Flexible navigation with neuromodulated cognitive maps

Krubeal Danieli

Contents

1 Introduction

During navigation, animals dynamically create rich representations of the environment, forming personalized cognitive maps. The hippocampal area CA1 features spatial cells that adapt based on behavior and internal states. Computational models have usually obtained spatial tuning by training a deep recurrent network for solving navigation tasks such as path integration [1, 2, 3], lasting multiple numerous epochs and using backpropagation. However, these training methods do not closely align with real-time local learning paradigms used by animals.
In this study, it is introduced a rate model that generates place cells in one-shot as the agent navigates the environment by simply assigning the current spatial observation to a selected neuron while ensuring a sparse representation (*i.e.* spaced place fields).

An important ingredient for the learning dynamics of our model is neuromodulation. Neuromodulation is an important ingredient for biological neuronal dynamics, with different molecules covering a wide range of functions. Previous models [4, 5] inspired by experimental results crafted a simple spiking plasticity rule for reward-directed navigation where acetilcholine mediates explorative behaviour and dopamine reinforces memory of reward locations. Other approached using deep artificial networks have applied neuromodulation in conjuction with other training practices, such as dropout probability [6]. In this work, modulators are described as synaptic resources that are consumed by plasticity events, and their dynamics are modelled as leaky integrators. Further, acetilcholine is used to mediate the generation of new place fields, while dopamine mediates the slow remapping of the place centers in conjunction with a reward signal. The concentration of acetilcholine is affected by the presence of active neurons or by the occurrence of a weight update. Dopamine, on the other hand, is influenced by the presence of a reward.

This model successfully creates a representation of visited areas and recurrent connections are defined among similarly tuned cells. Importantly, plasticity hyper-parameters such as the equilibrium concentration and decay time-constant of modulators influence the density of place cells, impacting the encoding of behaviorally relevant information [7].

This network is then used to solve a goal-directed navigation task, where the agent is trained to reach a target location. The agent is equipped with a policy that modulates the exploration behaviour and the decision-making process.

2 Methods

2.1 Architecture

The formation of a spatial representation is achieved by tuning new neurons to a representation of the current position. The model is constructed with a feedforward architecture, where an input position vector $\mathbf{x} = (x, y)$ is passed through a spatial layer M_{sp} and then to the main network M_{pc} of un-tuned cells. In this work, the intermediate spatial layer serves the purpose of increasing the dimensionality of the input for robustness reasons, since our synaptic plasticity works better with more discrete vectors. Ideally, position-related information is extracted from more or less raw sensory data, but for simplicity here it is chosen a network of place cells with hard-coded Gaussian fields tiling the entire environment.
The activity of neurons in M_{pc} is defined by the sum of the feedfoward input $\mathbf{x}_{\text{sp}}$ and recurrent activity:

$$\mathbf{u} = \mathbf{W}_{\text{ff}}\mathbf{x}_{\text{sp}} + \mathbf{W}_{\text{rec}}\sigma(\mathbf{u}) \tag{1}$$

where σ is a generalized sigmoid used as activation function.

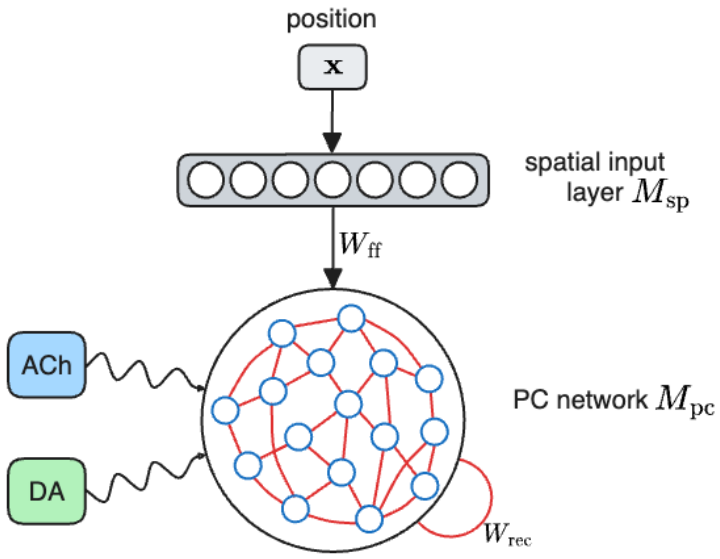

Figure 1: Architecture of the model

The feedfoward connections W_{ff} are plastic and their weight update is dependent on the concentration of acetilcholine (ACh) and dopamine (DA).
The formation of new cells is mediated by ACh, while DA medites the remapping on cells. The recurrent connections W_{rec} are updated everytime a new

