Towards a Predictive Coding Based Model of Simple Biological Intelligence

Jonah Brenner & Adam Boesky

Link to GitHub repository.

Introduction

A fundamental but unsolved problem in computational neuroscience is explaining simple biological intelligence. To our knowledge, there is currently no model that exhibits the basic properties of even the most simple nervous systems. Such a model should (Li et al., 2023):

1. Seek and exploit different types of reward
2. Balance exploratory and exploitative states
3. Learn associations
4. Adjust readily to changes in the environment
5. Act autonomously
6. Learn with local learning rules

Requirements 1-5 are ubiquitous phenomena—even organisms with as few as 300 neurons display them. Requirement (6) is needed to ensure that the model is biologically plausible, as neurons in an organism can only update the strength of their synaptic connections with information that is present at the synapse (Bredenberg and Savin, 2023). Artificial intelligence algorithms traditionally proposed as models of biological intelligence, such as reinforcement learning (RL), struggle to exhibit these traits. RL agents typically require complex reward shaping to pursue different goals (1), often do not balance exploratory and exploitative states (2), and struggle to adjust to changing environments (4). Moreover, they traditionally rely on global rather than local learning rules (6). As such, it is time to look for a more elegant model of biological intelligence.

Decades of research have demonstrated that predictive coding (PC) plays an important role in the nervous system (Huang and Rao, 2011). PC is founded on the idea that neurons attempt to predict their inputs. It is promising as a starting point for our model because it immediately satisfies requirements (3) and (6). PC networks can learn associations if we fix their inputs and outputs, as they approximate backpropagation (Millidge et al., 2020). They also have local energy-based learning algorithms to update neuronal activities and weights, meaning synaptic strengths are updated based only on information that is plausibly available at biological synapses.

Recently, active PC networks have also been linked to reward-seeking behavior (Ororbia and Mali, 2023; Rao et al., 2022), a step towards requirement (1). However, these models currently rely on external goal signals, a violation of (5) (Ororbia and Mali, 2023) or other conditions that prevent them from constituting a plausible model of biological intelligence. Therefore, Chenguang proposed a new model that uses active PC, along with weight noise, activity noise, and sensory adaptation (all biologically plausible phenomena) to explain simple biological intelligence (Li et al., 2023).

To investigate its properties, Chenguang placed her model in a classical RL bandit task, where the agent selects "levers," receiving their corresponding rewards as sensory feedback in the next time step. When the agent remains at a single lever, the input and output to the network will be effectively fixed. Because PC networks learn associations in this situation, the agent will learn to predict the output neural activity that leads to the reward stimulus it received. Since the action it takes is a function (the `argmax`) of its output activities, it will learn to return to the source of its most recent reward.

Without any noise, this model will fall victim to the dark room problem, staying in one location (regardless of its reward) forever. However, Chenguang showed that injecting noise into the network weights and neuron activities causes the model to explore. This makes sense intuitively – adding noise makes the network make more random decisions. However, it will move to locations with high and low rewards symmetrically, only searching for locations that minimize its prediction error when it makes a random move.

To reproduce the reward-seeking behavior seen in organisms, Chenguang added an "adaptation" term that decays the reward sensed by the model as a function of the time spent at a location. This is biologically plausible, and it causes behavior that appears reward-seeking because, when the sensory input is small, the output of the network is dominated by the neural activity noise, and the agent will randomly move to a new location. Since it takes longer for larger rewards to decay, the agent can stick to these locations for longer, thereby spending more time at locations of higher reward. She also showed that this behavior translates to a two-dimensional reward landscape. Nicely, seeking multiple different types of rewards at once is a simple generalization because it amounts only to including new sensory neurons.

The model therefore satisfies requirements (1), (2), (5), and (6). We have yet to illustrate (4). Moreover, while we know that PC networks can learn associations during target propagation training phases, it is not immediately clear that it achieves this dynamically in an environment in order to exploit reward. Therefore, our goal in this project was to work with Chenguang to illustrate that the agent can adjust to changing environments and learn associations to exploit reward, and make adjustments to the model if it cannot.

