# POLICY OPTIMIZATION EMERGES FROM NOISY REPRESENTATION LEARNING

**Jonah W. Brenner**
Harvard College
Cambridge, MA 02138
jbrenner@college.harvard.edu

**Chenguang Li**
Biophysics Program
Harvard University
Cambridge, MA 02138
chenguang_li@fas.harvard.edu

**Gabriel Kreiman**
Boston Children's Hospital
Harvard Medical School
Boston, MA 02115
gabriel.kreiman@childrens.harvard.edu

## ABSTRACT

Nervous systems learn representations of the world and policies to act within it. We present a framework that uses reward-dependent noise to facilitate policy optimization in representation learning networks. These networks balance extracting normative features and task-relevant information to solve tasks. Moreover, their representation changes reproduce several experimentally observed shifts in the neural code during task learning. Our framework presents a biologically plausible mechanism for emergent policy optimization amid evidence that representation learning plays a vital role in governing neural dynamics. Code is available at: NeuralThermalOptimization.

## 1 INTRODUCTION

Nervous systems learn representations of the world. They configure their weights to extract certain information from the environment: for instance, the presence of objects or their temporal dynamics (Li & DiCarlo (2008); de Vries & Wurm (2023)). This learning is driven by synaptic plasticity (Markram et al. (1997); Bi & Poo (1998); Dan & Poo (2004)). Representation learning objectives provide a normative framework for proposing plasticity rules (Lipshutz et al. (2023); van Hateren (1992); Pehlevan et al. (2015)). Mounting evidence suggests that these objectives largely determine synaptic plasticity. Rules derived from these objectives emulate experimentally observed spike-rate dependent plasticity (Pehlevan & Chklovskii (2014); Pehlevan et al. (2017); Qin et al. (2023); Sengupta et al. (2018); Halvagal & Zenke (2023); Tang et al. (2024); Millidge et al. (2024)). At the same time, these rules reproduce population-level changes to the neural code seen under a variety of experimental conditions (Raju et al. (2024); Qin et al. (2023); Halvagal & Zenke (2023)). However, organisms do not only build representations of the environment; they also use them to optimize goal-directed behavior. The mechanisms that underlie this behavioral learning remain unclear.

This article introduces **NE**ural **T**hermal **O**ptimization (NETO), a framework for how policy optimization can emerge from noisy representation learning. NETO treats the brain's intrinsic noise as a feature rather than a bug. It hypothesizes that the noise in the nervous system is reward-dependent, decreasing in magnitude when the nervous system's weight configuration yields a rewarding policy. This modulation, akin to thermal cooling, alters the attractive points in weight space, encouraging the network to optimize its policy while learning a representation objective.

We illustrate the framework by considering an agent whose one-layer network operates under a simple, linear representation learning objective. We analyze the agent's learning process in two tasks. In the first, the features defined by the agent's objective capture all task-relevant information. Through analytical arguments and simulations, we show that the agent is guaranteed to learn an optimal policy by diffusively searching the space of equivalent representations. In the second task,

the agent needs to learn features not captured by its representation learning objective to perform well. We show that NETO balances normative and task-relevant feature learning to solve the task. These results suggest that NETO is a plausible theory for policy learning in nervous systems.

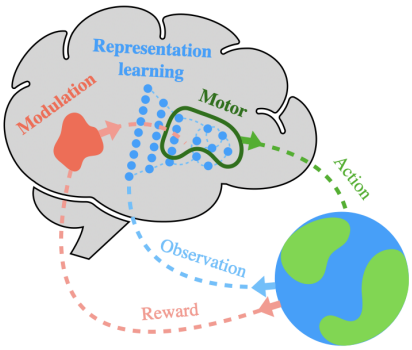


**Figure 1:** *A schematic of the Neural Thermal Optimization framework. A representation learning network (blue) receives observations from the environment and extracts features. A motor function (green) determines an action given the network activities, defining the agent's policy. A modulatory system (red) tunes the noise in the representation learning network according to its estimate of the policy reward.*

## 2 Related Work

A key question in theoretical neuroscience is how organisms learn to interact in environments. Early work showed that reward-modulated spike-timing dependent plasticity (STDP) rules can implement reinforcement learning (Florian (2007); Legenstein et al. (2008)). This model gained traction because its structure is consistent with experimental observations. Organisms with vastly different nervous systems, from the *Caenorhabditis elegans* worm to humans, possess neuromodulatory systems (Shine et al. (2019); Qin & Wheeler (2007); Alcedo & Prahlad (2020)). Moreover, their STDP rules are consistent with those measured across organisms. However, artificial networks operating under this framework can only solve simple tasks like discriminating different spike trains.

Representation learning may offer a path forward. In artificial agents, representation learning objectives greatly improve task-learning capabilities (Jaderberg et al. (2016); Ha & Schmidhuber (2018)). For biological models, promising work showed that spiking neural networks that learn policies from a world model solve simple Atari games (Capone & Paolucci (2024)). However, this work was based on rules with little experimental validation and necessitated separate modules for representation and policy learning.

We also use representation learning as a foundation but do not impose an additional module for policy learning. Instead, we return to the idea that a neuromodulatory system acts on a network operating under experimentally validated plasticity rules. However, we allow the modulation to tune the nervous system's intrinsic noise. The use of noise as a computational tool is not new (Kappel et al. (2017); Maass (2014); Li et al. (2024)). But, to our knowledge, the key idea that noise can act as an effective weight temperature, "cooling" the network in rewarding configurations, has not yet been explored.

## 3 Framework

### 3.1 The Markov Decision Process

A natural framework for an organism interacting in an environment is the Markov Decision Process (MDP). We define an MDP with the tuple $(\mathcal{S}, \mathcal{A}, P, r, \gamma)$. The environment's dynamics are defined by the state space $\mathcal{S}$, the agent's action space $\mathcal{A}$, and $P(s'|s, a)$, the probability of transitioning from

a state $s$ to a state $s'$ given an action $a$. Upon this transition, the agent receives a reward specified by the function $r(s, a)$.

The agent acts according to a policy $\pi : \mathcal{S} \times \mathcal{A} \to [0, 1]$ which describes the distribution of actions that it could take in a state $s$. We denote this distribution $\pi(a|s)$. Its goal is to learn a policy that maximizes its reward over a finite time horizon $T$. If we let the subscript $t$ denote the timestep, the reward is given by $R(\pi) = \mathbb{E}[\sum_{t=1}^{T} \gamma^t r(s_t, a_t)]$, where $a_t \sim \pi(a|s_t)$ per the definition of the policy, and $\gamma \in (0, 1]$ is the temporal discount factor.

### 3.2 Neural Thermal Optimization

Here, we introduce Neural Thermal Optimization (NETO). In NETO agents, a single noisy neural network learns representations and controls actions, while a neuromodulatory system tunes the noise. As such, NETO has three components: a representation learning network, a motor function that defines the agent's policy, and reward-dependent noise controlled by a neuromodulatory system. We illustrate each in Figure 1, and introduce them below, along with the example that we consider throughout this work.

#### 3.2.1 A representation learning objective governs the network dynamics

Consistent with biology, we consider an online agent, meaning that it learns through interacting continuously with its environment. We begin by describing its representation learning network. Let the neural network be parameterized by activities $\boldsymbol{x} \in \mathbb{R}^N$ and weights $\theta$. This network can have any architecture. The network's goal is to extract features from the environment. It builds representations of the MDP, in principle capturing aspects of the distribution of $s$ or the dynamics of the state transitions defined by $P$.

The network extracts features by optimizing a representation learning objective $\mathcal{L}(\boldsymbol{x}, \theta, s)$. We refer to the features that it learns as "normative" because $\mathcal{L}$ specifies how the network weights should change to extract them. Because the agent is online, its activities and weights respond together to each state $s$. These responses are noisy, and are given by the stochastic differential equations

$$d\boldsymbol{x} = -\eta_x \frac{\partial \mathcal{L}}{\partial \boldsymbol{x}} dt + \sigma_x dB \tag{1}$$

$$d\theta = -\eta_\theta \frac{\partial \mathcal{L}}{\partial \theta} dt + \sigma_\theta dB \tag{2}$$

where the $dB$ are independent standard Wiener processes in the necessary number of dimensions, $\eta_x$ and $\eta_\theta$ are the weight and activity learning rates, and $\sigma_x$ and $\sigma_\theta$ are the diffusion tensors of the stochastic process. Typically, $\eta_x >> \eta_\theta$ such that the activity timescale is much faster than the weight timescale. The activities respond to the state $s$, and then the weights change according to these responses.