neuron gets tuned, and are calculated with a K-Nearest-Neighbor algorithm (KNN) based on the cosine distance between weight vectors ($W_{\text{ff};1,}, W_{\text{ff};2,}$) and a maximum distance threshold.

2.2 Plasticity

The generation of new place cells follows a heterosynaptic plasticity rule, where the weight update is proportional to the current acetylcholine level (c_{ACh}) and the difference between the input vector and the current weights. Importantly, the neuron to be plastic (i) is randomly selected from the pool of un-tuned neurons. This approach rapidly forms new tuning while relying on competition to ensure sparse encoding.

The weight update for new place cell generation is given by:

$$\Delta W_i = \text{ReLU}(c_{\text{ACh}} - \theta_{\text{ACh}})(x_i - W_i)\phi_- \tag{2}$$

where ϕ_- implements lateral inhibition, a homeostatic mechanism preventing the overlap of multiple place fields, and ReLU is a rectified linear function with threshold θ_{ACh}.

The remapping of existing place cells is mediated by dopamine and implemented similarly, but with an additional term ϕ_+ that accounts for the effect of distance between the place field and the current location:

$$\Delta W_i = \eta\ \text{ReLU}(c_{\text{DA}} - \theta_{\text{DA}})(W_i - x_i)\ \phi_-\ \phi_+ \tag{3}$$

where η is the learning rate, c_{DA} is the dopamine concentration, ϕ_- is lateral inhibition as before, and ReLU is a rectified linear function with threshold θ_{DA}.

For more details see the Appendix 4.

2.2.1 Neuromodulation

The dynamics of both neuromodulators, acetylcholine (ACh) and dopamine (DA), follow leaky integrator models and are constrained within the range $[0, 1]$.

Acetylcholine It is present by default with an equilibrium concentration of 1. However, its concentration can be reduced to zero under two conditions: (1) when an active neuron exceeds a threshold θ, or (2) during an ACh-dependent weight update. The dynamics of ACh concentration are described by:

$$\tau \dot{c}_{\text{ACh}} = 1 - c_{\text{ACh}} - \mathbf{1}_{\max(u)>\theta} - \mathbf{1}_{\sum_{i,j} \Delta W_{\text{ff},ij}>0} \tag{4}$$

where τ is the time constant, c_{ACh} is the ACh concentration, u represents neuronal activities, and $W_{\text{ff},ij}$ are the feedforward weights.

Dopamine In contrast to ACh, dopamine has a default equilibrium concentration of zero, reflecting the occasional nature of remapping events. The DA dynamics are similar to those of ACh but are additionally modulated by reward. The equilibrium value E_{DA} increases to 1 in the presence of a reward. The DA concentration is governed by:

$$\tau \dot{c}_{\text{DA}} = E_{\text{DA}} - c_{\text{DA}} - \mathbf{1}_{\sum_{i,j} \Delta W_{\text{ff},ij} > 0} \tag{5}$$

where c_{DA} is the DA concentration, and E_{DA} is the equilibrium value (0 by default, 1 in the presence of reward). In both equations, $\mathbf{1}_{\{\cdot\}}$ denotes the indicator function, which equals 1 when the condition in the subscript is true and 0 otherwise.

2.3 Reinforcement Learning

The model described above is incorporated into an agent trained to solve a simple goal-directed navigation task. The learning process is divided into two phases: exploratory behavior and goal-directed behavior.

