Training Spiking Neural Network with Reinforcement Learning

黄越 张卓豪

1. Introduction

The brain has a collection of high-order functions, guiding us to feel, act and learn. However, the brain is so complex that we know little about how brain learn new functions. Recently, *in vitro* biological neural network (BNN) was shown to be able to play Pong within a close-loop training system[1]. Previous to this work, scientists used BNNs to perform some easier tasks such as moving a cursor in a certain direction[2]. In these works, they all used multielectrode arrays to stimulate and record from the cells. Signals from recorded cells are used to manipulate a cursor or a paddle. Feedback from the task or pong game will be converted to electrical signal to stimulate the cells via electrodes. If the task is done well, "reward signal" would be delivered to the cells and otherwise "penalty signal". The neural network is believed to change its synaptic connectivity according to the feedback signal and perform better in later trials.

The choice of feedback signal is a little tricky. Some work[1] believed neural network would obey the free energy principle[3]. This theory indicates that the neural network would minimize the difference between its predictions about the environment and observed sensations (prediction error). Therefore, they used regular stimuli as "reward signal" and noise as "penalty signal". Others used "patterned training stimuli" as "penalty signal" to induce network plasticity and shuffled background stimulation as "reward signal" to stabilize accumulated plasticity[2]. The applied "patterned training stimuli" would induce network plasticity, but how much the plasticity is and the result from the plasticity is largely unknown and unpredictable.

According to different purposes, these two experimental studies were pre-designed with a mode of feedback signals respectively. However, it is not clear whether a more appropriate feedback mode exists. Thus, a better strategy to derive "reward signal" and "penalty signal" is needed.

Based on the experiments[2], computational models were derived and similar results were observed using spiking neural network with spike-timing dependent plasticity[4]. Here, we try to use the framework of reinforcement learning to find the optimal "reward signal" and "penalty signal" for a given spiking neural network.

2. Methods

2.1 Network

We used the Neural Circuit SIMulator (http://www.lsm.tugraz.at/) to simulate our networks. The network consisted of 1000 leaky integrate-and-fire neurons, with ~50000 synapses. 30% of the synapses are inhibitory synapses, the weight of which were fixed at -2.03e-7. The other synapses are excitatory and obeyed spike-timing-dependent plasticity, and the weight of the synapses would be modulated by the firing of the connected neurons. These neurons were randomly placed in a 3mm x 3mm area, where 60 electrodes were uniformly placed with equal intervals (Figure 1). Each electrode recorded 5 closest neurons and could stimulate 76 closest neurons.

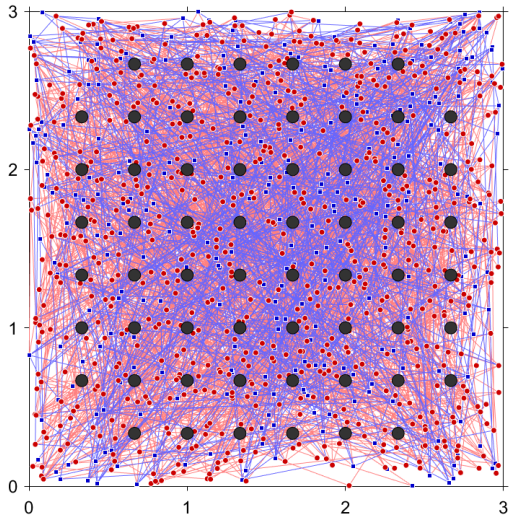

Figure 1. Simulated network. The black circles mark the 60 electrodes. Small red circles mark excitatory neurons and small blue circles mark inhibitory neurons. The red lines mark excitatory connections and blue lines mark inhibitory connections.

2.2 Task

The goal for the animat is to keep itself inside the circle. In one step, if the animat made a desired movement, then a random background stimulus (RBS) would be delivered through the electrodes (Figure 2). Otherwise, a patterned training stimulation (PTS) would be delivered. After RBS or PTS, based on which quadrant the animat was in, the network would receive a CPS which contains the information about the quadrant, and then we would record the activity of the network. Center of activity (CA) would be calculated from the recordings, and the final movement would be figured out. When the animat reached outside the outer circle, it would be randomly placed back to the inner circle.

