Assignment 2: The BCM rule (v2.0)

Neural Computation 2010-2011

23rd December 2010

Simulation framework

In this exercise we study the Bienenstock, Cooper and Munro (BCM) rule of synaptic plasticity. This Hebbian plasticity rule has a sliding threshold, which is intended to prevent run-away excitation in the post-synaptic cell. We will assume that all units simulated are rate units, i.e. using a single floating-point activation level rather than modelling individual spikes. In each time step, an input activity pattern $\mathbf{x}$ (a vector of floating-point values) is provided for the set of presynaptic units, and the firing-rate activity level of a post-synaptic unit y is calculated using a dot product of the input pattern and a vector of weights $\mathbf{w}$, i.e., $y = \mathbf{wx}$. For simplicity, the post-synaptic unit is fully linear, and both weights and activities are allowed to be negative. After each input presentation, the weight changes according to the rule:

$$\Delta w_j = \eta x_j y(y - \theta)$$

whereη is a learning rate, x_j is the firing rate of input j, $w_j(t)$ is the synaptic strength associated with that input at time t, $\Delta w_j = w_j(t+1) - w_j(t)$, and θ is a threshold determining the strength and sign of the weight change.

The threshold θ is set to the average of the squared activity, updated each time-step with $\Delta\theta = \frac{1}{\tau}(y^2 - \theta)$, where τ is a time constant determining how quickly the threshold is updated. Take $\eta = 0.01$ initially; for simplicity we work with dimension-less units throughout this assignment.

First, we study a very simplified case in which there is just one input x_1 with a fixed initial rate $x_1 = f_0$.

Question 1 (10 points) Suppose that equilibrium is reached such that $\Delta w_1 = 0$ and θ is constant. Given that the input rate is f_0, calculate (without simulation) the value of the equilibrium synaptic weight $w_{1,\infty}$, i.e. the weight value for which the change is zero, and the value of the equilibrium threshold θ_∞. Be sure to show all the important steps in your calculation and explain anything not obvious.

Question 2 (10 points) Simulate the conditions in question **1**, taking $\tau = 10$, $f_0 = 0.5$, $w_1 = w_{1,\infty}$ (i.e., set the initial weight to your calculated equilibrium value), and $\theta = \theta_\infty$(i.e., set the threshold to your calculated equilibrium value), and simulate for 10,000 time steps. Verify that the system is initially in equilibrium, and test the ability of the system to return to equilibrium after a disturbance d by increasing the input rate by 50% (let d=1.5; $x_1 = df_0$) 1/3 of the way through the simulation, and then returning it to the original value ($x_1 = f_0$) for the final 1/3 of the simulation. Plot the output rate and the synaptic weight versus time to demonstrate both the initial equilibrium and the behavior in response to the disturbances. Discuss the shape of the curves – does the system respond the same to increases and decreases in the input activity level? Why or why not, and what are the implications?

Question 3 (5 points) Repeat the simulation and plots from question **2** with $\tau = 200$. Explain the difference(s) in results and their implications. Consider other values of τ, f_0 (always using the appropriate calculated $w_{1,\infty}$value for that f_0), or the amount and direction (i.e., less than or greater than 1) of disturbance d, each time changing only one parameter from question **2**, and choose two of these regimes (of potentially many) that have interesting and qualitatively different patterns of behavior. Show and explain each of your chosen two regimes, describing why they behave differently and what implications that may have for the function of this system or the validity of this model. You may need to use more than 10,000 iterations in some cases to see the full behavior of the model, e.g. if you can see the system heading in one direction but it hasn't yet made it there over the time simulated.

For the rest of the simulations take $\tau = 10$.

Binocular input

We will now use the BCM rules to model the development of connectivity patterns to two eyes, such as the connections underlying ocular dominance columns, in an abstract but tractable way. Extend the model so it receives 24 inputs, 12 from the left eye, 12 from the right eye. The input patterns correspond roughly to spots of light with one of `npats` positions, along a 1-dimensional vector approximating a slice across the retina. In Matlab, the patterns are implemented as follows

```
npats=12;
nin=12;
pats=zeros(npats,nin);
for ipat=1:npats
    i=[1:nin];
    c=cos(2*pi/npats*(i-ipat));
    pats(ipat,:)=exp(-(1-c));
end
```

Spontaneous activity should be added to the patterns, modelled by uncorrelated Gaussian noise with a standard deviation of 0.1. Note that because the remaining problems involve simulating random processes, you will often need to do a few runs to be sure that the behavior you observe is not due to chance alone. Be sure to set the random seeds explicitly so that you can have reproducible results; otherwise you will get very confused. It is not necessary to run tests for statistical significance, as long as you do enough runs with different seeds to see how the results vary (and can report such variation, if appropriate for that question).

Question 4 (10 points) At each time step, pick one of the patterns at random, make two copies of the pattern (one for each eye), and add noise independently so that each eye will see a slightly different input. Starting from random synaptic weights, simulate the development of the weights. Plot and compare the initial and the final weight vector (with data from each eye plotted separately for clarity). Discuss your results, focusing especially on the binocularity of the inputs and any grouping that may be apparent, and relate the results to what you know of the processing for each eye in animals. Do you see evidence of anything like ocular dominance columns forming?