We can make this discussion more concrete by introducing our simple example agent. As we will see, this agent exhibits many important features of NETO while remaining easily interpretable and analytically tractable. We let our agent's representation learning network optimize the Similarity Matching (SM) objective, introduced in (Pehlevan et al. (2015)). Nicely, this choice bestows it with Hebbian weight updates, which emphasizes that our model is biologically plausible.

The SM objective encourages the network to project its inputs onto their principal subspace. Consider a one-layer network with activities $\boldsymbol{x} \in \mathbb{R}^2$ and weights $\theta = (\boldsymbol{W}_0, \boldsymbol{W}_1)$. The first weight matrix $\boldsymbol{W}_0$ is a feedforward matrix, and $\boldsymbol{W}_1$ is a matrix of lateral inhibitory connections[1]. Suppose that the agent receives $D$-dimensional state vectors $\boldsymbol{s} \in \mathbb{R}^D$ from the environment. Then $\boldsymbol{W}_0 \in \mathbb{R}^{D\times 2}$ and $\boldsymbol{W}_1 \in \mathbb{R}^{2\times 2}$. Previous work (Pehlevan et al. (2017)) showed that the network satisfies the SM objective if it optimizes

$$\mathcal{L}(\boldsymbol{x}, \boldsymbol{W}_0, \boldsymbol{W}_1, \boldsymbol{s}) = \mathrm{Tr}(\boldsymbol{W}_0^T \boldsymbol{W}_0) - \frac{1}{2}\mathrm{Tr}(\boldsymbol{W}_1^T \boldsymbol{W}_1) - 2\boldsymbol{x}^T \boldsymbol{W}_0 \boldsymbol{s} + \boldsymbol{x}^T \boldsymbol{W}_1 \boldsymbol{x} \tag{3}$$

[1] $W_1$ must be symmetric and positive-definite.

giving it the dynamics

$$d\boldsymbol{x} = \eta_x(\boldsymbol{W}_0\boldsymbol{s} - \boldsymbol{W}_1\boldsymbol{x})dt \tag{4}$$

$$d\boldsymbol{W}_0 = \eta_\theta(\boldsymbol{x}\boldsymbol{s}^T - \boldsymbol{W}_0)dt + \sqrt{\eta_\theta}\sigma_\theta dB \tag{5}$$

$$d\boldsymbol{W}_1 = \eta_\theta(\boldsymbol{x}\boldsymbol{x}^T - \boldsymbol{W}_1)dt + \sqrt{\eta_\theta}\sigma_\theta dB \tag{6}$$

For simplicity, we set the activity noise to zero to make the network's activity dynamics deterministic. We will see soon that this also makes the agent's policy deterministic, which will simplify future analysis. We also let $\sigma_\theta$ be a scalar rather than a tensor. At each timestep of the MDP, the activities evolve under the differential equation (4) until they converge, followed by a discrete weight update.

#### 3.2.2 A MOTOR FUNCTION DEFINES THE AGENT'S POLICY

Under NETO, the same neural network learns representations and controls actions. We now introduce the motor function, which defines the agent's policy. Let $M : \mathbb{R}^N \to \mathcal{A}$ be a motor function that maps the network activities to an action. We assume that $M$ is independent of time[2]. Biologically, we can imagine that this motor function arises due to projections from the neural activities $\boldsymbol{x}$ to a motor control system or from movements of the muscles directly controlled by $\boldsymbol{x}$.

This motor function defines the agent's policy, the distribution of possible actions $a$ that it could take in response to a state $s$. In the general case of noisy activity dynamics, a distribution $f_\theta(\boldsymbol{x}|s)$ describes the possible network responses to a state $s$. The policy induced by the motor function is given by the distribution of the actions associated with these network responses

$$\pi_\theta(a|s) = \int f_\theta(\boldsymbol{x}|s)\ \delta(a - M(\boldsymbol{x}))\ d\boldsymbol{x} \tag{7}$$

where $\delta(.)$ is the Dirac delta function. This relation holds in general for both discrete and continuous actions.

Now we return to our example. For simplicity, our agent has the discrete action space $\mathcal{A} = \{-1, 1\}$. We define the motor function $M(\mathbf{x}) = \text{sign}(\boldsymbol{v}^T\boldsymbol{x})$, where $\boldsymbol{v}^T = (-1, 1)$. This choice defines a linear classifier in activity (representation) space. Notice that our agent's policy is deterministic; we set its activity noise to zero, so it maps a state $s$ deterministically to an activity pattern, which in turn defines an action through $M$.

#### 3.2.3 REWARD-DEPENDENT NOISE FACILITATES POLICY OPTIMIZATION

The final and key component of NETO is the reward-dependent noise. We define the agent's neuromodulatory system so that it shapes the representation learning network's weight diffusion tensor according to an estimate of the policy reward $R$

$$\sigma_\theta = \sigma_\theta(R) \tag{8}$$

where the eigenvalues of $\sigma_\theta(R)$ are monotonically decreasing functions of $R$. Intuitively, this reward-dependent noise "cools" the system when the agent's policy is rewarding. More formally, it balances optimization of $\mathcal{L}$ and $R$ by changing the system's stochastically attractive points in weight space. It is not unreasonable to conceptualize $R$ as a regularizer on $\mathcal{L}$.

In our example agent, we define a modulatory system that estimates $R$ and controls the noise through $\eta_\theta = \eta_\theta(R)$, as the noise is proportional to $\sqrt{\eta_\theta}$ [3]. In particular, we let

[2] Time independence is an approximation that is not necessary but will simplify the discussion.

[3] Since all weights have the same learning rate, we implicitly restrict the diffusion tensor to a multiple of the identity here. This restriction likely hinders policy learning efficiency, as discussed in the Limitations section.

$$\eta_\theta(R) = 0.01e^{-\beta R}. \tag{9}$$

for $\beta > 0$. This choice of monotonically decreasing function is arbitrary.

While the focus of this work is to introduce NETO for policy optimization in representation learning networks, we feel it is important to note that the framework is actually more general. We discuss this generality in Appendix A.1.

# 4 RESULTS

## 4.1 A DIFFUSIVE SEARCH FACILITATES POLICY OPTIMIZATION WHEN $\mathcal{L}$ CAPTURES THE TASK-RELEVANT INFORMATION

An ideal representation learning objective captures enough information to solve a variety of tasks. We begin by analyzing the agent's learning process when this is this case. We construct a contextual bandit task that the agent can solve with the features that $\mathcal{L}$ encourages it to extract. We then show that the agent is guaranteed to maximize reward as time goes to infinity [4]. In the process, we illustrate that a diffusive search through the space of equivalent representations facilitates its policy learning.

### 4.1.1 THE CONTEXTUAL BANDIT TASK

The contextual bandit task is a special case of the Markov Decision Process where the state transitions are independent of the agent's action. Let our contextual bandit task have states $\boldsymbol{s} \in \mathbb{R}^{10}$. The states are drawn from two Gaussian clusters in 10 dimensions. In particular, we define them as $\boldsymbol{s} = k\boldsymbol{y}$, where $\boldsymbol{y} \sim \mathcal{N}(\boldsymbol{\mu}, \boldsymbol{C})$, and $k$ is a random sign. The largest eigenvalue of the covariance matrix $\boldsymbol{C}$ is 0.1, and the rest are 0.05. The mean vector $\boldsymbol{\mu}$ is the second eigenvector of $\boldsymbol{C}$ and has magnitude 2. We also define the agent's action space as $\mathcal{A} = \{1, -1\}$, and let $r(\boldsymbol{s}, a) = \mathbf{1}_{(\mathrm{a=k})}$. The agent must learn to take the action corresponding to the random sign. It is possible to learn this task because the Gaussian clusters are linearly separable. Figure 2 illustrates the setup.

