL divergence (defined in the main text) between prior preferences and observations expected under each policy $D_{KL}(q(o|\pi)||p(o))$. Finally, because entropy is a negative quantity, we swap the sign between the two terms, leaving us with the canonical form of EFE as *risk* plus *ambiguity* in line 6 – where lower risk indicates a higher probability of preferred outcomes under a policy and lower ambiguity indicates more precise (informative) observations expected under a policy.

It is important to highlight here that generative models in active inference also maintain confidence estimates for the model parameters themselves, via a form of distribution called a Dirichlet distribution that encodes priors over these parameters (see main text for an introduction to this type of distribution). These distributions contain what are called concentration parameters, where higher concentration parameter values indicate lower uncertainty in the parameters of each distribution. The above expressions of EFE assume that the concentration parameters are saturated (i.e., that they

are maximally precise), and hence that there is no uncertainty. However, when there is uncertainty in parameters (as is the case for real organisms), agents must also learn the values for those parameters via the selection of appropriate policies. This means that parameter uncertainty now enters the equation for EFE. For example, equation A.6 shows the form of EFE when the parameters of the likelihood $p(o|s)$ must be learned. Note that this likelihood is termed the **A** matrix in active inference models (see **Table 1**).

$$
\begin{aligned}
G_\pi \;&= \mathrm{E}_{\overline{\mathrm{q}}}[\ln q(s, \mathbf{A}|\pi) - \ln p(o, s, \mathbf{A}|\pi)] && \text{A.6}\\
&= \mathrm{E}_{\overline{\mathrm{q}}}[\,\ln q(s|\pi) + \ln q(\mathbf{A}) - \ln p(\mathbf{A}|s, o, \pi) - \ln p(s|o, \pi) - \ln p(o)]\\
&\approx \mathrm{E}_{\overline{\mathrm{q}}}[\,\ln q(s|\pi) + \ln q(\mathbf{A}) - \ln q(\mathbf{A}|s, o, \pi) - \ln q(s|o, \pi) - \ln p(o)]\\
&= \mathrm{E}_{\overline{\mathrm{q}}}[\,\ln q(s|\pi) + \ln q(\mathbf{A}) - \ln q(\mathbf{A}|s, o, \pi) - \ln q(s|o, \pi)] - \mathrm{E}_{\overline{\mathrm{q}}}[\ln p(o)]\\
&= \mathrm{E}_{\overline{\mathrm{q}}}[\,\ln q(s|\pi) - \ln q(s|o, \pi)] + \mathrm{E}_{\overline{\mathrm{q}}}[\ln q(\mathbf{A}) - \ln q(\mathbf{A}|s, o, \pi)] - \mathrm{E}_{\overline{\mathrm{q}}}[\ln p(o)]\\
&= -\mathrm{E}_{\overline{\mathrm{q}}}[\,\ln q(s|o, \pi) - \ln q(s|\pi)] - \mathrm{E}_{\overline{\mathrm{q}}}[\ln q(\mathbf{A}|s, o, \pi) - \ln q(\mathbf{A})] - \mathrm{E}_{\overline{\mathrm{q}}}[\ln p(o)]
\end{aligned}
$$