2.3.1 Exploratory Behavior

In the initial phase, the agent is programmed to explore the environment extensively, forming a comprehensive place cell representation. The exploration policy is based on the current position representation $\mathbf{u}_t$, the proximal positions representation $\mathbf{u}_{\text{prox}} = \sigma(W\text{rec} \cdot \mathbf{u}_t)$, and a modulation function: $\varphi(\mathbf{u}_{\text{prox}}, \lambda)$

The function φ modulates the representation of proximal positions, potentially biasing nearby or distant place cells. The parameter λ is computed by an external policy (either a reinforcement learning algorithm or a hard-coded heuristic) that takes as input the distance between current and proximal representations. This external policy also computes Θ, a set of crucial hyperparameters for the place cell network, such as the acetylcholine time constant. The velocity vector $\mathbf{v}_t$ is then calculated using a function ϕ: $\mathbf{v}_t = \phi(\mathbf{u}_t, \varphi(\mathbf{u}_{\text{prox}}, \lambda))$.

2.3.2 Goal-Directed Behavior

Once a robust spatial representation is established, a goal is randomly placed within the explored area. The agent's objective is to reach this goal using only locally available information. We designed a decision-making algorithm based on the spatial representation of the current and target positions, assuming the target lies within the explored area (i.e., the agent "knows" where it is). Figure 2 illustrates the policy diagram:

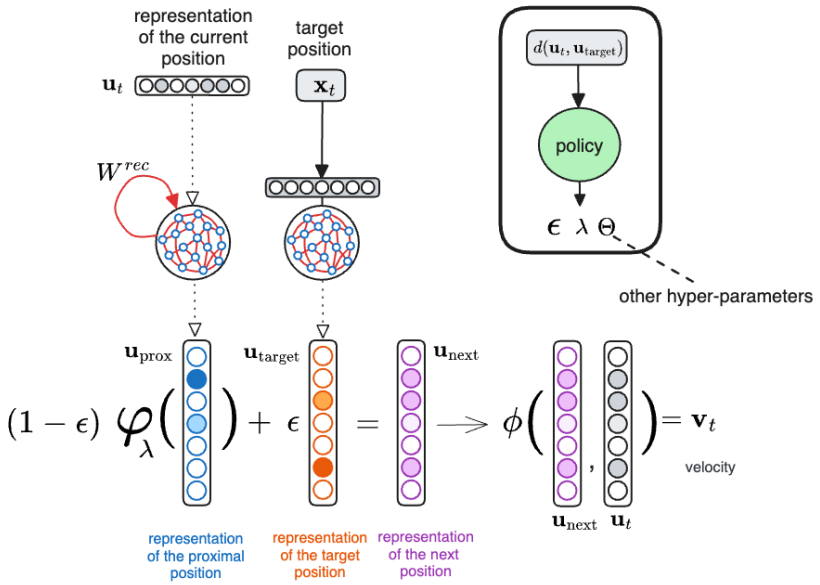

Figure 2: Policy diagram

The functions φ and ϕ operate as described earlier. The parameter ϵ is a scalar that determines the weight of the target position in calculating the next representation. The external policy now receives the distance between current and target position representations as input and returns the current values of parameters ϵ, λ, and Θ. This approach serves two primary purposes. First, it generates behavior solely based on the population vector of the formed place cells. Second, it allows for flexibility in accounting for the drive towards particular proximal positions (given by λ), and the direction of the target according to the agent's state (represented by the distance between the two representations); the parameter ϵ provides the ability to balance these factors in the decision-making process.

3 Results

3.1 Cognitive map formation

The model was validated through the generation and remapping of place cells within a square environment featuring a fixed reward area. The agent's trajectory was simulated as a bio-inspired random walk with constant speed, as illustrated in Figure 3.

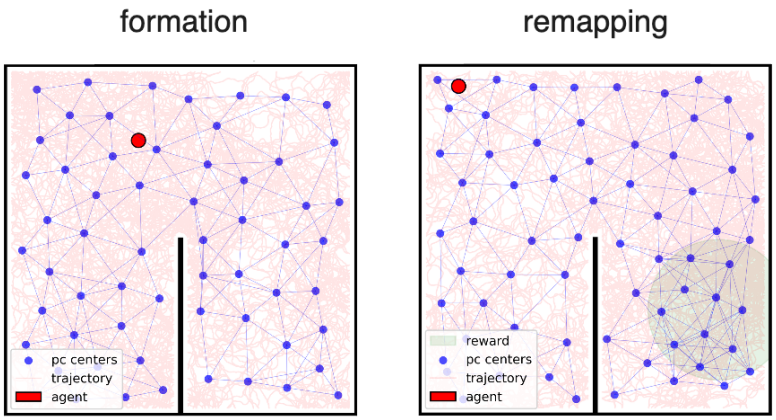

Figure 3: Generation of a Cognitive Map - Left: *place cells (blue dots) are formed online as the agent traverses the environment (red trajectory), supported by cholinergic activity.* Right: *a subsequent run with a reward region in the lower right corner, potentially triggering dopamine spikes.*