2.2.1 Stimulus

All stimulus consisted of pulses. The amplitude of the pulse was 200 nA and each pulse lasted 20 ms.

For RBS, the pulse was delivered randomly at one of the 60 electrodes with inter-pulse-intervals ranging from 200 ms to 400 ms.

For CPS, four CPS sequences (CPS_{Q1}, CPS_{Q2}, CPS_{Q3}, CPS_{Q4}) were used. Only one CPS was delivered based on the location of the animat. Each CPS consisted of a sequence of 3 pulses from 3 randomly selected electrodes with inter-pulse-intervals ranging from 200 ms to 400 ms. The four CPS sequences were manually chosen in order to make the CAs point to different angles, and the last electrode in the sequence is termed “probe”. CPSs were fixed after well chosen.

For PTS, four pools of PTS (PTS_{Q1}, PTS_{Q2}, PTS_{Q3}, PTS_{Q4}) were used. The PTS was delivered based on the location of the animat and the training algorithm. Each pool of PTS consisted of 660 stimuli. A PTS consisted of repetitive stimulation at two electrodes. The location of the first electrode was chosen as the probe electrode used in the preceding CPS. The location of the first electrode could be chosen from one of 60 electrodes. And the interval of two stimulus could be chosen from 11 values: -100, -80, -60, -40, -20, 0, 20, 40,

60, 80, and 100 ms. Therefore, each PTS pool has 60*11 = 660 PTSs.

2.2.2 Computation of the animat's movement

After delivering CPS, we recorded the network activity for 100 ms with 60 electrodes. We counted the spike number at each electrode and calculated the firing rate (FR). CA was then computed with the following equation.

$$CA = [CA_X, CA_Y] = \frac{\sum_{k=1}^{60} FR(k) \cdot [Col(k) - 4.5, Row(k) - 4.5]}{\sum_{k=1}^{60} FR(k)}$$

CA is a 2-dimension vector. Col(k) means the column number of electrode k. Row(k) means the row number of electrode k. Then the movement [dX, dY] is computed:

$$[dX, dY] = \hat{T} * CA$$

where $\hat{T}$ is a manually predefined 2x2 matrix, which transformed CA into the movement of the animat. The transformations $\hat{T}$ for each CPS were defined so that the average movement in each quadrant would be the desired movement. For example, the desired movement in quadrant 1 is $[\sqrt{2}, \sqrt{2}]$.

2.3 Training rules

The training rule used in the paper is as follows (from Chao et al., 2008): if the animat's performance was desirable (moving inward), then RBS was delivered for 5 seconds until the next sensory input was evaluated (5 to 2 in Figure 2A). If the animat's performance was not desired (moving outward), then training was applied (5 in Figure 2A): a PTS was randomly selected from the corresponding pool; if the previous CPS was CPS_{Q1}, then the PTS was selected from PTS_{Q1} (6 in Figure 2A) and delivered for 5 seconds (2 in Figure 2A). If the performance of the animat was improved but still not desirable after the PTS (still moving outward but at a slower rate), then the same PTS would be used for the next training. Initially, the probability of choosing a PTS from a pool was identical (1/660). Every time a PTS improved the performance of the animat after the next probe, a copy was added into its pool. Thus, the size of the pool increased, and the probability of this ''favorable'' PTS being chosen later was increased. In contrast, if that PTS worsened the performance of the animat (moving outward faster), it was removed from the pool, unless only one PTS of this specific type remained.