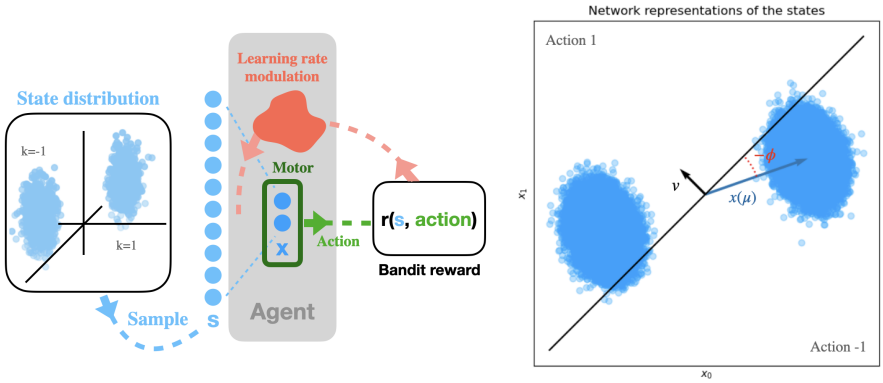


**Figure 2:** *(Left) Schematic of the agent in the contextual bandit task. The agent receives a state $\boldsymbol{s}$ (light blue), and its activities respond, resulting in an action. The neuromodulatory system receives the reward associated with this action and adjusts the weight learning rate. The weights then update according to $\mathcal{L}$, and the task continues. The agent's goal is to take the action corresponding to the random sign $k$. (Right) Illustration of all relevant variables in the network's representation space, where the axes denote the activity of its two neurons. Royal blue points show the network representations of the states $\boldsymbol{s}$, denoted $\boldsymbol{x}(\boldsymbol{s})$. The vector $\boldsymbol{v}$ defines a classifier that maps these representations to the action 1 or $-1$. Finally, $\phi$, the angle between $\boldsymbol{v}$ and $\boldsymbol{x}(\boldsymbol{\mu})$, tracks the rotational diffusion of the network responses through policy learning.*

[4] In practice, this occurs much more quickly.

#### 4.1.2 $\mathcal{L}$ CAPTURES THE TASK-RELEVANT FEATURE AND HAS A SOLUTION SPACE

We aim to show that the agent learns an optimal policy, even though the network that controls its actions only explicitly optimizes $\mathcal{L}$. We begin by demonstrating that (1) the network's objective captures the information necessary to solve the task, and (2) that this objective has a solution space: the network representations are free to rotate. We will use these properties to equate policy learning to a diffusive search through the solution space of $\mathcal{L}$. Finally, we will compute the limiting distribution of this search with reward-dependent noise. This process will show us that the agent preferentially resides in rewarding orientations.

Recall that $\mathcal{L}$ encourages the network to project the states $\boldsymbol{s}$ onto their principal subspace. We chose $\boldsymbol{\mu}$ so that its only nonzero component is in the principal subspace of $\boldsymbol{s}$. This means that the network needs to extract the features defined by $\mathcal{L}$ to solve the task. After it does so, its activities contain the information needed to pick the action associated with the random sign $k$. However, the agent does not necessarily use this information correctly; the stimulus-to-action mapping defined by $M$ is arbitrary.

The agent learns a rewarding policy because its representation learning network's weights diffuse through the solution space of $\mathcal{L}$. Representation learning objectives often have solution spaces because they are degenerate or the network is overparameterized. Here, the objective is degenerate. We begin by finding its solution space. This was done in previous work (Qin et al. (2023)), but we review it here because it is necessary for the rest of our discussion.

There are infinite ways to project the states $\boldsymbol{s}$ onto their principal subspace. In particular, any transformation of the form

$$\boldsymbol{W}_0 \to \boldsymbol{W}_0' = \boldsymbol{U}\boldsymbol{W}_0 \tag{10}$$

$$\boldsymbol{W}_1 \to \boldsymbol{W}_1' = \boldsymbol{U}\boldsymbol{W}_1\boldsymbol{U}^T \tag{11}$$

which yields

$$\boldsymbol{x} \to \boldsymbol{x}' = \boldsymbol{U}\boldsymbol{x} \tag{12}$$

where $\boldsymbol{U}^T\boldsymbol{U} = \boldsymbol{I}$ leaves $\mathcal{L}$ invariant. In other words, any rotation of the network representations is still a projection onto the principal subspace. Note that, to arrive at equation (12), we assumed that the activity dynamics (4) converge before a weight update.

We can now describe the solution space of $\mathcal{L}$ in terms of the network activities. Suppose that the network projects its inputs onto their principal subspace (such that its weights are in the solution space of $\mathcal{L}$). Let $\boldsymbol{x}(\boldsymbol{s})$ denote its response to the state $\boldsymbol{s}$. Then the space of all responses to this state that satisfy $\mathcal{L}$ are

$$\mathcal{H}_{\boldsymbol{s}} = \{\boldsymbol{U}\boldsymbol{x}(\boldsymbol{s}) : \boldsymbol{U} \in SO(2)\} \tag{13}$$

where $SO(2)$ is the special orthogonal group in two dimensions[5]. This space is a circle.

#### 4.1.3 WITH NOISE, THE AGENT EXPLORES THE SOLUTION SPACE OF $\mathcal{L}$

When the weight noise is reward-independent, the network representations converge to and explore their respective $\mathcal{H}_{\boldsymbol{s}}$. Random weight transformations of the form (10-11), with $\boldsymbol{U}$ as an infinitesimal rotation, generate this noisy exploration. The representations undergo rotational diffusion. We can describe the system's configuration in the solution space by tracking the evolution of $\phi$, the angle between $\boldsymbol{x}(\boldsymbol{\mu})$ and $\boldsymbol{v}$. Previous work used linear stability analysis to show that the diffusion coefficient characterizing this process is linear in $\eta_\theta$. So, its evolution is given by

[5]Technically, $\boldsymbol{U} \in O(2)$, but this extension is not important to our discussion because we will not consider reflections.

$$d\phi \propto \sqrt{\eta_\theta} dB \tag{14}$$

where we work up to proportionality because we are only interested in its limiting distribution. Since all representations transform together, this single parameter $\phi$ describes the network configuration during policy learning.

#### 4.1.4 REWARD-DEPENDENT NOISE BIASES THE NETWORK TOWARDS REWARDING POLICIES

Now that we understand the network's solution space exploration, we can analyze it with reward-dependent noise modulated by $\eta_\theta(R)$. In a general task with reward-dependent noise, we cannot guarantee that the network responses converge to $\mathcal{H}_{\boldsymbol{s}}$. Reward-dependent noise can stabilize points outside of the solution space. We will discuss this in greater detail later.

However, Figure 3 shows that the principal subspace error rapidly decreases and remains near optimal throughout this task. Here, all representations $\boldsymbol{x}(\boldsymbol{s})$ are approximately contained to their $\mathcal{H}_{\boldsymbol{s}}$. Intuitively, this occurs because these representations contain the information needed to solve the task. Reasonable policies exist within the solution space of $\mathcal{L}$. We can therefore analyze the reward-dependent rotational diffusion of the representation $\boldsymbol{x}(\boldsymbol{s})$ through $\mathcal{H}_{\boldsymbol{s}}$.

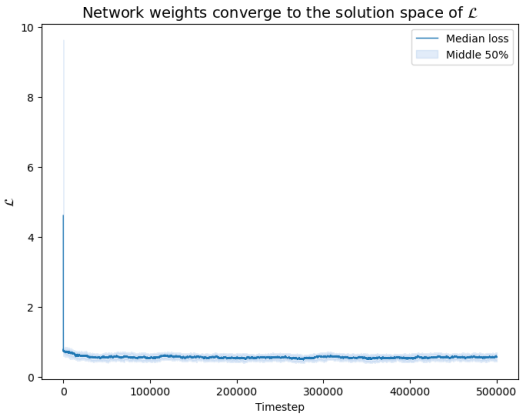


**Figure 3:** *The principal subspace error over time, quantified by* $\frac{\|\boldsymbol{F}^T\boldsymbol{F}-\boldsymbol{U}\boldsymbol{U}^T\|_F}{\|\boldsymbol{U}\boldsymbol{U}^T\|_F}$. *The columns of* $\boldsymbol{U}$ *are the top two eigenvectors of the covariance matrix* $\boldsymbol{C}$, *and* $\boldsymbol{F} = \boldsymbol{W}_1^{-1}\boldsymbol{W}_0$ *is the projection performed by the network after the convergence of (4). The operation* $\|.\|_F$ *denotes Frobenius norm.*

Reward-dependent rotational diffusion is equivalent to angle-dependent diffusion. This equivalence arises from the fact that $R$ is a function of $\phi$. One can see graphically in Figure 2, but as an example, notice that $M[\boldsymbol{x}(\boldsymbol{\mu}|\phi = \frac{\pi}{2})] = 1$, whereas $M[\boldsymbol{x}(\boldsymbol{\mu}|\phi = -\frac{\pi}{2})] = -1$. Different angles correspond to different policies.