The spatial representation developed by the agent reflects its experiences during exploration. In the absence of reward (left plot), the distance between adjacent place fields remains relatively constant, with variations dependent on recent acetylcholine (ACh) consumption rates and trajectory changes. When a reward is present and the agent approaches the reward area (right plot), elevated dopamine levels induce plasticity, causing place fields to cluster closer to the current position.

3.2 Navigation and Reinforcement Learning

[**NB** *this is a draft, the target location here is not a reward area as in the experiment above and dopamine is not activated. That is work in progress not ready to be written down here, for now it is reported only the agent navigation ability with a uniform spatial representation*]
To evaluate the practical utility of the cognitive map, we tested the agent's navigation capabilities in a square environment with walls and a goal placed within the explored area. Figure 4 illustrates the results.

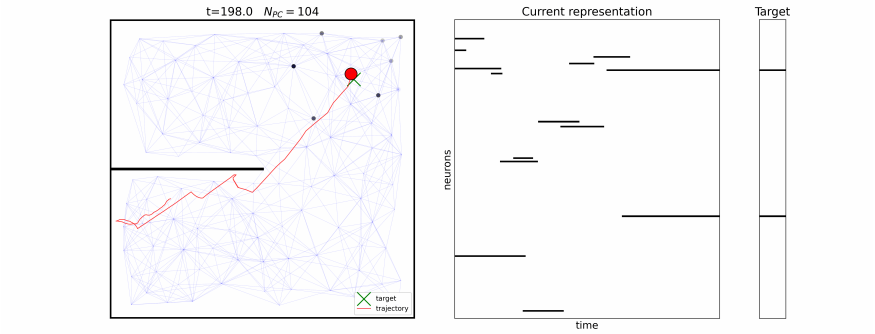

Figure 4: Goal-Directed Behavior - Left: *agent trajectory in a square environment with the target in the upper right corner. Black dots represent the centers of active place cells in proximal positions, connected by blue lines.* Center: *time evolution of the current position representation, with rows corresponding to neurons and columns to time steps.* Right: *representation of the target location.*

Employing the policy outlined in the methods 2.3.2, the agent successfully reached the target via a relatively short path. When a wall is encountered between its starting position and the target, the agent initially collided but managed to change its strategy to avoid it. This adaptation was achieved through temporary adjustments to the parameters ϵ and λ, prioritizing movement towards proximal positions with higher place cell activity over direct target approach.

This demonstrates the effectiveness of the cognitive map in supporting flexible navigation strategies, even with a relatively simple decision-making algorithm.
Importantly, the results presented in Figure 4 were obtained using our hard-coded heuristic policy.

Finally, the agent's ability to dynamically adjust its behavior based on environmental constraints and its internal representation showcases the robustness and adaptability of the developed model.

See also `https://ikiru-hub.github.io/showcase/` for a video demonstration of the place cell generation and navigation tasks.

References

[1] Ben Sorscher, Gabriel Mel, Surya Ganguli, and Samuel Ocko. A unified theory for the origin of grid cells through the lens of pattern formation. In *Advances in Neural Information Processing Systems*, volume 32. Curran Associates, Inc., 2019.

[2] Christopher J. Cueva and Xue-Xin Wei. Emergence of grid-like representations by training recurrent neural networks to perform spatial localization, March 2018.

[3] Andrea Banino, Caswell Barry, Benigno Uria, Charles Blundell, Timothy Lillicrap, Piotr Mirowski, Alexander Pritzel, Martin J. Chadwick, Thomas Degris, Joseph Modayil, Greg Wayne, Hubert Soyer, Fabio Viola, Brian Zhang, Ross Goroshin, Neil Rabinowitz, Razvan Pascanu, Charlie Beattie, Stig Petersen, Amir Sadik, Stephen Gaffney, Helen King, Koray Kavukcuoglu, Demis Hassabis, Raia Hadsell, and Dharshan Kumaran. Vector-based navigation using grid-like representations in artificial agents. *Nature*, 557(7705):429–433, May 2018.