Algorithm 1

```
Inputs: weights $\{\mathbf{W}_1, ..., \mathbf{W}_{L-1}\}$; activities $\{\mathbf{x}_1, ..., \mathbf{x}_L\}$; learning rates $\gamma, \alpha = 0.01$; noise parameters $\sigma_w, \sigma_x$; activity history update rate $\beta = 0.1$; adaptation rate $\tau$; constant $C = 0.25$, activity settling time $J = 100$
$\mathbf{r_0} \leftarrow \mathbf{0}$ {Initialize reward value}
$\mathbf{x} \leftarrow \mathbf{0}$ {Initialize activities}
$\hat{\mathbf{x}} \leftarrow \mathbf{x}$ {Initialize activity histories}
for $t = 1; t = T; t + 1$ do
    $E \leftarrow$Energy$(\mathbf{W}, \mathbf{x}, \hat{\mathbf{x}}, C)$
    for $i = 1; i = J, i + 1$ do
        for $l = 2; l < L; l + 1$ do
            $\mathbf{x}_l \leftarrow \mathbf{x}_l - \gamma \frac{\partial E}{\partial \mathbf{x}_l} + \sigma_x \mathcal{N}(0,1)$ {Update activities}
        end for
    end for
    $E \leftarrow$Energy$(\mathbf{W}, \mathbf{x}, \hat{\mathbf{x}}, C)$
    for $l = 1; l = L - 1; l + 1$ do
        $\mathbf{W}_l \leftarrow \mathbf{W}_l - \alpha \frac{\partial E}{\partial \mathbf{W}_l} + \sigma_w \mathcal{N}(0,1)$ {Update weights}
    end for
    $\hat{\mathbf{x}} \leftarrow (1 - \beta)\hat{\mathbf{x}} + \beta \mathbf{x}$ {Update activity histories}
    $\mathbf{r_t} \leftarrow$ Env$(\mathbf{x})$ {Get reward from environment}
    $\mathbf{x}_1 \leftarrow \mathbf{r_t} - \mathbf{r_{t-1}} + (1 - \tau)\mathbf{x}_1$ {Set reward as input, with adaptation}
end for
```

All code used in this study can be found at this GitHub repository.

Methods

The Model. We implement a predictive coding network that minimizes an energy term that is a sum of prediction errors so that neurons learn to anticipate future activities based on prior layers' activations and weights. Specifically, the energy term is

$$E = \sum_{l=2}^{L} \frac{1}{2} \left(\frac{\mathbf{x}_l - \mathbf{W}_{l-1}\mathrm{Relu}(\mathbf{x}_{l-1})}{\hat{\mathbf{x}}_l + C} \right)^2 \qquad \textbf{(1)}$$

where $l \in \{1, \ldots, L\}$ is the network layer index, $\mathbf{x}_l$ are the neuron activities, $\mathbf{W}_l$ are the weight matrices from layers l to $l+1$, and C is a constant used to control for near-zero denominators. Additionally, we divide the energy by the recent activity $\hat{\mathbf{x}}_l$ in Equation 1 to encourage nonzero neuron activities. Intuitively, minimizing this energy function with respect to the $\mathbf{x}_l$ adjusts the neurons' activities to match their predicted values, $\mathbf{W}_{l-1}\mathrm{Relu}(\mathbf{x}_{l-1})$, and then minimizing with respect to weights $\mathbf{W}_l$ improves these local predictions.

In previous studies, predictive coding models have been used to memorize associations by fixing the first and last layers, and training until the energy converges to some minimum. In this study, we place an agent in a reinforcement learning-style environment in which the agent chooses actions corresponding to predetermined rewards, and the maximally activated neuron at the last layer (the motor layer) determines the action that the agent chooses. In Algorithm 1, we show a trial loop for one lifetime of our model (which we will call an "agent") from $t = 1$ to T. Throughout the agent's lifetime, the network's energy is minimized with respect to $\mathbf{x}$ and $\mathbf{W}$. Note that the activities and weights are crucially updated using only local terms. For each timestep, we inject normally distributed noise into both the activities and the weights with mean 0 and variance σ_x and σ_w, respectively. At each step, the agent chooses a reward corresponding to the activations of its output activities, and the reward is fed (with adaptation) into the agent as an input. The agent thus navigates its environment autonomously. As previously discussed, injecting noise into the network and including sensory adaptation are essential for getting agents to display reward-seeking behavior.

