Problem Set 4

Due Jan 3, 2024

Happy Holidays!

A toy model of neural integrator

An animal moves on a one-dimensional track. In order to navigate properly (e.g., to find its way back home) the brain must compute the 'on line' position of the animal given sensory information about its velocity. We will assume that the position is computed by means of a network composed of two neurons, whose input is the animal's velocity. The network obeys the following equations:

$$\begin{aligned} \tau \frac{dx}{dt} &= -ax - 2y + \tau V(t); \\ \tau \frac{dy}{dt} &= -(3-a)x - y - \tau V(t) \end{aligned} \tag{1}$$

where the value of $x(t)$ represents the estimated position of the animal at time t.

(a) Study the dynamics of the network for constant velocity, i.e., $V(t) = V_0$ for all $t > 0$: First, find the fixed point of the network and the region of values of the parameter a for which this fixed point is stable. Next, write down the solution for $x(t)$ given that $(x(0), y(0)) = (0, 0)$ and explain what happens as time increases.

(b) Position Estimation. Assume that at $t = 0$, the animal starts out at the origin, and begins moving at a constant velocity $V(t) = 0.1$ m/s. Use $\tau = 100$ ms. If the system acted as a perfect integrator of the velocity, the expected position at $t = 10$ s would be 1 m. Is there a parameter choice (for parameter a) for which this system acts as a perfect integrator? (If so, take any appropriate limits to prove it.) For what range of parameters does the system act as a leaky integrator, i.e., the readout is within 1 cm error of the estimated position from a perfect integrator after 10 seconds? What happens to $x(t)$ when a is outside this range?

Hopfield model

In the Hopfield network, we make even a more drastic assumption that the activity of a neuron is a binary variable, $s_i \in [+1, -1]$. In the deterministic version of this model, the state of a cell is set according to the following equation

$$s_i(t + \Delta t) = \mathbf{sgn}(\sum_j w_{ij} s_j), \tag{2}$$

where the recurrent weight matrix is specified by the memory patterns $\xi^\mu, \mu = 1...P$ to be stored in the network:

$$w_{ij} = \frac{1}{N} \sum_{\mu=1}^{P} \xi_i^\mu \xi_j^\mu \tag{3}$$

We assumed that individual components in a memory pattern is a random variable $\in [+1, -1]$ such that different memory patterns are not correlated with each other.

(1) To make this argument more precise, consider random binary vectors $\mathbf{v}$ in an N-dimensional space: $\mathbf{v} \equiv [v_1, v_2, ..., v_N]$, where each $v_i = \pm 1$ is chosen independently and at random. The angle ψ between two such vectors is defined in the usual way by normalizing the dot product:

$$\cos \psi = \frac{1}{N} \mathbf{v}^{(1)} \cdot \mathbf{v}^{(2)} \tag{4}$$

Calculate the mean and variance of $\cos \psi$.

(2) Show that when $P \ll N$, ξ is the fixed point of the Hopfield model.

(3) Check my lecture note and show that any mixture of an odd number of memory patterns, such as

$$\xi^{mix} = \mathbf{sgn}(\xi^{\mu_1} + \xi^{\mu_2} + \xi^{\mu_3}) \tag{5}$$

are fixed point of the Hopfield model.

(4) We consider an asynchronous update rule for each neuron (choosing either a random or fixed update order and update s_i according to that order at each clock cycle). Numerically simulate the dynamics of Hopfield network with random unbiased memory patterns, using 500 binary neurons. We want to draw some statistical conclusions, so in the tasks that follow, you should run many simulations using different sets of random stored patterns and, for each, start from many different initial conditions (so I recommend writing reasonably efficient routines, e.g. minimize the number of for-loops in MATLAB). For simplicity, you can choose the order of update of the neurons by going through neuron 1 to N all the time.

i) Probability of perfect recall: Store 10 patterns, run simulations to find the probability of perfect recall. That is, start with a pattern, corrupt some fraction of its bits, input the corrupted pattern to the network, and wait until the network settles into a fixed point. Plot the fraction of times you get perfect recall of the original state against the fraction of corrupted bits.

ii) Memory retrieval with errors: Start the dynamics with one of the memory states. Plot the fraction of errors (fraction of incorrectly recalled bits) in the final state as a function of the number of stored patterns P. Change P from 10 to 100.