[4] Zuzanna Brzosko, Wolfram Schultz, and Ole Paulsen. Retroactive modulation of spike timing-dependent plasticity by dopamine. *eLife*, 4:e09685, October 2015.

[5] Zuzanna Brzosko, Sara Zannone, Wolfram Schultz, Claudia Clopath, and Ole Paulsen. Sequential neuromodulation of Hebbian plasticity offers mechanism for effective reward-based navigation. *eLife*, 6:e27756, 2017.

[6] Jie Mei, Rouzbeh Meshkinnejad, and Yalda Mohsenzadeh. Effects of neuromodulation-inspired mechanisms on the performance of deep neural networks in a spatial learning task. *iScience*, 26(2):106026, February 2023.

[7] Katie C. Bittner, Aaron D. Milstein, Christine Grienberger, Sandro Romani, and Jeffrey C. Magee. Behavioral time scale synaptic plasticity underlies CA1 place fields. *Science*, 357(6355):1033–1036, September 2017.

4 Appendix

4.1 Neural dynamics

Activation function σ
It is a generalized sigmoid function:

$$\sigma(z) = [1 + \exp(-\beta(z - \alpha))]^{-1} \tag{6}$$

Additionally, the activity is clipped:

$$\text{clip}(z) = \begin{cases} 0 & \text{if } z < 10^{-3} \\ z & \text{otherwise} \end{cases} \tag{7}$$

4.1.1 Lateral inhibition function

The lateral inhibition function ϕ_- is implemented using the feedforward connectivity matrix W_{ff}, hereafter referred to as W. It is calculated based on the cosine similarity among tuned neurons. The process involves several steps:

1. Normalize each row of matrix W by its Euclidean norm:

$$W_{\text{norm};i} = \frac{W_i}{\|W_i\|_2} = \frac{W_i}{\sqrt{\sum_{j=1}^{n} W_{ij}^2}}, \tag{8}$$

where W_{ij} represents the element at the i-th row and j-th column of W, and the division is performed element-wise across rows. For numerical stability, any resulting `NaN` values are set to zero.

2. Compute the repulsion matrix by taking the dot product and setting diagonal elements to zero:

$$R = \left(W_{\text{norm}} \cdot W_{\text{norm}}^T\right) \odot (1 - I), \tag{9}$$

where I is the identity matrix, and $\odot$ denotes element-wise multiplication.

3. Calculate the maximum repulsion value for each row:

$$R_{\max} = \max(R)_i \tag{10}$$

4. Finally, determine the repulsion vector by thresholding the maximum repulsion values:

$$\phi_- = \begin{cases} 1 & \text{if } R_{\max} < \theta_{\text{rep}}, \\ 0 & \text{if } R_{\max} \geq \theta_{\text{rep}}. \end{cases} \tag{11}$$

The threshold θ_{rep} is set to 0.7.

4.1.2 Attraction function

The attraction function ϕ_+ modulates the remapping of place cells with respect to a new position. This weight update depends on dopamine levels and the distance between the current place field and the location where the reward was experienced. The function is computed as follows a cosine similarity between the weight vector $W_{\text{ff};i;}$ and the input vector $\mathbf{x}$, and then passed through a generalized sigmoid function.

This formulation ensures that the attraction effect is strongest for place cells whose fields are closest to the current location, gradually decreasing with distance.

4.2 Decision-making and RL

4.2.1 Velocity calculation

Below, it is described the algorithm the agent uses to reach a goal location given the parameters ϵ and λ. These two parameters, together with the place cell network hyperparameters Θ are defined by an external policy: proximal policy optimization (PPO), deep-Q network (DQN, for which the actions have been discretized into bins), or a hard-coded heuristic.