If we find $\eta_\theta(\phi)$, we can express the rotational diffusion in a self-contained manner. We show in Appendix A.2 that $R(\phi)$ is approximately

$$R(\phi) \approx \Phi\left(\pm\frac{\|\boldsymbol{\mu}\|}{\sqrt{\lambda_1}}\tan\phi\right) \tag{15}$$

where $\Phi$ is the standard normal CDF, and $\lambda_1$ is the largest eigenvalue of $C$. We take the negative sign when $\phi \in [\frac{\pi}{2}, \frac{3\pi}{2}]$ and the positive sign otherwise. We can therefore write $\eta_\theta(\phi) = \eta_\theta(R(\phi))$, which is given by

$$\eta_\theta(\phi) \approx \exp\left[-\beta\Phi\left(\pm\frac{\|\boldsymbol{\mu}\|}{\sqrt{\lambda_1}}\tan\phi\right)\right] \tag{16}$$

Plugging (16) into (14) gives the equation describing the rotational diffusion of the network response to $\boldsymbol{\mu}$ through the agent's learning process. The same expression holds for all network responses. Under the Ito prescription, this is equivalent to the Fokker-Planck equation (up to proportionality)

$$\frac{\partial p(\phi,t)}{\partial t} \propto \frac{\partial^2}{\partial\phi^2}\left[\eta_\theta(\phi)p(\phi,t)\right]. \tag{17}$$

where $p(\phi,t)$ is the distribution of $\phi$. Its stationary distribution $p(\phi)$ is therefore

$$p(\phi) \propto \frac{1}{\eta_\theta(\phi)} \tag{18}$$

Finally, we can use $p(\phi)$ to find the expected reward received by an agent. In this limit, it is given by

$$\mathbb{E}[R] = \int_0^{2\pi} R(\phi)p(\phi)\, d\phi \tag{19}$$

which we compute numerically. In Figure 4, we see that as $\beta$ increases, $\mathbb{E}[R] \to 1$. With the appropriate choice of $\eta_\theta(R)$, the agent is guaranteed to learn a policy that maximizes reward.

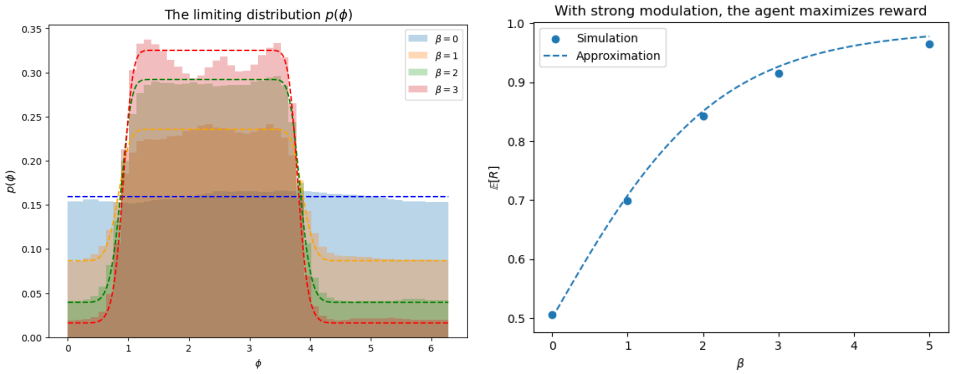


**Figure 4:** *(Left) The limiting distribution of $\phi$. The dashed lines denote our approximations for different $\beta$, and the histograms show the simulation data. With $\beta = 0$ (no noise modulation), $\phi$ is uniformly distributed in the solution space of $\mathcal{L}$. With larger $\beta$, the agent becomes biased towards rewarding angles. Data generated from 100 seeds of 500,000 timesteps. (Right) The expected reward received by the agent in the limit that time goes to infinity. The agent is guaranteed to maximize reward for large $\beta$ (stronger modulation). Our approximation agrees with simulations.*

Notice that policy optimization emerges by modulating a noisy, reward-independent objective. The agent first learns to extract normative features (the principal subspace projection). Then, it uses this information to optimize its policy (by exploring the informatically equivalent rotational configurations). In the next example, we will see that NETO is not restricted to this sequential learning: It can balance normative feature extraction with policy optimization by performing both simultaneously.

### 4.2 NETO BALANCES NORMATIVE REPRESENTATION AND POLICY LEARNING TO SOLVE TASKS

The contextual bandit task gives the impression that NETO agents learn representations and policies in separate phases. However, in general, NETO couples representation and policy learning. This coupling allows agents to balance normative and task-dependent representation learning, extracting features that $\mathcal{L}$ does not capture to improve task performance.

The contextual bandit example illustrates that we can consider the $R$-dependent noise as $\theta$-dependent. This equivalence arises because the same network learns representations and controls actions. Recent work in the context of stochastic gradient descent found that the weight diffusion tensor $\sigma_\theta(\theta)$ and the loss $\mathcal{L}(\theta)$ jointly determine the attractive points in weight configuration space (Chen et al. (2024)). This fact gives the network the freedom to extract information that is not optimal with respect to $\mathcal{L}$ if it helps the agent perform well in the task.

We demonstrate this principle with Cart Pole. Cart Pole is a classical control problem with a four-dimensional observation space. It is simple enough for our one-layer, linear network to solve. The goal of Cart Pole is to keep a pole upright on a cart while preventing it from falling off a frictionless track. The agent can push the cart right or left at each timestep. Importantly, the agent cannot learn an optimal policy for Cart Pole if its weights project the state observations onto their two-dimensional principal subspace. Assume for contradiction that the agent can find an optimal policy with this information. By definition, this policy minimizes the variance of the pole's deviation from its center. However, because this variance is small, the pole angle information is not in the principal

subspace. Without this information, the agent cannot determine which action to take. It therefore cannot have an optimal policy, a contradiction. Our agent cannot solve the task with only normative features. It must learn to adjust its representations so that they contain information that is suboptimal with respect to $\mathcal{L}$.

The NETO framework is most likely to extract these task-relevant features if they exist near the solution space of $\mathcal{L}$. So, we scale the Cart Pole observation space so that the information needed to learn an optimal policy exists near this region. Even with this normalization, solving the task requires the agent to extract features resulting in representation learning losses that are orders of magnitude larger than in the contextual bandit task, as seen in Figure 5 (left).

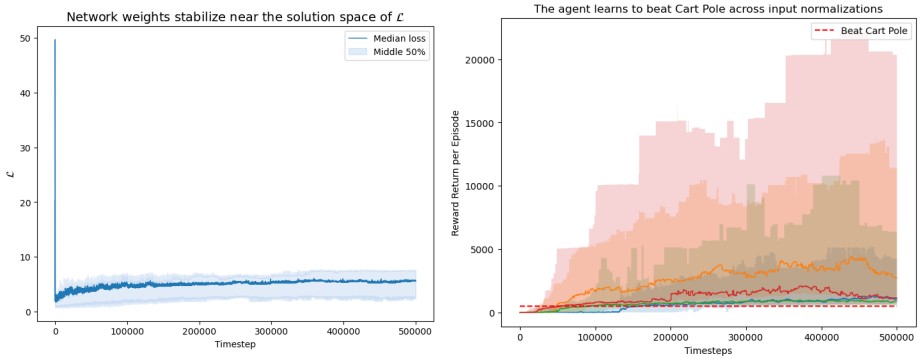


**Figure 5:** *(Left) The principal subspace loss over time. The loss decreases rapidly through initial feature learning but settles an order of magnitude larger than in the contextual bandit task. Reward-dependent noise stabilized the weights in a suboptimal configuration to extract task-relevant features. (Right) The agent learns to beat Cart Pole across a range of input normalizations. Median and middle 50% percentile over 100 seeds plotted through 500,000 timesteps of continuous interaction. Different colors correspond to different normalizations, listed in Appendix A.3.*

Remarkably, the agent can solve the task across a range of input normalizations, as seen in Figure 5 (right). Since the agent's weights stabilize at a larger representation learning loss, it learns to extract features not specified by $\mathcal{L}$. Note that these features are not arbitrary. They are still partially determined by $\mathcal{L}$; the loss settles to be many orders of magnitude lower than where it began.

This simple example illustrates a general principle: NETO agents balance normative and task-relevant representation learning to facilitate policy optimization. Policy learning acts as a regularizer on weights that are otherwise selected to extract normative features defined by $\mathcal{L}$. Indeed, representation and policy learning are interdependent, emerging together from the same system.

## 5 Discussion

In this article, we present the hypothesis that policy optimization emerges from the noisy learning of representation objectives in nervous systems. We explore the consequences of this idea in a linear, one-layer network. Through simulations and analytical arguments, we show that this network learns the optimal policy in tasks that it can reasonably be expected to solve. Each task serves to provide intuition for its learning process. When $\mathcal{L}$ captures all task-relevant features, the network performs a diffusive search through its solution space to discover an optimal policy. The agent can also modify its representations to capture task-relevant features, even if the resulting representations are slightly suboptimal with respect to $\mathcal{L}$.

