Excercise 1:

Contents

question

Day 3, Exercise # 1: BCM

Implement a simulation of the competition between two input patterns $\mathbf{x}_1 = (20, 0)$ and $\mathbf{x}_2 = (0, 20)$. At each timestep, one of the two patterns is presented to the neuron, the pattern is chosen randomly with a probability 0.5 of being pattern $\mathbf{x}_1$ and 0.5 of being pattern $\mathbf{x}_2$. The weights follow the BCM rule. The output of the neuron at each timestep is given by

$$y(t) = \mathbf{w}^T \mathbf{x}(t), \tag{1}$$

where $\mathbf{x}(t)$ is the input presented at time t.

Recall from the lecture that the weight update for the BCM rule is:

$$\frac{d}{dt}\mathbf{w} = \eta \mathbf{x} y (y - \theta), \tag{2}$$

where θ is the sliding threshold: $\theta = \frac{\langle y^2 \rangle}{y_0}$. Note that the average $\langle y \rangle$ must be computed online. The threshold will therefore obey the rule:

$$\tau \frac{d}{dt}\theta = -\theta + \frac{y(t)^2}{y_0} \tag{3}$$

Implement this simulation using a total time $T = 10$ s, $\eta = 10^{-6}$ ms^{-1}, $y_0 = 10$, $\tau = 50$ ms. Use the Euler method with a timestep of 1ms. You should also put a hard bound for the weights at 0 (when $w_i < 0$, set it to 0).

Plot the weights $\mathbf{w}$, the sliding threshold θ and the output y. What do you expect to see? Run the code a few times. What do you see?

Thanks to Dr. Claoudia Clopath who proposes this problem in CCCN Course

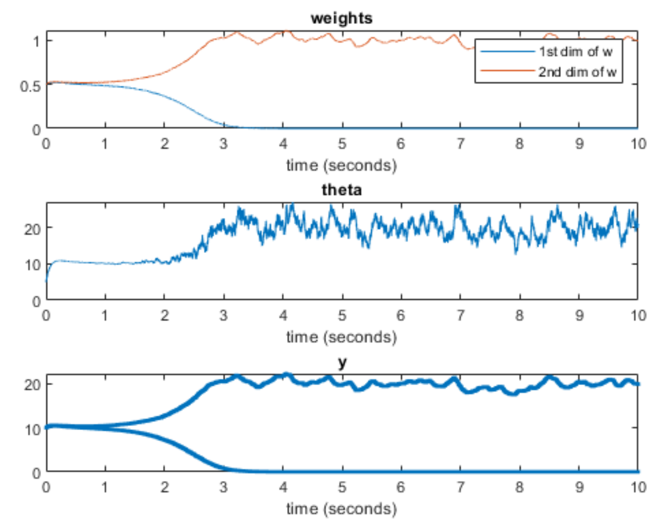
The Bienenstock–Cooper–Munro (BCM) learning rule provides a simple setup for synaptic modification that

combines a Hebbian product rule with a homeostatic mechanism that keeps the weights bounded.

Answer

```
clc
clear
close all

T = 10; %total time in seconds
etha = 10^(-6); %in ms^-1
y0 = 10;
Taw = 50; %in ms
dt = 1; %in ms
total_data_points = T*1000/dt;

y = zeros(1, total_data_points);
w = zeros(2, total_data_points);
w(:,1) = [0.5; 0.5];
theta = zeros(1, total_data_points);
theta(1) = 5;

x1 = [20; 0];
x2 = [0; 20];

for i = 1:total_data_points
        z = rand;
        if z > 0.5
          x_stim = x1;
        else
          x_stim = x2;
        end

        y(i) = transpose(w(:,i)) * x_stim;
```

```
        w(:,i + 1) = w(:,i) + dt*(etha*x_stim*y(i) * (y(i) - theta(i))) ;

        if w(1,i) < 0
            w(1,i) = 0;
        end
        if w(2,i) < 0
            w(2,i) = 0;
        end

        theta(i + 1) = theta(i) + (dt/Taw)*(-theta(i) + (y(i)^2)/y0 );
end
y(total_data_points+1) = y(total_data_points);

time = linspace(0,T,total_data_points+1);

figure
subplot(311)
plot(time, w(1,:)),   hold on,        plot(time, w(2,:)),      title('weights'),     xlabel('time (seconds)'),        legend('1st dim of w', '2nd dim of w')

subplot(312)
plot(time, theta), title('theta'),     xlabel('time (seconds)')

subplot(313)
plot(time, y,'.') , title('y'),     xlabel('time (seconds)')
```



Result:

```
disp('')
```

When we run the code multiple times, we will see that every time each of the patterns wins randomly

Also Y follows the winner and loser pattern in seperate ways. That's why we plotted y with '.' notation

We can add a bias to one of the patterns by changing the 0.5 probability

Published with MATLAB® R2023a