Algorithm 1 Position and Velocity Calculation in Neural Space

Require:

$\mathbf{x}_t \in \mathbb{R}^2$: current 2D position
$\mathbf{x}_{\text{target}} \in \mathbb{R}^2$: target position
$\epsilon \in [0, 1]$: interpolation parameter
$\lambda \in \mathbb{R}$: modulation parameter
$W_{\text{rec}} \in \mathbb{R}^{n \times n}$: recurrent weight matrix
$\sigma(\cdot)$: generalized sigmoid activation function
$\varphi(\cdot, \lambda)$: modulation function
$\phi(\cdot, \cdot)$: velocity extraction function

Ensure:

$\mathbf{v}_t \in \mathbb{R}^2$: velocity vector
function CalculateVelocity($\mathbf{x}_t, \mathbf{x}_{\text{target}}, \epsilon, \lambda$)
    Calculate neural representation of the current position: $\mathbf{u}_t \leftarrow f(\mathbf{x}_t)$
    Calculate representation of proximal positions: $\mathbf{u}_{\text{prox}} \leftarrow \sigma(W_{\text{rec}}\mathbf{u}_t)$
    Calculate neural representation of target position: $\mathbf{u}_{\text{target}} \leftarrow f(\mathbf{x}_{\text{target}})$
    Modulate proximal representations: $\mathbf{u}'_{\text{prox}} \leftarrow \varphi(\mathbf{u}_{\text{prox}}, \lambda)$
    Interpolate between proximal and target: $\mathbf{u}_{\text{next}} \leftarrow (1-\epsilon)\mathbf{u}'_{\text{prox}} + \epsilon\mathbf{u}_{\text{target}}$
    Extract velocity from neural representations: $\mathbf{v}_t \leftarrow \phi(\mathbf{u}_t, \mathbf{u}_{\text{next}})$
    return $\mathbf{v}_t$
end function

Where f is simply a forward pass to the model (see figure 1). The calculation of velocity through the function ϕ relies on the extraction of a 2D position $(\bar{x}, \bar{y})$

from a spatial representation (*i.e.* a population vector $\mathbf{u}$). This is carried out by taking an average of the place cells center weighted by their activations:

$$\begin{aligned} \bar{x} &= \frac{1}{\sum_i u_i} \sum_i x_i u_i \\ \bar{y} &= \frac{1}{\sum_i u_i} \sum_i y_i u_i \end{aligned} \tag{12}$$

Then, velocity is defined with ϕ by calculating the angle ω between the two computed positions and a given speed s as: $\mathbf{v} = \begin{bmatrix} s \cdot \cos(\omega) & s \cdot \sin(\omega) \end{bmatrix}$.

4.2.2 Proximal modulation

The modulation of the proximal positions representation is done by a function φ, which is described by the following algorithm:

Algorithm 2 Proximal modulation algorithm

Require:

$\mathbf{a} \in \mathbb{R}^n$: population vector for n neurons
$\theta \in \mathbb{R} \in [0, 1]$: activation threshold
$\lambda \in [-5, 5]$: modulation parameter

Ensure:

$\mathbf{y}'$: modulated output array
function ProximalModulation($\mathbf{u}, \theta, \lambda$)
    Indices of super-threshold elements: $\mathcal{I} \leftarrow \{i \in \{1, \ldots, n\} \mid u_i > \theta\}$
    Extract super-threshold values: $\mathbf{v} \leftarrow \mathbf{u}_{\mathcal{I}}$
    Compute the mean of super-threshold values: $\bar{v} \leftarrow \frac{1}{|\mathcal{I}|} \sum_{i \in \mathcal{I}} v_i$
    Apply drift towards mean: $\mathbf{v}' \leftarrow \mathbf{v} + \lambda(\bar{v}I - \mathbf{v})$
    Update super-threshold elements: $\mathbf{u}'_{\mathcal{I}} \leftarrow \mathbf{v}'$
    Preserve sub-threshold elements: $\mathbf{u}'_{\{1,\ldots,n\} \setminus \mathcal{I}} \leftarrow \mathbf{u}_{\{1,\ldots,n\} \setminus \mathcal{I}}$
    return $\mathbf{u}'$
end function

In the exploration phase, no target position is provided. In this case, step 5 is skipped and step 7 is reduced to an equality $\mathbf{u}_{\text{next}} = \mathbf{u}_{\text{prox}}$, resulting in a behaviour solely relying on the proximal positions.