Flexible navigation with neuromodulated cognitive maps

Krubeal Danieli

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

The model is constructed around the concept of a cognitive map, which an agent builds by freely navigating a closed environment and reaching a discovered goal location. The architecture relies on the core assumption that the agent has receives minimal external information, consisting solely of a reward and collision input as two binary values. These two signals are used to enrich the cognitive map with experience-dependent data, which is then used to guide the agent's behavior.

The formation of the spatial representation is instead based on idiothetic information, which is the agent's perception of self-motion. In particular, here we assume this cue to be the factual velocity vector, namely the actual displacement of the agent in the environment. In the brain, this signal is thought to result from the integration of inertial and relative motion cues.

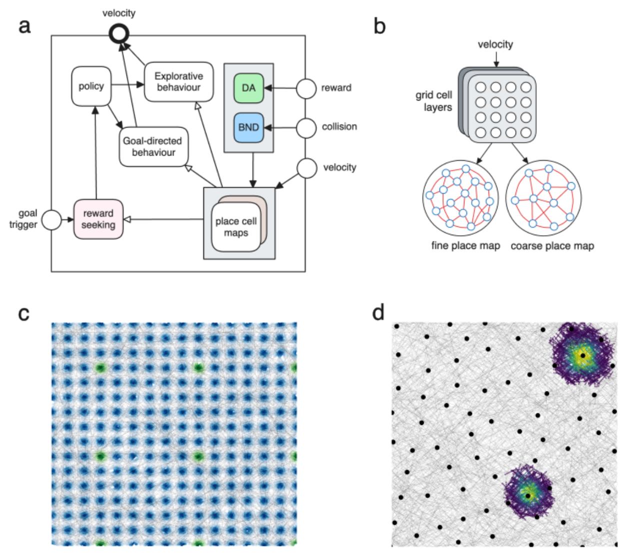

Figure 1: Model layout and spatial representations - **a**: *the full architecture of the model, consisting of three main sensory input, targeting the two modulators and the cognitive map module, and the executive components, represented by a policy module, two behavioural programs and a reward receiver.* **b**: *the cognitive map component, organized with a stack of grid cell modules receiving the velocity input and projecting to two layer of place cells with different place field granularity.* **c**: *the neural activity of a grid cell module from a random trajectory; in blue the repeating activity of all cell, while in green the activity of only one, highlighting the periodicity in space.* **d**: *the distribution in space of the place cells centers, together with the activity of two cells showing the size of their place field*

2.0.1 Place cell map

A grid cell module i has been defined as a set of N^{gc} neurons with gaussian tuining curve evenly distributed over the surface of a two dimensional torus $\mathbf{T}^2$. When the agent moves in the environment, a two dimensional Euclidean space $\mathbf{R}^2$, with a velocity $\mathbf{v} = \{\mathrm{x}, \mathrm{y}\}$, its position on the torus is updated by the same vector but scaled by a speed $\mathrm{s}_{\mathrm{i}}^{\mathrm{gc}}$ local to the grid cell module i. The initial position on the torus is randomly chosen at the beginning of each episode, since what matters is the sequence of displacements, and the speed is specific for each grid cell module.

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.

3 Appendix

3.1 Neural dynamics

Activation function σ
It is a generalized sigmoid function:

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

Additionally, the activity is clipped:

$$\mathrm{clip}(\mathrm{z}) = \begin{cases} 0 & \text{if } \mathrm{z} < 10^{-3} \\ \mathrm{z} & \text{otherwise} \end{cases} \tag{2}$$

3.1.1 Lateral inhibition function

The lateral inhibition function ϕ_- is implemented using the feedforward connectivity matrix $\mathrm{W_{ff}}$, 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:

$$\mathrm{W_{norm;i}} = \frac{\mathrm{W_i}}{\|\mathrm{W_i}\|_2} = \frac{\mathrm{W_i}}{\sqrt{\sum_{\mathrm{j}=1}^{\mathrm{n}} \mathrm{W_{ij}^2}}}, \tag{3}$$

where $\mathrm{W_{ij}}$ 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:

$$\mathrm{R} = \left(\mathrm{W_{norm}} \cdot \mathrm{W_{norm}^T}\right) \odot (1 - \mathrm{I}), \tag{4}$$