Experiments. In this study we use two different types of environments to test the behavior of the agents: 1. a so-called "bandit task" environment in which the agent is presented with two levers of different value and picks a lever to get reward from each timestep, and 2. a smooth, 2-dimensional grid of reward value in which the agent chooses to move north, south, east, or west each timestep. The last row of neurons in our agent corresponds to the motor actions of its environment (lever 1/lever 2 or north/south/east/west) and the agents navigate their environments by moving to the location corresponding to the highest motor neuron activation, taking the reward (and in some cases, environmental context cues) as input, conducting an energy minimizing step that results in motor activations, and so on.

Two-Dimensional Reward Contours. To investigate whether our predictive coding model has the capacity to adapt to changes in its environment, we set up an experiment with a changing 2-dimensional landscape and studied its ability to continue to pursue rewards. We created two reward contours in the shape of square pyramids whose apexes were in opposite locations which are shown in the left side of Figure 1. The maximum reward of the contour is 500 which is at the apex of the pyramid, and the edges of the contour have reward 0. We deploy 500 agents to random locations on the contour and let them operate according to Algorithm 1 for $T = 100,000$ timesteps. If an agent chooses to continue past the edge of the contour, they are transported to the other side. Every $10,000$ timesteps, we alternate between the two contours with opposing apex locations. The architecture of the agents is a 3-layer network with 1 (input), 30 (hidden), and 4 (motor) neurons. The parameters for the algorithm are also included in Algorithm 1.

Learning Association. With this experiment, we would like to illustrate that the agent can learn associations while interacting with an environment, and then use these associations to seek reward. In previous bandit tasks, there were no associations to learn. In collaboration with Chenguang, we develope a new experiment that can test whether or not the agents learn associations that help them seek reward.

We place the agent in a bandit task with two levers that correspond to rewards of 0.0 and 0.5. Contrary to typical bandit tasks, the rewards assigned to the two levers flip periodically with some timescale. Each reward configuration is associated with a context signal, 0.0 or 0.55. Some of the

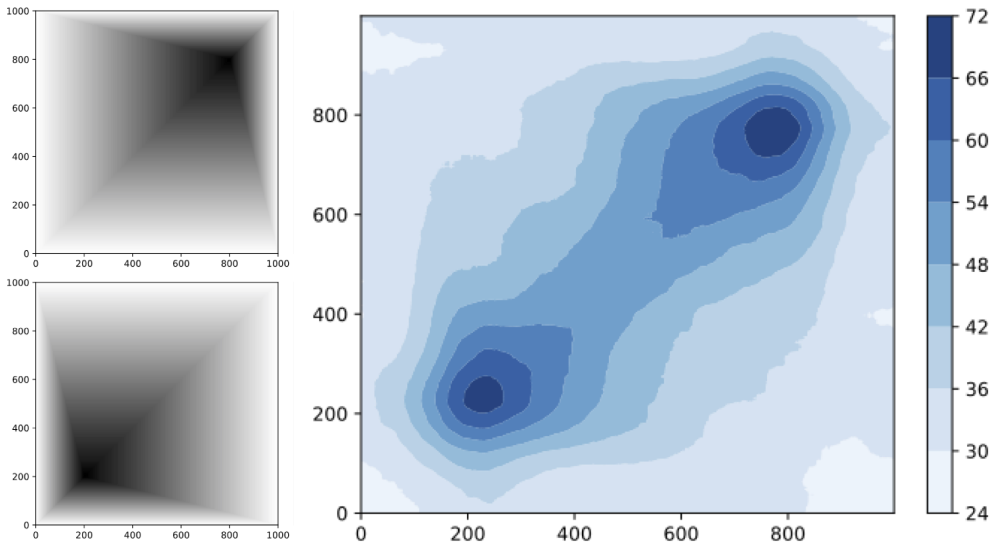

Fig. 1. The left two panels show the square pyramid shaped reward contours that agents operate on throughout their loops. Described in Section , the environment alternates between these two reward contours every $10,000$ timesteps. The right panel shows the path density of the 500 agents over the contour. To smooth the results, we apply a uniform filter of size 100.

agents that we test will receive this cue through a second input neuron.