Here $\overline{q} = q(o, s, \mathbf{A}|\pi)$. Line 1 shows the form of EFE when **A** is treated as a random variable. Line 2 breaks the approximate posterior and generative model into separate terms using the product rule of probability $p(o, s, \mathbf{A}|\pi) = p(\mathbf{A}|s, o, \pi)p(s|o, \pi)p(o)$. It also uses the mean field approximation – which assumes that the approximate posterior factorizes into the product of independent marginal distributions – to express the approximate joint distribution as $q(s, \mathbf{A}|\pi) = q(s|\pi)q(\mathbf{A})$. Line 3 approximates line two, replacing the exact posteriors $p(\mathbf{A}|s, o, \pi)$ and $p(s|o, \pi)$ with approximate posteriors $q(\mathbf{A}|s, o, \pi)$ and $q(s|o, \pi)$. Line 4 takes pragmatic value $\mathrm{E}_{\overline{\mathrm{q}}}[\ln p(o)]$ out of the first expectation term. Line 5 breaks the first expectation in line 4 into two quantities. The first, $\mathrm{E}_{\overline{\mathrm{q}}}[\,\ln q(s|\pi) - \ln q(s|o, \pi)]$) is the epistemic value term that we saw in the previous expression of EFE without uncertainty in the parameters. As we now have two types of epistemic value, to distinguish them we call the first *salience* and the second *novelty*. Salience scores the reduction of uncertainty about states afforded by observations (driving "state exploration" behavior), while novelty $\mathrm{E}_{\overline{\mathrm{q}}}[\ln q(\mathbf{A}) - \ln q(\mathbf{A}|s, o, \pi)]$ scores the reduction in uncertainty about parameters of the generative model afforded by states and observations (driving "parameter exploration" behavior; see (Schwartenbeck et al., 2019)). As with line 3 in equation A.4, line 6 here flips the terms inside the first and second expectation to make the salience and novelty terms negative (i.e., such that maximizing these terms brings their value closer to zero). This makes it clearer why maximizing the difference between prior and posterior beliefs – here about both states and parameters – will minimize EFE. The parameters of the model are the sufficient statistics of Dirichlet distributions, which, in the case of learning **A,** essentially count the number of times a particular categorical state is inferred when a particular outcome is observed (proportional to the posterior probability over each state). Like the epistemic value term in the previous expression of EFE, to minimize EFE here agents must maximize both salience and novelty by seeking out observations that 1) reduce uncertainty about hidden states,

and/or 2) reduce uncertainty about parameters. The reduction of uncertainty of the parameters via the maximization of novelty encourages agents to explore novel parts of the state space that are less familiar (i.e., states that have low concentration parameters).

The Softmax Function

The softmax (or normalized exponential) function, denoted by σ, takes a vector x of length k and normalizes the vector such that the elements 1) have a monotonic relationship with the elements of the input vector, and 2) sum to one and can thus be treated as a categorical probability distribution over 1 to k mutually exclusive states. Importantly, the vector is weighted by a precision parameter denoted by γ, which controls the extent to which differences between the elements are amplified or dampened by the exponential.

$$\sigma(\mathrm{x}) = \frac{\mathrm{e}^{\gamma \mathrm{x}_\mathrm{i}}}{\sum_\mathrm{k} \mathrm{e}^{\gamma \mathrm{x}_\mathrm{k}}} \tag{A.7}$$

For example, for $x = [1\ 2\ 3\ 4]^\mathrm{T}$, when $\gamma = 1$, $\sigma(\mathrm{x}) = [0.0321\ 0.0871\ 0.2369\ 0.6439]^\mathrm{T}$.

When $\gamma = 0.1$, $\sigma(\mathrm{x}) = [0.2138\ 0.2363\ 0.2612\ 0.2887]^\mathrm{T}$.

When $\gamma = 2$, $\sigma(x) = [0.021\ 0.0158\ 0.1171\ 0.8650]^\mathrm{T}$.

The Gamma Function

The gamma function (denoted by Γ) is a generalization of the factorial function that, unlike the factorial function (whose domain is restricted to positive integers), is well defined for complex and real valued (i.e., non-integer) inputs (except for the negative integers). For positive, real-valued, and complex numbers, the gamma function is defined by the following definite integral.

$$\Gamma(z) = \int_0^\infty x^{z-1}\, e^{-x} dx \tag{A.8}$$

For the positive integers, the gamma function reduces to the factorial function (i.e., $\mathrm{n}! = \mathrm{n} \times (\mathrm{n}-1) \times (\mathrm{n}-2) \times (\mathrm{n}-3) \times \ldots \times 3 \times 2 \times 1$) but is shifted by 1.

$$\Gamma(\mathrm{n}) = (n-1)! \tag{A.8}$$

$$\Gamma(2) = (2-1)! = 1 \times 1 = 1$$

$$\Gamma(3) = (3-1)! = 2 \times 1 = 2$$

$$\Gamma(4) = (4-1)! = 3 \times 2 \times 1 = 6$$

To gain an intuition for how the gamma function relates to the factorial function , we will use integration by parts (i.e., $\int_b^a f(x)g'(x)dx = [f(x)g(x)]_b^a - \int_b^a f'(x)g(x)dx$) to show that $\Gamma(n+1) = (n)!$.