### 5.1 Consistency with experimental observations

While the NETO framework is more general, we focus on the case where $\mathcal{L}$ is a representation learning objective and $R$ denotes reward. This choice gives NETO the potential to explain several recently observed phenomena describing neural representation changes during task learning. We discuss three examples: representation drift, neural reassociation, and the orthogonalization of task-relevant features.

Representation drift describes the slow change in the neural codes of stimuli over time. Crucially, these changes maintain the information encoded in the neural population (Driscoll et al. (2017); Rubin et al. (2015); Ziv et al. (2013)). Representation drift emerges naturally from the NETO framework when $\mathcal{L}$ is a representation learning objective. We saw representation drift clearly in the contextual bandit task: the network responses rotated while maintaining their population structure. This result has been noted before in (Qin et al. (2023)). NETO suggests a potential feature of this drift: It can aid task learning by sampling behavioral policies, even when behavioral performance stabilizes.

In neural reassociation, the brain forms new stimulus-action associations by *reassociating* stimulus-code pairs instead of generating new activity patterns. Researchers observed neural reassociation in primate motor cortex during task learning (Golub et al. (2018)). NETO can reproduce neural reassociation with the correct motor function. Consider a nonlinear motor function $M(\boldsymbol{x})$ from some unspecified brain region to motor cortex (instead of directly to an action). Let the function be nonlinear so that changing the distribution of inputs does not change the range of $M(\boldsymbol{x})$, which we can think of an as a manifold in activity space. Suppose NETO operates only at the neurons in the domain of $M$. Then, task learning will occur by changing the structure of the neural activity in its domain to re-map stimuli to different but already existing patterns in motor cortex. If we were to record the neural activity in motor cortex during task learning, we would not observe significant changes in the manifold of response patterns. Instead, we would see the associations between these stimuli and the existing patterns shift. NETO can therefore explain neural reassociation.

Nervous systems have also been observed to orthogonalize task-relevent features to facilitate task learning (Failor et al. (2021); Gurnani & Cayco Gajic (2023)). NETO can orthogonalize task-relevant features by exploring the solution space of a nonlinear representation learning objective. As a simple example, consider the non-negative similarity matching (NNSM) objective introduced in (Pehlevan & Chklovskii (2014)). In (Sengupta et al. (2018)), the authors found that networks operating under the NNSM objective learn localized receptive fields. Moreover, the locations of these receptive fields drift under noisy weight updates (Qin et al. (2023)). Drifting receptive fields is an example of feature orthogonalization: Two stimuli can go from sharing a receptive field (large neural code overlap) to existing in different receptive fields (low overlap). This strongly suggests that a NETO network with $\mathcal{L}$ as the NNSM objective would orthogonalize task-relevant features if this improved its ability to seek reward. The observation that NETO may reproduce a number of experimental observations supports it as a biologically plausible theory for emergent policy learning.

### 5.2 Limitations & Future Directions

NETO's primary limitation is that it relies on diffusive searches, which are notoriously inefficient in high dimensions. Consider scaling the agent's representation space to $N$ dimensions. Then suppose that it interacts in a contextual bandit task, this time with $N$ actions and $N$ input clusters. The time that it takes the network to orient these clusters with a diffusive search scales exponentially in $N$.

There are two potential solutions to this issue. The first is a biological argument that downplays it, and the second presents an avenue to address the problem. Biological neural codes tend to be low dimensional (Iyer et al. (2022); Shine et al. (2019); Ebitz & Hayden (2021)). If the diffusive search is largely restricted to this low-dimensional subspace, then it avoids the exponential scaling that comes with an increasingly high-dimensional representation space. This argument, while plausible, avoids the issue. To solve it, we need to construct a more efficient modulatory system. In all our experiments, we assume that the modulatory system acts uniformly on all neurons in the network. This unnecessary assumption restricts the weight diffusion tensor to a multiple of the identity. Neuromodulation in natural systems is not uniform, which may allow it to shape more complex diffusion tensors. It is possible that carefully shaping the diffusion tensor could make policy learning more efficient. This becomes clear when we discretize time into larger episodes, and consider the policy in each episode as a "sample" of the reward function $R$ at a point in weight space. Under this picture, we see that a modulatory system following an evolutionary algorithm like Covariance Matrix Adaptation may be more efficient.

Animals learn tasks quickly by applying previously acquired knowledge to new problems. They achieve this rapid, generalizable learning by leveraging their world model: a powerful predictive representation of the environment and how it evolves under their actions. Future work could also

explore whether agents with expressive representation learning objectives that build world models can solve challenging tasks. It would also be interesting to explore other applications of the NETO framework: for instance, online supervised learning.

### 5.3 Conclusion

Though we show that a one-layer network can solve basic tasks, the principal contribution of this work is to introduce NETO as a theory for emergent policy learning in nervous systems. This result suggests further exploration into reward-dependent noise in the nervous system.

## References

Joy Alcedo and Veena Prahlad. Neuromodulators: an essential part of survival. *Journal of Neurogenetics*, 34(3-4):475–481, 2020.

Guo-qiang Bi and Mu-ming Poo. Synaptic modifications in cultured hippocampal neurons: dependence on spike timing, synaptic strength, and postsynaptic cell type. *Journal of Neuroscience*, 18 (24):10464–10472, 1998.

Cristiano Capone and Pier Stanislao Paolucci. Towards biologically plausible model-based reinforcement learning in recurrent spiking networks by dreaming new experiences. *Scientific Reports*, 14(1):14656, 2024.

Feng Chen et al. Stochastic collapse: How gradient noise attracts sgd dynamics towards simpler subnetworks. *Advances in Neural Information Processing Systems*, 36, 2024.

Luke T. Coddington, Sarah E. Lindo, and Joshua T. Dudman. Mesolimbic dopamine adapts the rate of learning from action. *Nature*, 614(7947):294–302, 2023.

Yang Dan and Mu-ming Poo. Spike timing-dependent plasticity of neural circuits. *Neuron*, 44(1): 23–30, 2004.

Ingmar EJ de Vries and Moritz F. Wurm. Predictive neural representations of naturalistic dynamic input. *Nature Communications*, 14(1):3858, 2023.

Laura N. Driscoll et al. Dynamic reorganization of neuronal activity patterns in parietal cortex. *Cell*, 170(5):986–999, 2017.

R. Becket Ebitz and Benjamin Y. Hayden. The population doctrine revolution in cognitive neurophysiology. *Neuron*, 109:3055–3068, 2021.

Samuel W. Failor, Matteo Carandini, and Kenneth D. Harris. Visuomotor association orthogonalizes visual cortical population codes. *BioRxiv*, pp. 2021–05, 2021.

Răzvan V. Florian. Reinforcement learning through modulation of spike-timing-dependent synaptic plasticity. *Neural Computation*, 19(6):1468–1502, 2007.

Matthew D. Golub et al. Learning by neural reassociation. *Nature Neuroscience*, 21(4):607–616, 2018.

Harsha Gurnani and N. Alex Cayco Gajic. Signatures of task learning in neural representations. *Current Opinion in Neurobiology*, 83:102759, 2023.

David Ha and Jürgen Schmidhuber. Recurrent world models facilitate policy evolution. *Advances in Neural Information Processing Systems*, 31, 2018.

Manu Srinath Halvagal and Friedemann Zenke. The combination of hebbian and predictive plasticity learns invariant object representations in deep sensory networks. *Nature Neuroscience*, 26(11), 2023.

Kartik K. Iyer et al. Focal neural perturbations reshape low-dimensional trajectories of brain activity supporting cognitive performance. *Nature Communications*, 13(1):4, 2022.

Max Jaderberg et al. Reinforcement learning with unsupervised auxiliary tasks. *arXiv preprint arXiv:1611.05397*, 2016.

David Kappel et al. Reward-based stochastic self-configuration of neural circuits. *arXiv preprint arXiv:1704.04238*, pp. 1162–1167, 2017.

Sven Kroener et al. Dopamine modulates persistent synaptic activity and enhances the signal-to-noise ratio in the prefrontal cortex. *PloS One*, 4(8):e6507, 2009.

Robert Legenstein, Dejan Pecevski, and Wolfgang Maass. A learning theory for reward-modulated spike-timing-dependent plasticity with application to biofeedback. *PLoS Computational Biology*, 4(10):e1000180, 2008.

Chenguang Li et al. Neuron-level prediction and noise can implement flexible reward-seeking behavior, 2024. bioRxiv: 2024-05.