If the active PC network is capable of learning associations and using them to seek reward, then the agent should, through exposure to this environment, learn to associate the context cue with the correct lever to pull. Moreover, it should use this association to seek reward more effectively.

Results

Changing Environment. In Figure 1 we show how the predictive coding model adapts to a changing environment by continuing to maximize rewards. The agents clearly tend to ascend the gradient, as the number of steps tallied near the peaks of the gradients is around three times the number at low reward regions like the edges or the upper left and lower right corners. Furthermore, the agents seem to be able to adapt to the changing environment, which is demonstrated by *both* the peaks having a comparable number of steps even though the contour is changing throughout the lifetime of the agents.

In addition to the square pyramid, we followed the same methodology with a stationary, Gaussian-shaped reward contour at the center of the map, the results of which are shown in Figure 2. We observe that the agents tend to stay *away* from the center of the distribution, clearly indicating that the agents do not maximize their rewards. There does, however, seem to be a slightly higher density at the very center of the distribution. We hypothesize that the agents struggle to ascend the steep reward gradient at the sides of the Gaussian distribution, but if they are able to reach the top they stay there for a while. This is because the agent avoids large changes in rewards, which are unpredictable. Also, the model likely avoids the very small reward values on the edges of the landscape because the activity history term in the denominator of the energy encourages high neuronal activities, which are achieved with higher reward inputs. So, agents tend to remain at the middle ground between very small rewards and very large reward gradients, which is either the edge or the top of the Gaussian.

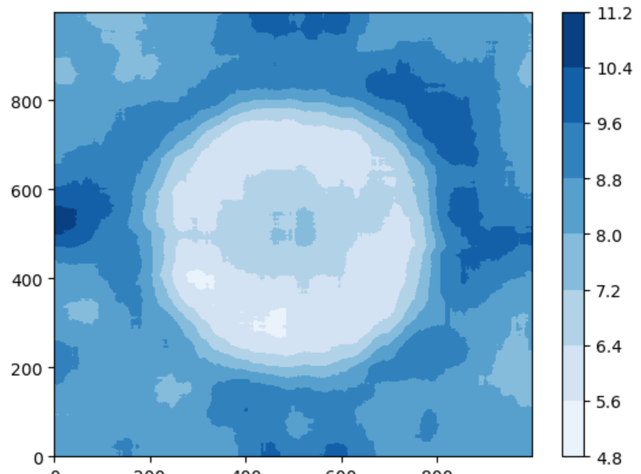

Fig. 2. The same as the right panel of Figure 1 except with a stationary, Gaussian shaped reward contour.

Learning Assocations. If a PC network learns associations and uses them to exploit reward, agents should display the following behaviors in the alternating bandit task experiment. (1) Feeding context and reward pairs into the network and performing a full feedforward pass should result in motor neuron activities that pull the reward lever. (2) After enough exposure to the environment, an agent that receives the context signal should accrue more reward than one that does not. Similarly, the agent that receives the context cue should learn to adjust more quickly to the new configuration after the context changes than an agent that does not get this signal.

To explore whether PC agents can learn associations to help exploit reward, we test a few different architectures.

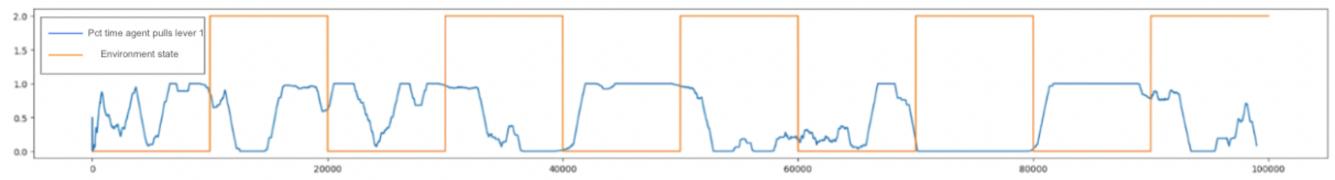

Fig. 3. The orange line shows the environment state, which oscillates every $10,000$ steps. The blue line is the percent of time that the agent pulls lever 1 over the previous $1,000$ steps.

Specifically, (as a sanity check for our experiment) we test a naive network that only receives the reward signal, a network with densely connected context and reward input neurons, and a network with a sparsely connected context neuron and a densely connected reward neuron.