$$\Gamma(z+1) = \int_0^\infty x^z\, e^{-x} dx = [-x^z e^{-x}]_0^\infty + \int_0^\infty z x^{z-1}\, e^{-x} dx \tag{A.9}$$

$$= \lim_{x\to\infty}(-x^2 e^{-x}) - (-0e^{-0}) + z\int_0^\infty x^{z-1}\, e^{-x} dx$$

Because e^{-x} grows faster than x^z, the first term is sent to zero leaving us with the following.

$$\Gamma(z+1) = z\int_0^\infty x^{z-1}\, e^{-x} dx \tag{A.10}$$

$$= z\,\Gamma(z)$$

If we plug in some examples, we see immediately that this is equivalent to the factorial function.

$$\Gamma(2+1) = 2\,\Gamma(2) = 2\,(2-1)! = 2 = 2! \tag{A.11}$$

$$\Gamma(3+1) = 3\,\Gamma(3) = 3\,(3-1)! = 6 = 3!$$

$$\Gamma(4+1) = 4\,\Gamma(4) = 4\,(4-1)! = 24 = 4!$$

In the context of the Dirichlet distribution, the gamma function is used to define a normalization constant that accounts for the combinatorics of the concentration parameters, which are positive real-valued numbers (i.e., they can take non-integer values), most of which are not defined when using the factorial function, which is why the gamma function is employed.

References

Adams, R., Shipp, S., & Friston, K. (2013). Predictions not commands: active inference in the motor system. *Brain Structure and Function,* 218, 611-643.
Addicott, M., Pearson, J., Sweitzer, M., Barack, D., & Platt, M. (2017). A Primer on Foraging and the Explore/Exploit Trade-Off for Psychiatry Research*.* 42, 1931-1939.
Andrews, M. (2020). The Math is not the Territory: Navigating the Free Energy Principle. *pittphilsci:18315*.
Attias, H. (2000). *A variational baysian framework for graphical models*.
Attias, H. (2003). *Planning by Probabilistic Inference.* Paper presented at the Proc. of the 9th Int. Workshop on Artificial Intelligence and Statistics.
Badcock, P. B., Friston, K. J., Ramstead, M. J. D., Ploeger, A., & Hohwy, J. (2019). The hierarchically mechanistic mind: an evolutionary systems theory of the human brain, cognition, and behavior. *Cogn Affect Behav Neurosci,* 19(6), 1319-1351.
Barto, A., Mirolli, M., & Baldassarre, G. (2013). Novelty or Surprise? *Frontiers in Psychology,* 4.
Beal, M. J. (2003). *Variational algorithms for approximate Bayesian inference*: university of London London.
Bekinschtein, T., Dehaene, S., Rohaut, B., Tadel, F., Cohen, L., & Naccache, L. (2009). Neural signature of the conscious processing of auditory regularities. *Proceedings of the National Academy of Sciences of the United States of America,* 106, 1672-1677.
Bishop, C. M. (2006). *Pattern recognition and machine learning*. New York: Springer.
Bogacz, R. (2017). A tutorial on the free-energy framework for modelling perception and learning. *Journal of mathematical psychology,* 76, 198-211.
Botvinick, M., & Toussaint, M. (2012). Planning as inference. *Trends Cogn Sci.,* 16(10), 485-488.
Brown, L. D. (1981). A Complete Class Theorem for Statistical Problems with Finite-Sample Spaces. *Annals of Statistics,* 9(6), 1289-1300.
Brown, T. H., Zhao, Y., & Leung, V. (2010). Hebbian plasticity. In *Encyclopedia of Neuroscience* (pp. 1049-1056).
Bruineberg, J., Dolega, K., Dewhurst, J., & Baltieri, M. (2020). The Emperor's New Markov Blankets. *pittphilsci:18467*.
Bucci, A., & Grasso, M. (2017). Sleep and dreaming in the predictive processing framework. In T. Metzinger & W. Wiese (Eds.), *Philosophy and Predictive Processing*): Johannes Gutenberg-Universität Mainz.
Buckley, C. L., Sub Kim, C., McGregor, S., & Seth, A. K. (2017). The free energy principle for action and perception: A mathematical review. *Journal of Mathematical Psychology,* 81, 55-79.
Burr, C., & Jones, M. (2016). The body as laboratory: Prediction-error minimization, embodiment, and representation. *Philosophical Psychology,* 29(4), 586-600.
Carlin, B. P., & Louis, T. A. (1998). *Bayes and empirical Bayes methods for data analysis*. Boca Raton: Chapman & Hall/CRC.
Clark, A. (2013). Whatever next? Predictive brains, situated agents, and the future of cognitive science. *The Behavioral and brain sciences,* 36, 181-204.
Clark, A. (2015). Surfing uncertainty: Prediction, action, and the embodied mind.
Clark, A. (2017). How to Knit Your Own Markov Blanket. In T. K. Metzinger & W. Wiese (Eds.), *Philosophy and Predictive Processing*). Frankfurt am Main: MIND Group.
Clark, J. E., Watson, S., & Friston, K. J. (2018). What is mood? A computational perspective. *Psychological Medicine*, 1-8.
Da Costa, L., Parr, T., Sajid, N., Veselic, S., Neacsu, V., & Friston, K. J. (2020). Active inference on discrete state-spaces: A synthesis. *Journal of Mathematical Psychology,* 99, 102447.