Nuo Li and James J. DiCarlo. Unsupervised natural experience rapidly alters invariant object representation in visual cortex. *Science*, 321:1502–1507, 2008.

David Lipshutz et al. Normative framework for deriving neural networks with multicompartmental neurons and non-hebbian plasticity. *PRX Life*, 1(1):013008, 2023.

Wolfgang Maass. Noise as a resource for computation and learning in networks of spiking neurons. *Proceedings of the IEEE*, 102(5):860–880, 2014.

Henry Markram et al. Regulation of synaptic efficacy by coincidence of postsynaptic aps and epsps. *Science*, 275(5297):213–215, 1997.

Beren Millidge et al. Predictive coding networks for temporal prediction. *PLOS Computational Biology*, 20(4):e1011183, 2024.

Cengiz Pehlevan and Dmitri B. Chklovskii. A hebbian/anti-hebbian network derived from online non-negative matrix factorization can cluster and discover sparse features. In *2014 48th Asilomar Conference on Signals, Systems and Computers*. IEEE, 2014.

Cengiz Pehlevan, Tao Hu, and Dmitri B. Chklovskii. A hebbian/anti-hebbian neural network for linear subspace learning: A derivation from multidimensional scaling of streaming data. *Neural Computation*, 27(7):1461–1495, 2015.

Cengiz Pehlevan, Anirvan M. Sengupta, and Dmitri B. Chklovskii. Why do similarity matching objectives lead to hebbian/anti-hebbian networks? *Neural Computation*, 30(1), 2017.

Jianhua Qin and Aaron R. Wheeler. Maze exploration and learning in c. elegans. *Lab on a Chip*, 7 (2):186–192, 2007.

Shanshan Qin et al. Coordinated drift of receptive fields in hebbian/anti-hebbian network models during noisy representation learning. *Nature Neuroscience*, 26(2), 2023.

Rajkumar Vasudeva Raju et al. Space is a latent sequence: A theory of the hippocampus. *Science Advances*, 10(31):eadm8470, 2024.

Alon Rubin et al. Hippocampal ensemble dynamics timestamp events in long-term memory. *eLife*, 4:e12247, 2015.

Anirvan Sengupta et al. Manifold-tiling localized receptive fields are optimal in similarity-preserving neural networks. volume 31, 2018.

James M. Shine et al. Human cognition involves the dynamic integration of neural activity and neuromodulatory systems. *Nature Neuroscience*, 22:289–296, 2019.

Mufeng Tang, Helen Barron, and Rafal Bogacz. Sequential memory with temporal predictive coding. *Advances in Neural Information Processing Systems*, 36, 2024.

Johannes H. van Hateren. A theory of maximizing sensory information. *Biological Cybernetics*, 68 (1):23–29, 1992.

Caitlin M. Vander Weele et al. Dopamine enhances signal-to-noise ratio in cortical-brainstem encoding of aversive stimuli. *Nature*, 563(7731):397–401, 2018.

Georg Winterer and Daniel R. Weinberger. Genes, dopamine and cortical signal-to-noise ratio in schizophrenia. *Trends in Neurosciences*, 27(11):683–690, 2004.

Yaniv Ziv et al. Long-term dynamics of ca1 hippocampal place codes. *Nature Neuroscience*, 16(3): 264–266, 2013.

# A APPENDIX

## A.1 NEURAL THERMAL OPTIMIZATION AS A MORE GENERAL FRAMEWORK

While the focus of this work is to introduce NETO for policy optimization, we feel that it is important to briefly introduce the framework in a more general setting. NETO is a framework for how the nervous system can satisfy two distinct objectives when the primary network's dynamics are governed by only one. Fundamentally, NETO states that:

1. Nervous systems are noisy,
2. A self-supervised objective $\mathcal{L}$ determines the network's deterministic weight dynamics,
3. A neuromodulatory system shapes the noise given the value of $R$.

They key insight behind NETO is that if the eigenvalues of the weight diffusion tensor are monotonically decreasing functions of $R$, the attractive points of the system change to balance the optimization of $\mathcal{L}$ and $R$. The only requirement is that $\mathcal{L}$ is differentiable with respect to the network weights. Nicely, $R$ does not have to satisfy this requirement. In a sense, NETO combines a gradient descent algorithm to optimize $\mathcal{L}$ with an evolutionary algorithm to optimize $R$. The evolutionary algorithm comes into play in the design of the weight diffusion tensor.

NETO's generality allows future work to explore other choices of $\mathcal{L}$ or $R$. For instance, if $\mathcal{L}$ were powerful enough to build an expressive world model, then the agent could likely solve challenging reinforcement learning problems. Moreover, if $R$ were a supervised learning loss instead of reward, the agent could learn a classification task online. There are a wealth of different objective combinations that one could consider.

We also note that one could further generalize this framework to any model defined by the parameters $\theta$ that aims to satisfy two objectives $\mathcal{L}$ and $R$, though only $\mathcal{L}$ is differentiable with respect to the model parameters. This framework may be useful if one believes that satisfying $\mathcal{L}$ will help the model approach optimal solutions with respect to $R$, thereby increasing the efficiency of an evolutionary algorithm.

## A.2 APPROXIMATION OF $R(\phi)$

In this section, we approximate $R(\phi)$ as a function of the contextual bandit state distribution. This calculation is necessary to formulate the agent's learning process as angle-dependent rotational diffusion. Our strategy will be as follows:

1. We will write $a_\phi(\boldsymbol{s})$, the agent's action under an observation $\boldsymbol{s}$ as a function of the angle $\phi$.
2. We will write $R(\phi)$ as an expectation value and eventually an integral.
3. We will see that evaluating this expression requires finding the distribution of the network responses $\boldsymbol{x}(\boldsymbol{s})$. To find this distribution, we compute the principal subspace of $\boldsymbol{s}$.
4. We then use this information to write the distribution of $\boldsymbol{x}$ as a function of $\phi$.
5. Finally, we approximate the integral in step (2).

### A.2.1 STEP 1: FINDING $a_\phi(\mathbf{s})$

We begin by finding $a_\phi(\boldsymbol{s})$. As in the main text, we assume that the agent's parameters are in the solution space of $\mathcal{L}$. It follows that its activity dynamics (4) that its parameters $\theta$ define a mapping from the observation space to the network's representation space given by

$$x(\boldsymbol{s}|\phi) = \boldsymbol{U}(\phi)\boldsymbol{P}\boldsymbol{s} \tag{20}$$

where the operator $P$ projects $\boldsymbol{s}$ into its 2-dimensional principal subspace, and $\boldsymbol{U}(\phi) \in \mathbb{R}^{2\times 2}$ rotates the result by $\phi$ radians. By convention, we choose $\boldsymbol{P}$ such that $\boldsymbol{x}(k\boldsymbol{\mu}|\phi = 0) = ck\boldsymbol{v}$ for some $c > 0$.

Since $M(\boldsymbol{x}) = \text{sign}(\boldsymbol{v}^T\boldsymbol{x})$, with $\boldsymbol{v}^T = \frac{1}{\sqrt{2}}(-1, 1)$, this corresponds to the action

$$a_\phi(\boldsymbol{s}) = \text{sign}(\boldsymbol{v}^T\boldsymbol{U}(\phi)\boldsymbol{P}\boldsymbol{s}) \tag{21}$$

### A.2.2 STEP 2: WRITE $R(\phi)$ AS AN INTEGRAL.

Recall that the goal of the agent is to pick the action $k$, the random sign. Then the reward at a single step, $r(\boldsymbol{s}, a)$, is given by

$$r(\boldsymbol{s}, a_\phi(\boldsymbol{s})) = \mathbb{I}\left[a_\phi(\boldsymbol{s}) = k\right] \tag{22}$$

Since the distribution of $\boldsymbol{s}$ is stationary and the agent's policy is fixed, the expected reward $R(\phi) = \mathbb{E}\left[\sum_{t=1}^{T} r(\boldsymbol{s}_t, a_\phi(\boldsymbol{s}_t))\right] = \mathbb{E}\left[r(\boldsymbol{s}, a_\phi(\boldsymbol{s}))\right]$. So, we can write $R(\phi)$ as

$$R(\phi) = \mathbb{E}\left(\mathbb{I}\left[\text{sign}\left(\boldsymbol{v}^T \boldsymbol{U}(\phi) P \boldsymbol{s}\right) = k\right]\right) \tag{23}$$

$$= P\left[\text{sign}\left(\boldsymbol{v}^T \boldsymbol{U}(\phi) \boldsymbol{P} \boldsymbol{s}\right) = k\right] \tag{24}$$

Let $f_{\boldsymbol{x}}(\boldsymbol{x}_1, \boldsymbol{x}_2 | k, \phi)$ be the distribution of the network representations $\boldsymbol{x}$ conditional on the random sign $k$ and given $\phi$. Then, considering the meaning of (24) geometrically, we see that is it equivalent to

$$R(\phi) = \int_{\boldsymbol{x}_1}^{\infty} \int_{-\infty}^{\infty} f_{\boldsymbol{x}}(\boldsymbol{x}_1, \boldsymbol{x}_2 | k = 1, \phi)\, d\boldsymbol{x}_1\, d\boldsymbol{x}_2 \tag{25}$$

We aim to evaluate this integral.