Sanity Check. To begin, we needed to ensure that a naive agent with no context cues still effectively exploits reward in an alternating bandit task. This is an important control for comparing with the performance of agents that receive context cues.

The architecture of the network in this sanity check was 1 sensory, 30 hidden, and 2 motor neurons. The environment switches reward configurations every $10,000$ timesteps, and the agent lives for $T = 100,000$ timesteps. The only input was the reward signal.

In Figure 3, we show that this naive model does exploit reward. As the environment switches, we see that the model's average lever choice alternates. Surprisingly, it also appears as though the agent learns to adjust more quickly to reward flips over time. This is unexpected because the agent receives no context cue that indicates an environment change.

We postulated that learning plasticity could be a potential explanation for this phenomenon. Learning plasticity is the phenomenon of a network altering its weight configuration to be able to switch its behavior more quickly. In the context of our experiment, a network demonstrating learned plasticity would possess a weight configuration that produces nearly identical (but different enough to be distinct after injecting noise) motor neuron responses when a reward is given. This resemblance in responses would enable it to swiftly alter its choice of lever when there are changes in the environment.

If models learn plasticity, it would already be an example of learning to seek reward, which would be an incredible result given the simplicity of the network.

To investigate this property, we evaluate the output of the network after a full feedforward pass given a reward input of 0.5 for all $t \in T$. We then calculate the absolute difference between the two motor neuron activities, which can be seen in Figure 4. If the network is indeed learning plasticity, we would expect to see this absolute difference decrease on average as a function of time. However, this did not appear to occur—the absolute difference shows no trend over time.

While the model did not seem to learn plasticity, we wanted to test whether PC agents without context cues on average learned to react more quickly to consistent environmental changes over time. To measure this, we defined the agent's "reaction time" as the number of timesteps after an

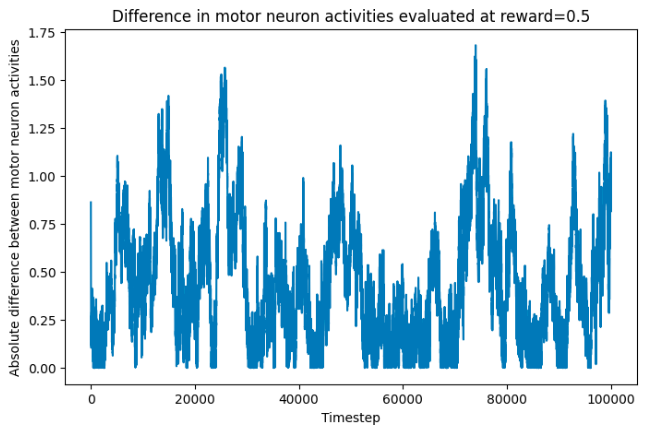

Fig. 4. The blue line shows the absolute difference between the response of the motor neurons to the reward input over time.

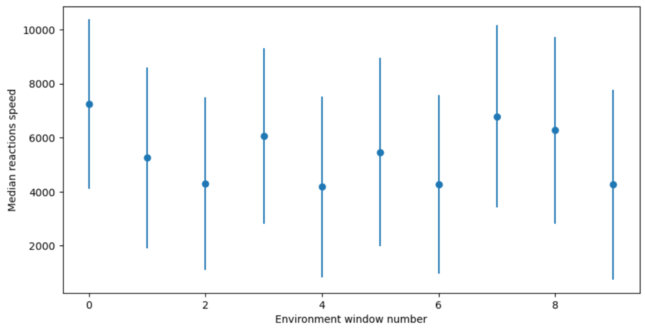

Fig. 5. The points denote the average reaction time of the agent to a change in the reward configuration as a function of the number of environmental oscillations.

environmental switch that it takes to receive reward 90% of the time over a window of $1,000$ timesteps. The reaction time can therefore range from $1,000$ timesteps (it reacted nearly immediately) to $10,000$ timesteps (it never switched to the new reward lever). While somewhat arbitrary, this definition captures the effect that we observed preliminarily in the original sanity check seeds, and its general trends remained consistent despite changes in both the percentage and the number of timesteps used.