Da Costa, L., Sajid, N., Parr, T., Friston, K. J., & Smith, R. (2020). The relationship between dynamic programming and active inference: the discrete, finite-horizon case. *arXiv*, arXiv:2009.08111.
Dauwels, J. (2007). On variational message passing on factor graphs. *IEEE International Symposium on Information Theory*, 2546-2550.
de Vries, B., & Friston, K. J. (2017). A Factor Graph Description of Deep Temporal Active Inference. *Front Comput Neurosci,* 11, 95.
Fabry, R. E. (2017). Transcending the evidentiary boundary: Prediction error minimization, embodied interaction, and explanatory pluralism. *Philosophical Psychology,* 30(4), 395-414.
Friston, K. J. (2019). A free energy principle for a particular physics. *arXiv:1906.10184*.
Friston, K. J., FitzGerald, T., Rigoli, F., Schwartenbeck, P., O Doherty, J., & Pezzulo, G. (2016). Active inference and learning. *Neuroscience and biobehavioral reviews,* 68, 862-879.
Friston, K. J., FitzGerald, T., Rigoli, F., Schwartenbeck, P., & Pezzulo, G. (2017). Active Inference: A Process Theory. *Neural Computation,* 29, 1-49.
Friston, K. J., Lin, M., Frith, C., Pezzulo, G., Hobson, J., & Ondobaka, S. (2017). Active Inference, Curiosity and Insight. *Neural Computation,* 29, 2633-2683.
Friston, K. J., Litvak, V., Oswal, A., Razi, A., Stephan, K. E., van Wijk, B. C. M., et al. (2016). Bayesian model reduction and empirical Bayes for group (DCM) studies. *NeuroImage,* 128, 413-431.
Friston, K. J., Mattout, J., Trujillo-Barreto, N., Ashburner, J., & Penny, W. (2007). Variational free energy and the Laplace approximation. *NeuroImage,* 34, 220-234.
Friston, K. J., Parr, T., & de Vries, B. (2017). The graphical brain: Belief propagation and active inference. *Network Neuroscience,* 1, 381-414.
Friston, K. J., Rosch, R., Parr, T., Price, C., & Bowman, H. (2018). Deep temporal models and active inference. *Neuroscience & Biobehavioral Reviews,* 90, 486-501.
Hesp, C., Smith, R., Allen, M., Friston, K. J., & Ramstead, M. J. D. (2020). Deeply Felt Affect: The Emergence of Valence in Deep Active Inference. *Neural Computation*, 1-49.
Hobson, J., & Friston, K. (2012). Waking and dreaming consciousness: neurobiological and functional considerations. *Progress in neurobiology,* 98, 82-98.
Hobson, J., Hong, C.-H., & Friston, K. (2014). Virtual reality and consciousness inference in dreaming. *Frontiers in Psychology,* 5, 1133.
Hohwy, J. (2014). The Predictive Mind.
Hohwy, J. (2016). The Self-Evidencing Brain. *Noûs,* 50, 259-285.
Hohwy, J., Paton, B., & Palmer, C. (2016). Distrusting the present. *Phenomenology and the Cognitive Sciences,* 15(3), 315-335.
Kaplan, R., & Friston, K. J. (2018). Planning and navigation as active inference. *Biol Cybern,* 112(4), 323-343.
Kuczma, M., & Gilányi, A. (2009). *An introduction to the theory of functional equations and inequalities : Cauchy's equation and Jensen's inequality* (2nd ed.). Basel ; Boston, MA: Birkäeuser.
Loeliger, H. A. (2004). An introduction to factor graphs. *IEEE Signal Processing Magazine,* 21(1), 28-41.
Markovic, D., Stojic, H., Schwoebel, S., & Kiebel, S. (2021). An empirical evaluation of active inference in multi-armed bandits. *arXiv:2101.08699*.
Mathys, C. D., Lomakina, E. I., Daunizeau, J., Iglesias, S., Brodersen, K. H., Friston, K. J., et al. (2014). Uncertainty in perception and the Hierarchical Gaussian Filter. *Front Hum Neurosci,* 8, 825.
Millidge, B. (2019). Combining Active Inference and Hierarchical Predictive Coding: A Tutorial Introduction and Case Study. *PsyArXiv*.
Mirza, M. B., Adams, R. A., Mathys, C., & Friston, K. J. (2018). Human visual exploration reduces uncertainty about the sensed world. *PLOS ONE,* 13, e0190429.
Oudeyer, P.-Y., & Kaplan, F. (2007). What is intrinsic motivation? a typology of computational approaches. *Frontiers in Neurorobotics,* 1, 6.