### A.2.3 STEP 3: FINDING THE PRINCIPAL SUBSPACE OF $\mathbf{s}$

To evaluate the integral (25), we must find the distribution of $\boldsymbol{x} = U(\phi) P \boldsymbol{s}$. Of course, to find this distribution we must first determine $P$. So, we need to find the principal subspace of $\mathbf{s}$. Recall that we can decompose $\mathbf{s} = k(\sqrt{\boldsymbol{C}}\mathbf{z} + \boldsymbol{\mu})$, where $\mathbf{z}$ is a standard normal random variable in 10 dimensions, $\boldsymbol{C}$ is the covariance matrix, $\boldsymbol{\mu}$ is the mean, and k is a random sign. Since $\mathbf{s}$ is centered (due to the random sign), its covariance matrix is given by

$$\mathbb{E}\left[\mathbf{s}\mathbf{s}^T\right] = \mathbb{E}\left[k^2(\sqrt{\boldsymbol{C}}\mathbf{z} + \boldsymbol{\mu})(\sqrt{\boldsymbol{C}}\mathbf{z} + \boldsymbol{\mu})^T\right] \tag{26}$$

$$= \boldsymbol{\mu}\boldsymbol{\mu}^T + \mathbb{E}\left[\sqrt{\boldsymbol{C}}\mathbf{z}\mathbf{z}^T\sqrt{\boldsymbol{C}}^T\right] \tag{27}$$

where we used the fact that $\mathbb{E}\left[\mathbf{z}\right] = 0$. Then, using $\mathbb{E}\left[\mathbf{z}\mathbf{z}^T\right] = \boldsymbol{I}$, and since $\boldsymbol{C}$ is symmetric, $\sqrt{\boldsymbol{C}}\sqrt{\boldsymbol{C}}^T = \sqrt{\boldsymbol{C}}\sqrt{\boldsymbol{C}^T} = \sqrt{\boldsymbol{C}^2} = \boldsymbol{C}$, we find

$$\mathbb{E}\left[\mathbf{s}\mathbf{s}^T\right] = \boldsymbol{\mu}\boldsymbol{\mu}^T + \boldsymbol{C} \tag{28}$$

In the contextual bandit task, we defined $\boldsymbol{C}$ such that $\boldsymbol{\mu}$ is an eigenvector. Let $\lambda_\mu$ be the eigenvalue associated with $\boldsymbol{\mu}$, and $\{\lambda_i\}_{i=1}^{9}$ denote the other nine eigenvalues with associated eigenvectors $\{\boldsymbol{u}^{(i)}\}_{i=1}^{9}$. Then the eigenvalues of $\mathbb{E}\left[\mathbf{s}\mathbf{s}^T\right]$ are given by $\{\|\boldsymbol{\mu}\|^2 + \lambda_\mu, \lambda_1, ..., \lambda_9\}$.

The mean vector $\boldsymbol{\mu}$ allows the agent to identify the random sign $k$. So, to ensure that there exists a weight configuration in the solution space of $\mathcal{L}$ that solves the task, we needed the projection of $\boldsymbol{\mu}$ onto the principal subspace of $\mathbf{s}$ to be nonzero. We therefore chose $\|\boldsymbol{\mu}\|^2 + \lambda_\mu > \lambda_1 > ... \geq \lambda_9$.

We see that the principal subspace of $\mathbf{s}$ is given by $\text{span}(\boldsymbol{\mu}, \boldsymbol{u}^{(1)})$. We can therefore define $\boldsymbol{P}$ as the projection onto this subspace with $P\boldsymbol{\mu} = \|\boldsymbol{\mu}\|\boldsymbol{v}$ and $\boldsymbol{P}\boldsymbol{u}^{(1)} = \frac{1}{\sqrt{2}}(-1, -1)^T$.

### A.2.4 STEP 4: FINDING THE DISTRIBUTION $f_{\mathbf{x}}(\mathbf{x}_1, \mathbf{x}_2 | k = 1, \phi)$

Now that we have determined the projection operator $\boldsymbol{P}$, we can find the distribution of $\boldsymbol{x}$ given the random sign $k = 1$. Notice that, conditioned on $k$, the distribution of $\boldsymbol{s}$ is Multivariate Normal. Any

projection or rotation of a Multivariate Normal is still Multivariate Normal. Nicely, this means that the distribution of $\boldsymbol{x}(s) = \boldsymbol{U}(\phi)\boldsymbol{P}\boldsymbol{s}$ is defined by its mean and covariance matrix. We need only compute these.

Recall that $\mathbf{s} = k(\sqrt{\boldsymbol{C}}\mathbf{z} + \boldsymbol{\mu})$, so

$$\mathbb{E}\left[\boldsymbol{x}|k=1\right] = \mathbb{E}\left[k\boldsymbol{U}(\phi)\boldsymbol{P}(\sqrt{\boldsymbol{C}}\mathbf{z} + \boldsymbol{\mu})|k=1\right] \tag{29}$$

$$= \mathbb{E}\left[\boldsymbol{U}(\phi)\boldsymbol{P}\sqrt{\boldsymbol{C}}\mathbf{z}|k=1\right] + \mathbb{E}\left[\boldsymbol{U}(\phi)\boldsymbol{P}\boldsymbol{\mu}|k=1\right] \tag{30}$$

$$= \boldsymbol{U}(\phi)\|\boldsymbol{\mu}\|\boldsymbol{v} \tag{31}$$

where, in the last line, we used $P\boldsymbol{\mu} = \|\boldsymbol{\mu}\|\boldsymbol{v}$ and $\mathbb{E}\left[\mathbf{z}\right] = 0$. Before we evaluate the covariance matrix, we make the following simplification. Let $\boldsymbol{\mu}' := \|\boldsymbol{\mu}\|\boldsymbol{v}$. Notice that

$$\boldsymbol{x}|(k=1) - \mathbb{E}[\boldsymbol{x}|k=1] = \boldsymbol{U}(\phi)\boldsymbol{P}(\sqrt{\boldsymbol{C}}\mathbf{z} + \boldsymbol{\mu}) - \boldsymbol{U}(\phi)\boldsymbol{\mu}' \tag{32}$$

$$= \boldsymbol{U}(\phi)\boldsymbol{P}\sqrt{\boldsymbol{C}}\mathbf{z} + \boldsymbol{U}(\phi)\boldsymbol{P}\boldsymbol{\mu} - \boldsymbol{U}(\phi)\boldsymbol{\mu}' \tag{33}$$

$$= \boldsymbol{U}(\phi)\boldsymbol{P}\sqrt{\boldsymbol{C}}\mathbf{z} \tag{34}$$

We can now evaluate the covariance matrix, which we denote $\boldsymbol{C}'$.

$$\boldsymbol{C}' = \mathbb{E}\left[(\boldsymbol{U}(\phi)\boldsymbol{P}\sqrt{\boldsymbol{C}}\mathbf{z})(\boldsymbol{U}(\phi)\boldsymbol{P}\sqrt{\boldsymbol{C}}\mathbf{z})^T\right] \tag{35}$$

$$= \boldsymbol{U}(\phi)\boldsymbol{P}\sqrt{\boldsymbol{C}}\,\mathbb{E}\left[\mathbf{z}\mathbf{z}^T\right]\sqrt{\boldsymbol{C}}^T\boldsymbol{P}^T\boldsymbol{U}(\phi)^T \tag{36}$$

$$= \boldsymbol{U}(\phi)\boldsymbol{P}\sqrt{\boldsymbol{C}}\sqrt{\boldsymbol{C}}^T\boldsymbol{P}^T\boldsymbol{U}(\phi)^T \tag{37}$$

$$= \boldsymbol{U}(\phi)\boldsymbol{P}\boldsymbol{C}\boldsymbol{P}^T\boldsymbol{U}(\phi)^T \tag{38}$$

$$= \boldsymbol{U}(\phi)\boldsymbol{D}\boldsymbol{U}(\phi)^T \tag{39}$$

where $\boldsymbol{D} = \mathrm{diag}(\lambda_\mu, \lambda_1)$. We made the last step by the definition of the projection operator $\boldsymbol{P}$.

