Practical 6: Ben-Yishai network of visual cortex

Neural Computation, Mark van Rossum
(after Dayan and Abbott exercise 7.1)

In this practical we discuss the Ben-Yishai model of orientation tuning in the primary visual cortex (PDF of original paper on website). We consider a network of rectifying neurons with feed-forward visual input and recurrent (lateral) connections. The neurons are modelled with a firing rate.

We label the neurons v with the angular variable $\theta = -\pi/2...\pi/2$. The neurons receive direct input h and input from recurrent connections, proportional to v itself. The firing rate evolves as:

$$\tau \frac{dv(\theta)}{dt} = -v(\theta) + [h(\theta) - T + \sum_{\theta'} M_{\theta,\theta'} v(\theta')]_+ \quad (1)$$

$$= -v(\theta) + [h(\theta) - T + \int_{-\pi/2}^{\pi/2} \frac{d\theta'}{\pi} \{-\lambda_0 + \lambda_1 \cos[2(\theta - \theta')]\} v(\theta')]_+ \quad (2)$$

where λ_0 describes the strength of uniform inhibition. The λ_1 describes the strength of tuned excitation (or inhibition depending on $\theta - \theta'$). The T denotes the threshold. Sketch for yourself the total recurrent strength as a function of $\theta - \theta'$.

The input h is written as

$$h(\theta) = Ac[1 - \epsilon + \epsilon \cos(2\theta)]$$

where A is the stimulus amplitude, c the stimulus contrast, and ϵ the input anisotropy: if $\epsilon = 0$ the input is flat, if $\epsilon = 1/2$ (its maximum value) the input is strongly tuned. Sketch this function for yourself. [1]The goal of the network is to see how lateral interactions change the output of this single layer network.

1 Simulation

a) Create an array of 100 neurons. Create an array for the input h as well.
Fill the input array h according to the above equation. Fix $A = 50$ for the rest of the practical. Take $c = 1.5$ and $\epsilon = 0.2$. Plot your input array, to make sure all is correct. A useful trick is:

```
th=(-pi/2,pi/100,pi/2-pi/100);
% creates array with 100 elements from -pi/2 ...  pi/2-pi/100
h = ...  cos(2*th);
```

b) Next we need code that calculates the integral, Eq.1, for given activity v. We can speed up things a lot by writing the integral as a matrix multiplication and calculating the weight-matrix M beforehand. To do this write the above equation as a matrix:

[1]In the lecture notes pg. 40 it is written that $\tau \frac{dv}{dt} = -v + [W.u + M.v]_+$. Here we write $h = W.u$.

`input = max(0,h+mat*v-T);`

Take care with the normalisation: $\int_{-\pi/2}^{\pi/2} \frac{d\theta}{\pi}$ becomes $\frac{1}{n}\sum_{i=1}^{n}$.

c) Take $\lambda_0 = 5$ and $\lambda_1 = 0$, $T = 25$. Start with the initial condition $v = 0$ and simulate the evolution of v over time. (See the integrate-and-fire model for time integration. You can take τ your lucky number, but take your time step equal or less than τ). Plot the activity of v as it evolves, and explain what you see. You can write a little routine that checks whether the steady state is already reached (see below).

Usually some 100...1000 iterations should be enough.

d) Take $\lambda_0 = 5$, $\lambda_1 = 0$, $\epsilon = 0.1$. This means that there is uniform recurrent inhibition. Vary the contrast c (range 0.1 to 10) and observe the steady state. You will see three regimes: no output, a rectified cosine, and a cosine plus offset.

e) Next, take a small value for ϵ, take $\lambda_0 = 2$, and vary λ_1 from 0 to 4. For which value of λ_1 do 'rectified' solutions appear? Are the output widths invariant across contrast c?

If everything went well, you should have found that for larger λ_1 tuning curve width does not depend on contrast, in contrast to your answer in d). In other words, tuned inhibition can produce contrast invariant tuning curves, uniform inhibition can not. Furthermore, even for $\epsilon = 0$ the output will be tuned although the input is flat. The attractor state is a line, or more accurately a ring attractor.

f) The dynamics in this model are interesting. After steady state has been reached, move the input profile $h(\theta) = Ac(1 - \epsilon + \epsilon cos(2\theta + 2\theta_0))$. How and how fast does the new equilibrium establish? Try both $\lambda_1 = 0$ and small ϵ separately. Try to explain your result qualitatively.

2 Mathematical approach

Take $T = 0$ (this does not change the essence of the model). In steady state the activity now satisfies

$$v(\theta) = [h(\theta) + \int_{-\pi/2}^{\pi/2} \frac{d\theta'}{\pi}[-\lambda_0 + \lambda_1 \cos(2(\theta - \theta'))]v(\theta')]_+$$

Assuming that $v(\theta)$ is symmetric about $\theta = 0$, $v(\theta)$ either takes the form

$$v(\theta) = \alpha[\cos(2\theta) - \cos(2\theta_C)]_+ \tag{3}$$

or else,

$$v(\theta) = \alpha \cos(2\theta) + v_0 \tag{4}$$

a) In case of Eq.3, show that for the steady state solution

$$\begin{aligned} \alpha &= \frac{Ac\epsilon}{1 - \frac{\lambda_1}{\pi}(\theta_c - \frac{1}{4}\sin(4\theta_c))} \\ cos(2\theta_c) &= \frac{\lambda_0}{\pi}[\sin(2\theta_C) - 2\theta_c \cos(2\theta_c)] - \frac{1-\epsilon}{\epsilon}[1 - \frac{\lambda_1}{4\pi}(\theta_c - \sin(4\theta_c)] \end{aligned} \tag{5}$$

b) Calculate α and v_0 in case that Eq.4 is valid.

c) We like to determine under which conditions there is always a solution of the form of Eq.3, no mattter ow small ϵ. Suppose that ϵ becomes very small, for Eq.5 this means that $[1 - \frac{\lambda_1}{4\pi}(\theta_c - \sin(4\theta_c)]$ needs to be close to zero. Why is that? What is the maximal value of $[\theta_c - \sin(4\theta_c)/4]$ on the interval $-\pi/2...\pi/2$? We want a solution with $\theta_c < \pi/2$, what does that mean for λ_1?

Test your result in the simulation.

3 Matlab script

Possible implementation:

```
n = 100;
dth= pi/n;

% parameters:  lam0 lam1 eps
lam0 = 5; % >=0
lam1 = 0; % >= 0
c = 1;
a = 50;
eps = 0.2; % 0 ...  0.5

% create input
h=zeros(1,n);
for i=1:n
  h(i)=a*c*(1-eps+eps*cos(2*(i*dth+pi/2)) );
end

mat = zeros(n,n);
th_ar = (-pi/2:pi/100:pi/2-pi/100);
for i=1:n
  thp=(i-1)*dth-pi/2;
  mat(i,:)= (-lam0+lam1*cos(2*(th_ar-thp)))/n;
end

v=ones(n,1);
irep=0;
err=1;
err_eps = 0.003;
nrepmax = 1000;
thr= 25;
while (err > err_eps & irep < nrepmax)
  in = max(mat*v + h'-thr,0); % rectifty
  vold = v;
  v = v+0.2*(-v+in); % integrate
  err = mean(abs(vold-v));%/mean(v);
  irep = irep+1;
  plot (v);
  hold on;
end

if (irep == nrepmax)
  'nrepmax reached !'
end

plot(v);
hold on
```