Parr, T., & Friston, K. J. (2017a). Uncertainty, epistemics and active inference. *Journal of the Royal Society, Interface,* 14.
Parr, T., & Friston, K. J. (2017b). Working memory, attention, and salience in active inference. *Scientific Reports,* 7, 14678.
Parr, T., & Friston, K. J. (2018a). The Anatomy of Inference: Generative Models and Brain Structure. *Frontiers in Computational Neuroscience,* 12, 90.
Parr, T., & Friston, K. J. (2018b). The Discrete and Continuous Brain: From Decisions to Movement—and Back Again. *Neural Computation*, 1-29.
Parr, T., Markovic, D., Kiebel, S., & Friston, K. J. (2019). Neuronal message passing using Mean-field, Bethe, and Marginal approximations. *Scientific Reports,* 9, 1889.
Pezzulo, G., Rigoli, F., & Friston, K. J. (2015). Active Inference, homeostatic regulation and adaptive behavioural control. *Progress in neurobiology,* 134, 17-35.
Pezzulo, G., Rigoli, F., & Friston, K. J. (2018). Hierarchical Active Inference: A Theory of Motivated Control. *Trends in Cognitive Sciences,* 22, 294-306.
Ramachandran, V. S. (1988). Perceiving shape from shading. *Scientific American,* 259(2), 76–83.
Rigoux, L., Stephan, K. E., Friston, K. J., & Daunizeau, J. (2014). Bayesian model selection for group studies - revisited. *Neuroimage,* 84, 971-985.
Sajid, N., Ball, P., Parr, T., & Friston, K. (2021). Active Inference: Demystified and Compared. *Neural Computation*, 1-39.
Sales, A. C., Friston, K. J., Jones, M. W., Pickering, A. E., & Moran, R. J. (2019). Locus Coeruleus tracking of prediction errors optimises cognitive flexibility: An Active Inference model. *PLoS Comput Biol,* 15(1), e1006267.
Schmidhuber, J. (2006). Developmental robotics, optimal artificial curiosity, creativity, music, and the fine arts. *Connection Science,* 18(2), 173-187.
Schwartenbeck, P., FitzGerald, T., Mathys, C., Dolan, R., & Friston, K. J. (2015). The Dopaminergic Midbrain Encodes the Expected Certainty about Desired Outcomes. *Cerebral Cortex,* 25, 3434-3445.
Schwartenbeck, P., Passecker, J., Hauser, T. U., FitzGerald, T. H., Kronbichler, M., & Friston, K. J. (2019). Computational mechanisms of curiosity and goal-directed exploration. *Elife,* 8.
Smith, R., Badcock, P. B., & Friston, K. J. (2020). Recent advances in the application of predictive coding and active inference models within clinical neuroscience. *Psychiatry and Clinical Neurosciences*.
Smith, R., Khalsa, S. S., & Paulus, M. P. (2019). An Active Inference Approach to Dissecting Reasons for Nonadherence to Antidepressants. *Biol Psychiatry Cogn Neurosci Neuroimaging,* (In Press).
Smith, R., Kirlic, N., Stewart, J. L., Touthang, J., Kuplicki, R., Khalsa, S. S., et al. (2021). Greater decision uncertainty characterizes a transdiagnostic patient sample during approach-avoidance conflict: a computational modeling approach. *Journal of Psychiatry & Neuroscience,* 46(1), E74-E87.
Smith, R., Kuplicki, R., Feinstein, J., Forthman, K. L., Stewart, J. L., Paulus, M. P., et al. (2020). A Bayesian computational model reveals a failure to adapt interoceptive precision estimates across depression, anxiety, eating, and substance use disorders. *PLoS Computational Biology,* 16(12), e1008484.
Smith, R., Lane, R., Parr, T., & Friston, K. J. (2019). Neurocomputational mechanisms underlying emotional awareness: Insights afforded by deep active inference and their potential clinical relevance. *Neurosci Biobehav Rev,* 107, 473-491.
Smith, R., Parr, T., & Friston, K. J. (2019). Simulating Emotions: An Active Inference Model of Emotional State Inference and Emotion Concept Learning. *Front Psychol,* 10, 2844.
Smith, R., Schwartenbeck, P., Parr, T., & Friston, K. J. (2020). An Active Inference Approach to Modeling Structure Learning: Concept Learning as an Example Case. *Front Comput Neurosci,* 14, 41.
Smith, R., Schwartenbeck, P., Stewart, J. L., Kuplicki, R., Ekhtiari, H., Investigators, T., et al. (2020). Imprecise Action Selection in Substance Use Disorder: Evidence for Active Learning Impairments When Solving the Explore-exploit Dilemma. *Drug and Alcohol Dependence,* 215, 108208.