This means that

$$\mathbf{x}|(k=1) \sim \mathcal{N}(\boldsymbol{U}(\phi)\boldsymbol{\mu}', \boldsymbol{U}(\phi)\boldsymbol{D}\boldsymbol{U}(\phi)^T) \tag{40}$$

We can now attempt to evaluate the integral (25).

#### A.2.5 Step 5: Approximating the integral for $R(\phi)$

We now see that

$$R(\phi) = \int_{\boldsymbol{x}_1}^{\infty}\int_{-\infty}^{\infty}\mathcal{N}(\boldsymbol{x};\,\boldsymbol{U}(\phi)\boldsymbol{\mu}', \boldsymbol{U}(\phi)\boldsymbol{D}\boldsymbol{U}(\phi)^T)\,d\boldsymbol{x}_1 d\boldsymbol{x}_2 \tag{41}$$

This integral is not analytically tractable because the Gaussian is not centered and has a potentially non-diagonal covariance matrix. However, we can make an approximation that will be valid for our contextual bandit task.

Notice that we defined the contextual bandit state distribution such that $\lambda_\mu << \|\boldsymbol{\mu}\|^2, \lambda_1$. We did this so that the agent can learn to solve the task consistently (the two Gaussian clusters need to be linearly distinguishable). We therefore approximate the expression with the limit that $\lambda_\mu \to 0$.

For geometric intuition, refer to Figure 6. We see that, in this limit, the integral would be exactly given by

$$R(\phi) = \int_{-\infty}^{\|\boldsymbol{\mu}'\| \tan(\phi)} \mathcal{N}(x; 0, \lambda_1)\, dx \tag{42}$$

$$= \Phi\left(\pm \frac{\|\boldsymbol{\mu}\|}{\sqrt{\lambda_1}} \tan(\phi)\right) \tag{43}$$

since $\|\boldsymbol{\mu}'\| = \|\boldsymbol{\mu}\|$, where we take the negative sign when $\phi \in [\frac{\pi}{2}, \frac{3\pi}{2}]$ and the positive sign otherwise.

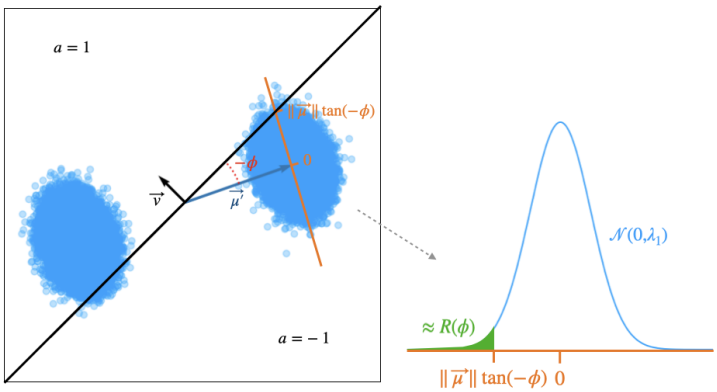


**Figure 6:** *A visualization of the $R(\phi)$ approximation. $R(\phi)$ is equivalent to the probability that $\boldsymbol{x}(\boldsymbol{s}|\phi, k = 1)$ corresponds to $a = 1$. Visually, this corresponds to the probability that the rightmost cluster is in the upper region of activity space. In the limit that $\lambda_\mu \to 0$, all variance perpendicular to the orange line disappears. So this quantity becomes the integral of the distribution $\mathcal{N}(0, \lambda_1)$ from $-\infty$ to $|\boldsymbol{\mu}\| \tan(\phi)$.*

### A.2.6 Our approximation agrees with simulations

In Figure 7, we verify our approximation by comparing it to simulations. We find good agreement when our assumption that $\lambda_\mu << \|\boldsymbol{\mu}\|^2, \lambda_1$ holds. To get simulation values of $R(\phi)$, we define the mapping $a_\phi(\boldsymbol{s})$. We then draw the states $\boldsymbol{s}$ according to the process defined in the main text and compute the expected value of the indicator that $a_\phi(\boldsymbol{s}) = k$, where $k$ is the random sign.

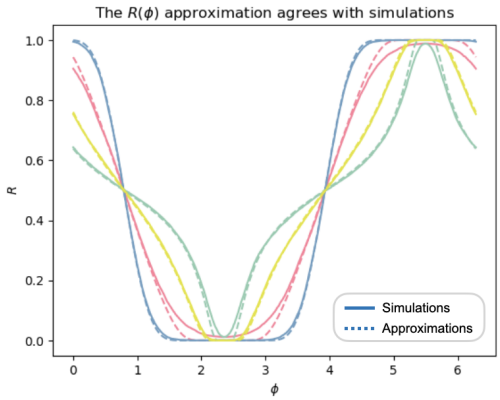


**Figure 7:** *The $R(\phi)$ approximation is reasonable across a range of examples where our assumption $\lambda_\mu << \|\mu\|^2, \lambda_1$ holds. Solid lines indicate simulation values, and dashed lines denote the analytical approximation. In these simulations, $\|\mu\|^2 \in \{0.25, 1\}$, $\lambda_1 \in \{0.1, 2\}$, and $\lambda_\mu = 0.05$.*

**Table 1:** *Agent Parameters*

| **Task** | $\beta$ | $\sigma_\theta^2$ | $\gamma$ | Input Normalizations |
|---|---|---|---|---|
| Contextual Bandit | $\{0, 1, 2, 3, 5\}$ | $0.001$ | 1 | N/A |
| Cart Pole | $\frac{1}{30}$ | 1 | 1 | Norm. 1: $(20, 0.5, 0.1, 1)$ |
| | | | | Norm. 2: $(20, 20, 0.1, 1)$ |
| | | | | Norm. 3: $(1, 1, 0.1, 1)$ |
| | | | | Norm. 4: $(0.5, 20., 0.1, 1)$ |

### A.3 Agent parameters in the two tasks

Table 1 provides the parameters that we used to run the agent simulations in the contextual bandit and Cart Pole tasks. The agent requires greater noise variance to learn features that its representation learning objective does not capture, as seen in the Cart Pole task. However, due to the large variance, we did not add noise to $\boldsymbol{W}_1$. We made this choice because the SM objective requires that $\boldsymbol{W}_1$ remain symmetric. Alternatively, we could have chosen to symmetrize this weight noise.

Also, note that the "Input Normalizations" column includes the scaling factors in the order (cart position, cart velocity, pole position, pole angular velocity). Because the range of the pole position values are an order of magnitude smaller than the other observations, we always scale this up by a factor of 10. From top to bottom, these normalizations correspond to the blue, orange, green, and red lines in Figure 5, respectively.

### A.4 The modulatory system shows promise for agreement with experiment

In our example agent, the neuromodulatory system controls the weight learning rate as a function of reward, thereby tuning the noise. Dopamine, a neuromodulator associated with reward signals, has been observed to adapt the weight learning rate in mice during policy optimization (Coddington et al. (2023)). Evidence also points to dopamine increasing the signal-to-noise ratio in neural activities (Vander Weele et al. (2018); Winterer & Weinberger (2004); Kroener et al. (2009)). This, in many ways, is equivalent to decreasing weight noise because activity responses drive weight changes. So, if neural activities are less noisy, their weight dynamics will be less noisy too. Therefore, NETO postulates a modulatory system that shows promise for consistency with features of biological neuromodulatory systems.

### A.5 Implementation of the modulatory system $R$ estimates

In this article, we often referred to a modulatory system that estimates $R$. However, we did not mention how this system implements this policy reward estimation.

#### A.5.1 The contextual bandit

In the contextual bandit task, the modulatory system estimates the reward $R$ with an exponential filter of length $T = 100$ timesteps. It updates its reward estimate at time $t$, $\hat{R}_t$, with the equation

$$\hat{R}_{t+1} = (1 - \frac{1}{T})\hat{R}_t + \frac{1}{T} r(s_t, a_t) \tag{44}$$

where $r(s_t, a_t)$ gives the reward received in that timestep.

#### A.5.2 Cart Pole

In Cart Pole, the network uses the last trial result as the $R$ estimate. This is equivalent to the contextual bandit implementation if we consider the exponential filter to be over trials instead of timesteps, and we set $T = 1$.