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

3. Calculate the maximum repulsion value for each row:

$$\mathrm{R_{max}} = \max(\mathrm{R})_\mathrm{i} \tag{5}$$

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

$$\phi_- = \begin{cases} 1 & \text{if } \mathrm{R_{max}} < \theta_{\mathrm{rep}}, \\ 0 & \text{if } \mathrm{R_{max}} \geq \theta_{\mathrm{rep}}. \end{cases} \tag{6}$$

The threshold θ_{rep} is set to 0.7.

3.1.2 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 $W_{ff;i;}$ 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.

3.2 Decision-making and RL

3.2.1 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:
 1: $\mathbf{x}_t \in \mathbb{R}^2$: current 2D position
 2: $\mathbf{x}_{target} \in \mathbb{R}^2$: target position
 3: $\epsilon \in [0,1]$: interpolation parameter
 4: $\lambda \in \mathbb{R}$: modulation parameter
 5: $W_{rec} \in \mathbb{R}^{n \times n}$: recurrent weight matrix
 6: $\sigma(\cdot)$: generalized sigmoid activation function
 7: $\varphi(\cdot, \lambda)$: modulation function
 8: $\phi(\cdot, \cdot)$: velocity extraction function
Ensure:
 9: $\mathbf{v}_t \in \mathbb{R}^2$: velocity vector
10: function CalculateVelocity($\mathbf{x}_t, \mathbf{x}_{target}, \epsilon, \lambda$)
11:     Calculate neural representation of the current position: $\mathbf{u}_t \leftarrow f(\mathbf{x}_t)$
12:     Calculate representation of proximal positions: $\mathbf{u}_{prox} \leftarrow \sigma(W_{rec}\mathbf{u}_t)$
13:     Calculate neural representation of target position: $\mathbf{u}_{target} \leftarrow f(\mathbf{x}_{target})$
14:     Modulate proximal representations: $\mathbf{u}'_{prox} \leftarrow \varphi(\mathbf{u}_{prox}, \lambda)$
15:     Interpolate between proximal and target: $\mathbf{u}_{next} \leftarrow (1-\epsilon)\mathbf{u}'_{prox} + \epsilon\mathbf{u}_{target}$
16:     Extract velocity from neural representations: $\mathbf{v}_t \leftarrow \phi(\mathbf{u}_t, \mathbf{u}_{next})$
17:     return $\mathbf{v}_t$
18: end function
```

Where f is simply a forward pass to the model (see figure **??**). 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:

$$\begin{aligned}\bar{x} &= \frac{1}{\sum_i u_i} \sum_i x_i u_i \\ \bar{y} &= \frac{1}{\sum_i u_i} \sum_i y_i u_i\end{aligned} \tag{7}$$

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

3.2.2 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:
 1: $\mathbf{a} \in \mathbb{R}^n$: population vector for n neurons
 2: $\theta \in \mathbb{R} \in [0, 1]$: activation threshold
 3: $\lambda \in [-5, 5]$: modulation parameter
Ensure:
 4: $\mathbf{y}'$: modulated output array
 5: function ProximalModulation($\mathbf{u}, \theta, \lambda$)
 6:     Indices of super-threshold elements: $\mathcal{I} \leftarrow \{i \in \{1, \dots, n\} \mid u_i > \theta\}$
 7:     Extract super-threshold values: $\mathbf{v} \leftarrow \mathbf{u}_{\mathcal{I}}$
 8:     Compute the mean of super-threshold values: $\bar{v} \leftarrow \frac{1}{|\mathcal{I}|} \sum_{i \in \mathcal{I}} v_i$
 9:     Apply drift towards mean: $\mathbf{v}' \leftarrow \mathbf{v} + \lambda(\bar{v}I - \mathbf{v})$