We ran the sanity check experiment with 100 agents for $T = 100,000$ timesteps. For each agent, we calculate the reaction time after every reward switch. In Figure 5, we plot the average reaction time over all the agents as a function of the number of environmental oscillations it has experienced in the experiment. If the agents were in fact learning to react more quickly, we would expect this metric to decrease as it experiences more oscillations. However, this is not the case – we observe no obvious trend and very high variance in reaction

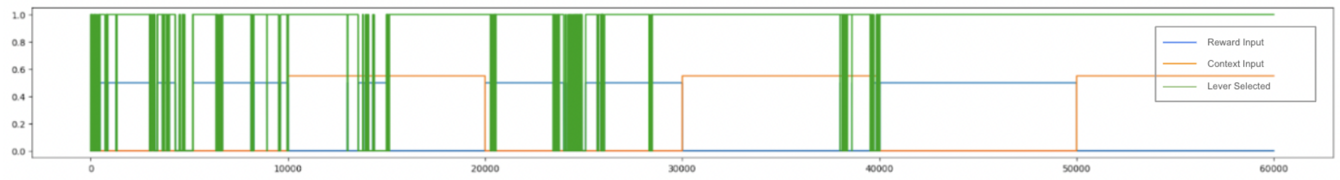

Fig. 6. The green line plots the agents lever selection over time. The blue line represents the reward the agent receives, and the orange line plots the contex cue that the agent senses. Note that either blue or orange is always nonzero when the agent selects lever 1.

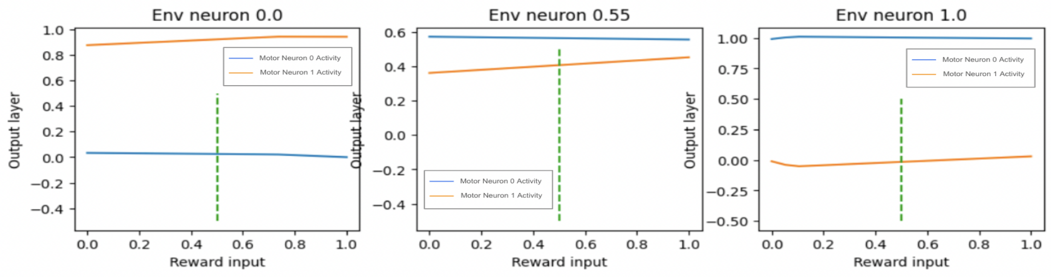

Fig. 7. The blue and orange lines are the responses of motor neurons 0 and 1 to the given inputs. Each plot represents the responses under a fixed environment input, shown in the title, as the reward input varies. The green line shows the nonzero reward that the agent can receive during the task.

times in Figure 5.

Learning Context Clues. After the sanity check, the next step is to introduce context cues. Naively, we began by adding a second sensory neuron to receive a context input. We quickly realized that this would not work; due to the symmetry between the two neurons, the network acts as if both inputs were reward signals. This destroyed all reward-seeking behavior. In Figure 6 we observe that an agent with the symmetric, densely connected configuration learns to remain at the lever that gives no reward when the context cue is 0.55. This behavior is a result of the agent treating both context and reward as reward—it remains at the lever that always gives a nonzero stimulus (i.e. when context = 0, reward = 0.5, and when context = 0.55, reward = 0). In principle, our model seeks reward by being driven away from small inputs, so because this configuration always has at least one relatively large input, the model stops seeking reward.

Encoding Context vs. Reward. The densely-connected context neuron model fails to seek reward; this raises a fundamental question: How can we encode the difference between context cues and rewards in the active PC network model?

It is clear that we must introduce an asymmetry between the two inputs. Here, we present Chenguang's proposed solution and preliminary results. While it is a great improvement, her model does not yet exhibit all the desired phenomena. It learns associations between context and reward, and it uses these associations to seek reward, but it does not do so in a dynamic and autonomous manner. We include her work to motivate our discussion of future steps and our proposed alternative model for encoding the difference between context and reward signals.

Chenguang proposes that we encode the difference by sparsely connecting the context neuron to the rest of the network while maintaining dense reward connections. Because the reward neuron is more connected, the network's activity should significantly decrease when it receives a zero input, even if the context input is nonzero. This will allow the activity noise to take over for low reward inputs, driving a random decision and moving the agent away from zero reward locations. However, because the context signal still gets propagated down the network, the agent should still be able to use context information to learn.