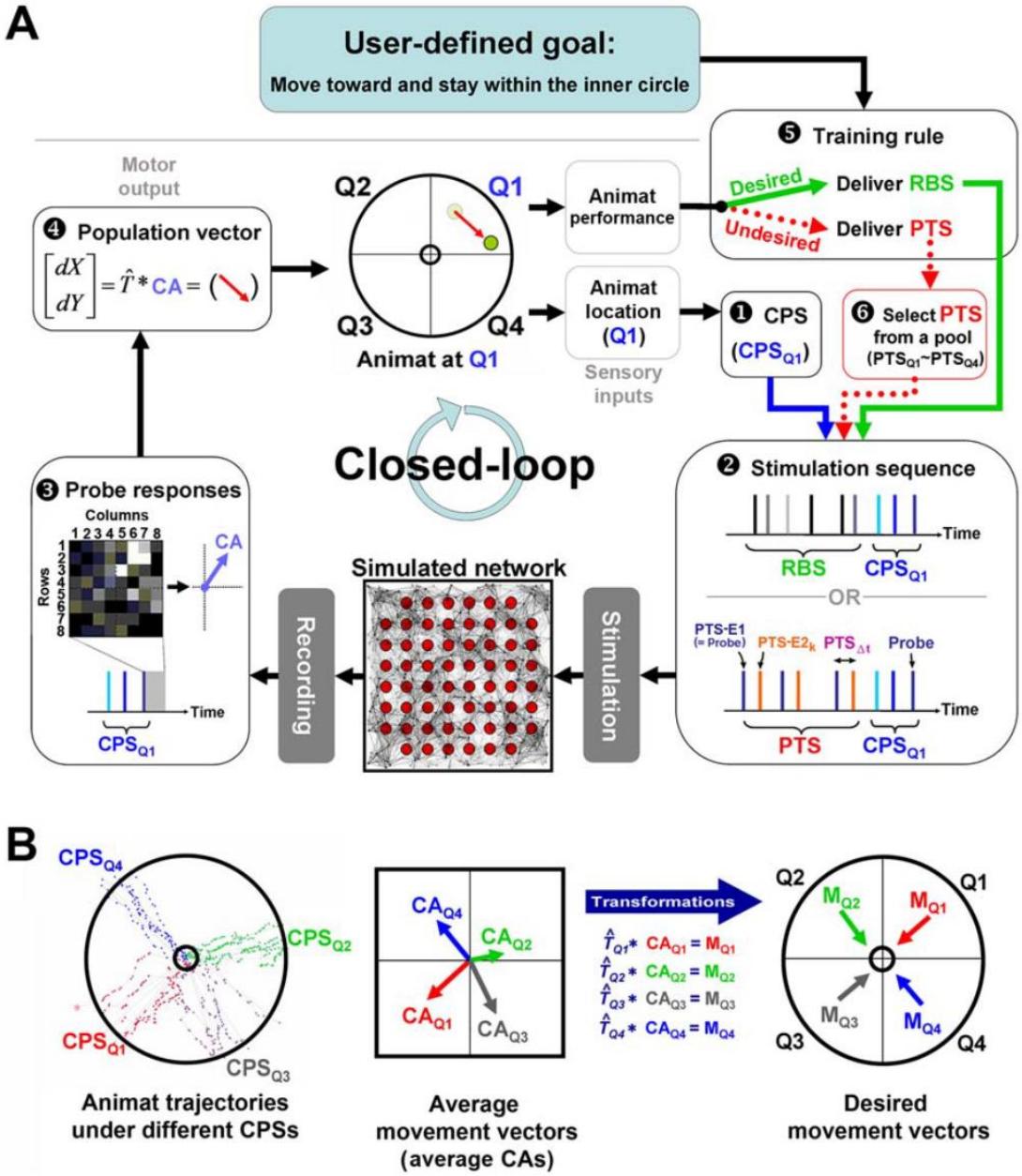

Figure 2. Closed-loop algorithm (from Chao et al, 2008). (A) Closed-loop design: the sensory mapping (1–2), the motor mapping (3–4), and the training rules (5–6). Refer to Methods for a detailed explanation. (B) Motor mapping transformation. Left: In the beginning of each experiment, each CPS (CPS_{Q1}–CPS_{Q4}) was continuously delivered every 5 seconds with RBS in between. After the animat reached the outer circle, it was moved back to the inner circle. Middle: The average CAs from probe responses to each CPS were calculated (CA_{Q1}–CA_{Q4}). The average CAs represent the average movements from each CPS. Right: The transformation (T_{Q1}–T_{Q4}) for each CPS was created so that the average movement in each quadrant would be the desired movement with a magnitude of 1 unit (M_{Q1}–M_{Q4}).