Model of strabismus

When strabismus is induced the inputs from the two eyes become uncorrelated.

Question 5 (5 points) To model strabismus, pick the patterns from left and right eye independently, again adding Gaussian noise independently. Plot the synaptic weights as before and explain your findings as before.

Model of monocular deprivation

Monocular deprivation (covering one eye) is a common experiment to study cortical development. We assume that activity in the covered eye is given by the spontaneous activity alone, with no other input pattern.

Question 6 (5 points) What would happen to a synaptic weight if its input would become zero? Discuss the realism of modelling the covered-eye input as noise, and also discuss the realism of using uncorrelated noise in particular. Simulate the situation described. Starting from a normally developed model, assume the right eye gets covered. Plot the evolution of the maximal weight from the left eye and maximal weight from the right eye. Vary the noise level, and explain the difference(s).

Recovery

Two varieties of recovery are commonly studied: 'normal recovery' in which both eyes are opened, and 'reverse suture' in which case the previously closed eye is opened, but the previously open one is closed.

To simulate deprivation and recovery within a reasonable time we first determine the final weights after monocular deprivation. Use noise standard deviation 0.1 and simulate monocular deprivation until the weights reach a steady state. Store the weights of the open eye.

To simulate recovery, start from the situation where the weights from the previously deprived eye are zero, and the weight from the previously open eye equal the stored weight vector. Set the standard deviation of the noise to 0.01 and use $\tau = 10$, $\eta = 0.01$. The value of $\max(w)$ can be used to evaluate the level of recovery.

Question 7 (5 points) Simulate both normal recovery and reverse suture for some 5000 time steps. Show that normal recovery leads initially to a faster recovery of previously depressed weights than reverse suture, but that later the reverse suture will be faster. Explain why this happens. How would normalisation models as discussed in the lecture notes §13.3 behave (no simulation necessary)?

General issues

To run your simulations, you may use Matlab, Python, or any other system approved with me beforehand, but note that all example code here uses Matlab syntax (e.g., with Fortran array order and numbering).

You will find that some questions are quite open-ended. A particularly well-researched answer can receive additional points, but core-dumping (just writing down all you can think of) will not. You must always include enough information and discussion to show the reader how you obtained your result (e.g. showing the mathematical calculation or plots from a numerical simulation), and to convince the reader that your result is correct. To be fully convincing, sometimes additional simulations other than those explicitly mentioned may be appropriate, in which case you should explain clearly why those were needed and what insight they give.

All plots must include axis labels and legends; they are nearly impossible for me to mark otherwise. Ideally, these will be shown directly on the plot; e.g. inkscape can be used to load just about any plot so that you can add arbitrary text if it is too awkward to do so in the original program.

Plagiarism

Copying results obtained by others is forbidden, and can lead to severe punishment (including the loss of your degree, even retroactively). It's fine to ask for help from your friends on technical issues. However, this help must not extend to copying or sharing code, results, or written text, nor to writing code or text together with a friend. If any students are found to have shared or copied work in this way, the University rules on plagiarism will apply to all parties involved.

Submission

Your work must be submitted by the specified time on the due date using the `submit` command on Informatics DICE machines (type `man submit` for more details). Your work should be in the form of one PDF file (*not* a Word .doc file!) for the assignment, plus separate .m or .py text files for each question that includes a simulation. For Matlab submissions, each file *must* be named as in this example `submit` command:

`submit msc nc cw2 asst.pdf q2.m q3.m q4.m q5.m q6.m q7.m <somefunction>.m`

where all files should be named exactly as written, except that *<somefunction>*`.m` should be replaced with the name(s) of whatever Matlab function(s) that you share between scripts. For Python submissions, the submit command would be:

`submit msc nc cw2 asst.pdf q2.py q3.py q4.py q5.py q6.py q7.py shared.py`

where all files should be named exactly as written. These guidelines ensure that I and the external examiners can find the appropriate file easily for each submission. As described in the standard Informatics MSc late policy (http://www.inf.ed.ac.uk/teaching/years/msc/courseguide10.html), late submissions will *not* be accepted without good reason, which must be arranged beforehand with the ITO, not the lecturer.

Getting a good mark

Be sure that you provide *evidence* that you did each part of this assignment. I can only judge what is actually submitted, so you should make sure that the files you submit make it clear that you have done everything requested, and thought about what each part means. This requirement means that you need to write English text for nearly every problem, not just showing mathematical or simulation results, but discussing their meaning and importance. Also please go over your assignment before you submit it, making sure that you addressed every question asked, so that you do not lose marks for omitted material.

It should not be necessary to consult scientific literature to answer these questions. If you do use information from outside the course material or your own experience, you must cite it appropriately.

Please read and follow my list of writing tips (http://homepages.inf.ed.ac.uk/jbednar/writingtips.html).