10:     Update super-threshold elements: $\mathbf{u}'_{\mathcal{I}} \leftarrow \mathbf{v}'$
11:     Preserve sub-threshold elements: $\mathbf{u}'_{\{1,\dots,n\} \setminus \mathcal{I}} \leftarrow \mathbf{u}_{\{1,\dots,n\} \setminus \mathcal{I}}$
12:     return $\mathbf{u}'$
13: 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.

3.3 Modulators

The modulator class is defined through a collection of attributes common among all modulators. These attributes include the dimensionality of the modulator, the inputs it receives, the range of values it can take, the action it performs, and the rule that governs its dynamics. Further, almost all modulators relies on an internal leaky variable that is updated at each time step. The dynamics of this variable are described as $\tau \dot{\mathrm{v}} = \mathrm{E} - \mathrm{v} + \mathrm{I}_{\mathrm{ext}}$. The parameters τ and E are specific to each modulator, as well as the type of external input they are sensitive to and the dimensionality of the variable v itself.

Acetylcholine [ACh]
Function: modulate the speed of formation of new place cells by modulating the intesity of the weight update.
Inputs: $\max_{\mathrm{i}} \sum_{\mathrm{j}} \Delta \mathrm{W}^{\mathrm{ff}}_{\mathrm{i,j}}$
Output: $\mathbb{1}(\mathrm{v} > \theta_{\mathrm{ACh}}) \in [0, 1)$
Action: multiplicative term in the weight update rule

Dopamine [DA]
Function: represent a positive valence in a given input position or region, binding place cells with the current reward value
Inputs: $\mathrm{R} \in \{0, 1\}$; $\mathrm{u} \in [0, 1]^{\mathrm{N}}$
Output: $(\mathrm{W}^{\mathrm{DA}} \cdot \mathrm{v}) \in \mathbb{R}^{\mathrm{N}}$
Action: additive term in the forward pass
Learning: the weights W^{DA} are updated according to the reward signal R (filtered through $\mathrm{v} = \dot{\mathrm{v}}(\mathrm{R})$) and the eligibility trace u. There is hebbian potentiation (LTP) $\Delta \mathrm{W}^{\mathrm{DA}}_{+} = \eta(\mathrm{v} \cdot \mathrm{H}(\mathrm{u}^{\mathrm{T}}))$ and homeostatic depression (LTD) $\Delta \mathrm{W}^{\mathrm{DA}}_{-} = \eta((1 - \mathrm{v}) \cdot \mathrm{H}(1 - \mathrm{u}^{\mathrm{T}}))$, where H is the Heaviside step function.

Boundaries [Bnd]
Function: it represents a negative valence in a given input position or region, binding place cells with the current collision value
Inputs: $\mathrm{C} \in \{0, 1\}$; $\mathrm{u} \in [0, 1]^{\mathrm{N}}$
Output: $(\mathrm{W}^{\mathrm{Bnd}} \cdot \mathrm{v}) \in \mathbb{R}^{\mathrm{N}}$
Action: additive term in the forward pass
Learning: the weights $\mathrm{W}^{\mathrm{Bnd}}$ are updated according to the collision signal C (filtered through $\mathrm{v} = \dot{\mathrm{v}}(\mathrm{C})$) and the eligibility trace u. There is hebbian potentiation and homeostatic depression like for dopamine.

Velocity Modulation [Vel]
Function: it represents the intensity of the change in position $\frac{\mathrm{d}\mathbf{x}}{\mathrm{dt}}$ (*i.e.* the derivative or velocity)
Inputs: $\mathbf{x} \in \mathbb{R}^2$
Output: $\mathrm{relu}_{\theta_{\mathrm{Vel}}}(|\mathbf{x} - \mathrm{v}|_1) \in \mathbb{R}$