Chenguang placed an agent with three sensory neurons: one reward, one context, and one bias, 30 hidden neurons, and two motor neurons into the same alternating bandit task. The bias and context neurons were only sparsely connected to the hidden layer. This agent once again seeks reward; however, it does not learn to associate the context with the reward location. So while we can give it context cues, it is not clear that the model can learn associations between context and reward location.

To test this, she introduced a passive training phase. She conducted target propagation before placing the agent in the bandit task by fixing inputs and outputs and letting the network's weights converge. In theory, this should work because PC networks approximate backpropagation.

First, to confirm that the training worked, she fed all possible input pairs through the network and plotted the motor neuron responses in Figure 7. These plots confirm that the network learned to associate the context cue to the reward during training—for a given environment input (which corresponds to the different panels), the agent selects the lever with the reward.

She then ran the agent through the alternating bandit task through 50 seeds, tracking the number of reward-seeking lever pulls in each. She compared the results of this experiment to one with a naive PC network that received no passive training phase. The results, shown in Figure 8, confirm what we see above: the pre-trained agent uses the learned associations between the context cues and the location of the reward to more effectively seek out reward stimuli.

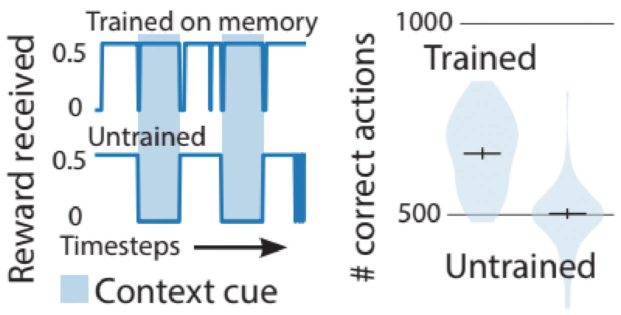

Fig. 8. From Chenguang. The figure to the left shows example behavior for a trained vs untrained agent. The figure to the right shows that the trained network takes more reward seeking actions on average. Bars denote the standard error.

Success at seeking reward for a sustained period of time suggests that the associations learned by the agent in the training phase persist throughout the task. To confirm, Chenguang re-plots the network outputs as a function of input after the task. The results were similar to Figure 7, highlighting that a pre-trained agent can use learned associations to seek out reward in a sustained fashion, an exciting result.

We note that, in this context, the fact that the pre-learned associations persist through the task could simply reflect a lack of dynamic learning. We could just as well have pre-trained the network to seek out zero reward levers given the context cues. Though we would have to confirm this experimentally, it seems likely that if we drop this agent into the bandit task, it would seek out zero reward levers (albeit perhaps with less success because of activity noise). If the network was a true model of biological intelligence, it would not seek out zero reward in a sustained fashion. Rather, over time, it would flip the context cue associations that it learned in order to seek reward instead. There is no evidence here to suggest that this dynamic unlearning of "bad habits" will occur. Future work is needed to refine this model so that the agent learns associations dynamically to produce reward-seeking behavior.

Discussion

Another Proposal for Encoding Context vs. Reward. Another approach for differentiating between context cues and reward is incorporating the reward signal as a change in the weight update rate. More reward corresponds to faster updates, and less reward corresponds to slower updates. Recent studies have argued that neuromodulatory systems like dopamine actually play this exact role in the brain (Coddington et al., 2023). This model may appear to violate requirement (6), but neuromodulatory signals are present at the synapse, so our locality condition is retained.

We argue that this may cause the agent to learn associations that help it seek reward because, during phases of reward exploitation, the inputs and outputs of the network are nearly fixed. This means that, if we allow the weights to update, we will approximate backpropagation. So if there is reward, the agent will learn to predict its environment, repeating its choices. If there is no reward, the weights will not update and the agent will not learn. Therefore, the only associations that the agent will be capable of learning are those between inputs and output pairs that correspond with rewards.