Stephan, K. E., Penny, W. D., Daunizeau, J., Moran, R. J., & Friston, K. J. (2009). Bayesian model selection for group studies. *Neuroimage,* 46(4), 1004-1017.
Tononi, G., & Cirelli, C. (2014). Sleep and the Price of Plasticity: From Synaptic and Cellular Homeostasis to Memory Consolidation and Integration. *Neuron,* 81, 12-34.
Tschantz, A., Barca, L., Maisto, D., Buckley, C. L., Seth, A. K., & Pezzulo, G. (2021). Simulating homeostatic, allostatic and goal-directed forms of interoceptive control using Active Inference. *bioRxiv*, 2021.2002.2016.431365.
Tu, S. (2014). *The dirichlet-multinomial and dirichlet-categorical models for bayesian inference.* Berkeley, CA: Computer Science Division, UC Berkeley.
Wald, A. (1947). An Essentially Complete Class of Admissible Decision Functions. *The Annals of Mathematical Statistics*, 549-555.
Whyte, C., & Smith, R. (2020). The Predictive Global Neuronal Workspace: A Formal Active Inference Model of Visual Consciousness. *Progress in Neurobiology*, 2020.2002.2011.944611.
Wilson, R., Geana, A., White, J., Ludvig, E., & Cohen, J. (2014). Humans use directed and random exploration to solve the explore-exploit dilemma. *Journal of experimental psychology. General,* 143, 2074-2081.
Winn, J., & Bishop, C. M. (2005). Variational message passing. *Journal of Machine Learning Research,* 6, 661-694.
Zeidman, P., Jafarian, A., Seghier, M. L., Litvak, V., Cagnan, H., Price, C. J., et al. (2019). A guide to group effective connectivity analysis, part 2: Second level analysis with PEB. *Neuroimage,* 200, 12-25.