Here, we used ε-greedy algorithm to deliver the appropriate PTS. We can see this problem as four independent multi-arm bandit problems. The number of arms is 660. In a

single step, we use ε-greedy algorithm to deliver a PTS. Then we would receive a reward with respect to the angle between the animat's movement and desired direction (Figure 3). The Q matrix is updated based on the reward. The detail of the training rule is shown here:

ε-greedy training rule

Initialize, for $a = 1$ to 660:

$Q(a) \leftarrow 0$

$N(a) \leftarrow 0$

Loop forever:

$$A \leftarrow \begin{cases} \text{argmax}_a Q(a) & \text{with probability } 1-\epsilon \\ \text{a random action} & \text{with probability } \epsilon \end{cases}$$

$R \leftarrow PTS(A)$

$N(A) \leftarrow N(A) + 1$

$Q(A) \leftarrow Q(A) + [R - Q(A)]/N(A)$

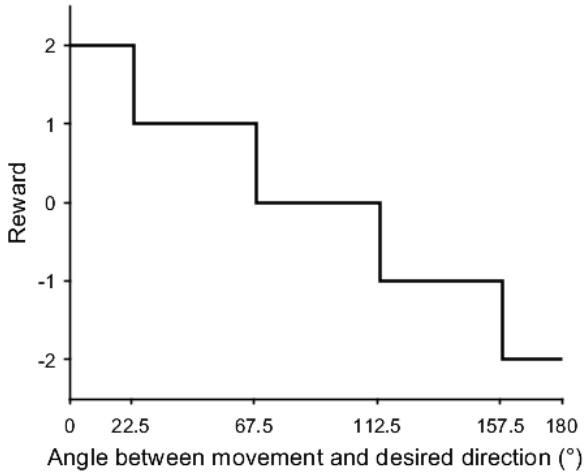

Figure 3. Reward of the ε-greedy training rule. More reward would be given when the angle between the animat's movement and desired direction is smaller.

3. Results

We trained the network for 1 hour and recorded the behavior of the animat. During the first 10 minutes, the motor mapping was unchanged so that the animat is expected to move towards the center of the circle. After 10 minutes, we exchanged the motor mapping of quadrant 1 and quadrant 3, and the animat is expected to move outwards in quadrant 1 and quadrant 3, but move inwards in quadrant 2 and quadrant 4. We delivered PTS using different training rules and checked if the network showed adaptation to the new motor mapping.

To test whether the network could adapt to new motor mapping without PTS, we trained it without PTS for 1 hour (Figure 4A). The animat went outside the outer circle from quadrant 1 or 3 at a high frequency. Using the training rule described in the paper, the animat moved outward more slowly (Figure 4B). And the network showed best adaptation to the changed motor mapping with **ε**-greedy training rule (Figure 4C). The animat did not

reach outside the outer circle after ~15 minutes of training.