However, this only explains how the agent could learn associations between context cues and reward locations. It does not explain the mechanism by which the agent will be pushed away from areas of low reward, which is also required if we want to produce reward-seeking behavior. One proposal to push the agent away from areas of low reward could be to implement weight decay in the network. This decay must be small enough to be negligible when the weights are updating (i.e., when there is reward) but large enough to tend to zero when the weights are not updating (i.e., when there is no reward). Weight decay when there is no reward will slowly send the network activities to zero, allowing activity noise to take over, thereby driving the agent away from the zero reward locations. This idea faces the potential drawback of slowly erasing previously learned associations. Future work will need to explore and refine this hypothesis. One possible alternative would be to drive the agent away from low rewards by combining our proposal with Chenguang's. The network could receive reward information both as densely connected input and as a modulation of the weight update rate.

Next Steps.

3 Months. As we hinted at earlier, future work is still needed to build a model that *dynamically* learns associations and uses them to seek reward. The best approach is not immediately clear. With 3 more months to work on this project, we would (and will) continue to explore the two proposals for encoding reward vs context signals to create an agent that learns associations in silico and uses them to seek reward.

When this is complete, we would design an experiment that can test the model in a more biological environment (i.e. one similar to a 2D reward landscape) to show that it satisfies all 6 requirements simultaneously! If successful, it would be interesting to design experiments that could generate behavioral predictions for comparison with simple organisms.

6 Months. With even more time, it would be interesting to start expanding our definition of simple biological intelligence to include slightly more complex behavior.

For instance, another ubiquitous trait across organisms is avoidance. Right now, the PC network passively leaves areas with little reward due to activity noise. This is different from learning to actively avoid danger. With the current model, it is not immediately clear how one might implement this. One interesting idea would be to add a sort of adversarial second network that has the same structure but senses "danger" instead of reward. This network could learn to associate context clues with danger, and whenever it is in an exploitative state (i.e. successfully predicting danger), try to drive the original PC network to move away.

An even more complex but interesting problem would also be to explore how one might introduce collective behavior between these models. One could imagine an agent receiving a context cue that depends on the actions of nearby agents. It would be exciting to explore how we could use this idea to induce coordinated behavior in an environment. For instance, consider a set of ants buried underground. Any given ant cannot dig themself out of the ground fast enough to avoid asphyxiation. However, if multiple ants try to dig in a coordinated fashion, they can all escape. Can we reproduce something along the lines of this simple cooperative behavior with active PC networks that satisfy the 6 requirements for simple biological intelligence? There is a whole host of interesting problems to explore.

Conclusions

We have shown that our initial PC network model of biological intelligence satisfies requirement (4), it can adjust readily to changes in the environment. We have also made the first steps to improving the model so that it can satisfy requirement (3) in a natural setting, learning while interacting with its environment, and using this understanding to more effectively seek reward.

ACKNOWLEDGEMENTS

We thank Chenguang Li for guidance and advice throughout the semester. We are grateful to both Chenguang and Professor Sharad Ramanathan for the course and the opportunity to engage in this exciting project.

References

Chenguang Li, Jonah Brenner, and Adam Boesky. Neuron-level prediction, adaptation, and noise can implement reward-seeking behavior, 2023.

Colin Bredenberg and Cristina Savin. Desiderata for normative models of synaptic plasticity, 2023.

Yanping Huang and Rajesh P. N. Rao. Predictive coding. *WIREs Cognitive Science*, 2(5):580–593, 2011. doi: https://doi.org/10.1002/wcs.142.

Beren Millidge, Alexander Tschantz, and Christopher L. Buckley. Predictive coding approximates backprop along arbitrary computation graphs, 2020.

Alexander Ororbia and Ankur Mali. Active predictive coding: Brain-inspired reinforcement learning for sparse reward robotic control problems. In *2023 IEEE International Conference on Robotics and Automation (ICRA)*, pages 3015–3021, 2023. doi: 10.1109/ICRA48891.2023.10160530.

Rajesh P. N. Rao, Dimitrios C. Gklezakos, and Vishwas Sathish. Active predictive coding: A unified neural framework for learning hierarchical world models for perception and planning, 2022.

Luke T. Coddington, Sarah E. Lindo, and Joshua T. Dudman. Mesolimbic dopamine adapts the rate of learning from action. *Nature*, 614(7947):294–302, January 2023. ISSN 1476-4687. doi: 10.1038/s41586-022-05614-z.