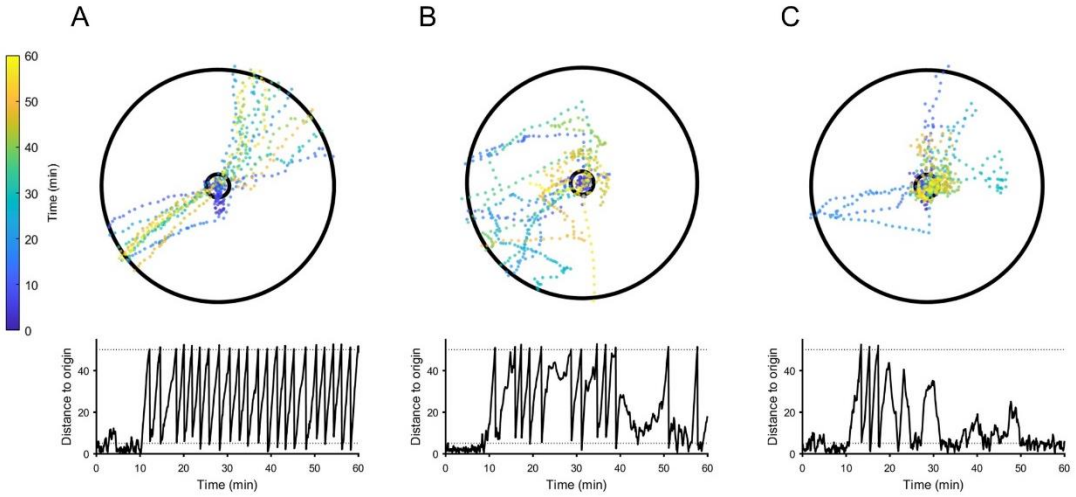

Figure 4. Training with different training rules. (A) Training without PTS. Top: the animat trajectory during the 60-minute training. Bottom: the distance to origin during training. During first 10 minutes, the sensory map was unchanged and the animat did not go outside the outer circle. After the sensory map changed, the animat went outside from quadrant 1 or 3 at a fast rate. (B) Training with the training rule described in Chao et al., 2008. The animat showed adaptation to the new sensory mapping. (C) Training with **ε**-greedy training rule. The animat adapted to the new sensory mapping very fast and did not reach outside after ~15 minutes.

4. Discussion

In this work, we designed a training rule to train a spiking neural network better than the training rule used in previous paper. Why did it outperform the previous training rule? We have the following possible answers: (1) The previous training rule abandoned the PTS that worsened the performance, but the PTS might be beneficial to the network in later trials. (2) We considered a more sophisticated reward based on current direction and desired direction, while the previous rule only considered whether the PTS made the performance better or worsened.

The PTSs in this work only considered the electrodes where the stimulus is delivered and the interval of two pulses. In fact, more stimulus parameters could be taken into account, such as the amplitude of stimulus and the duration of stimulus. In addition, we may not discretize the "inter-pulse interval" to make a continuous and larger action space.

Although we regarded the problems as a multi-arm bandit problem, it is actually different from the classical multi-arm bandit problem. The environment is non-stationary so that we would expect to receive different reward even if we made the same action. However, the **ε**-greedy training rule worked well here. It may be explained by the complex behavior of the neural network. Besides, the PTS may have a lasting effect to the network. Therefore, we could update Q matrix with the return or discounted return of an action instead of a single-step reward. With this new setting (infinite-horizon task), more sophisticated algorithms could be used.

Reference

1. Kagan, B.J., Kitchen, A.C., Tran, N.T., Habibollahi, F., Khajehnejad, M., Parker, B.J., Bhat, A., Rollo, B., Razi, A., and Friston, K.J. (2022). In vitro neurons learn and exhibit sentience when embodied in a simulated game-world. Neuron. 10.1016/j.neuron.2022.09.001.

2. Bakkum, D.J., Chao, Z.C., and Potter, S.M. (2008). Spatio-temporal electrical stimuli shape behavior of an embodied cortical network in a goal-directed learning task. J. Neural Eng. *5*, 310–323. 10.1088/1741-2560/5/3/004.

3. Friston, K. (2010). The free-energy principle: a unified brain theory? Nat Rev Neurosci *11*, 127–138. 10.1038/nrn2787.

4. Chao, Z.C., Bakkum, D.J., and Potter, S.M. (2008). Shaping Embodied Neural Networks for Adaptive Goal-directed Behavior. PLOS Computational Biology *4*, e1000042. 10.1371/journal.pcbi.1000042.