Advance Matlab

Exploring and Training Rate Models of Random Neural Networks

Merav Stern; OCNC 2018

Model

We wish to integrate the equation:

Here is the activity of neuron with where is the size of the network. is the rate of neuron . For simplicity we choose . The elements of the connectivity matrix are taken independently from a Gaussian distribution with zero mean and variance, with of the order . [1]

*Matlab*

(\*) Build and initial the network, relevant functions: *rand, randn*

(\*) Run (integrate) the network, relevant functions: *ode45*

Integrating chaotic systems is challenging.

(\*) compare the dynamics of the same connectivity (starting with the same initial conditions) that is achieved by using *ode* or a simple *for loop* (Euler integration). We will use the Euler integration in the next section.

(\*) Model: How does the parameter g affect the network behavior?

(\*) Explore the typical memory length as function of g, relevant functions: *xcorr (unbiased)*

Different modifications to the connectivity matrix were explored over the years, for example, restricting the sign of connections to include Dale's law. Here we wish to modify the connectivity to allow the dynamics to reach a nontrivial fixed point, for example by using the symmetric part of or adding an order 1 elements to the diagonal.

(\*) Modify and determine whether the dynamic reaches a fixed point  *ode-options:events*;   
 Hint: value = 1/(N^2\*var(J(:)))\*(-x +J\*tanh(x))'\*(-x +J\*tanh(x)) - (10^(-9));

Training

1. By batch:

We use the activity of the network to generate a desired output .

(\*) Run the network and find weights such that , relevant function: *pinv*

2. With feedback:

We alter the dynamical equation to follow:

(\*) Use the desired function for integrating the dynamics and repeat the batch solution. What happens when you continue to run the dynamics beyond the trained time?   
(\*) How does the internal activity change from previous step? Use PCA to distinguish between the solutions. Relevant function

3. Echo State:  
In order not to record activity , but rather to train "on the go" we can estimate the correlation matrix using recursive least square method.

(\*) Use the following code to update the readout weights during the integration:

k = P\*r;

rPr = r'\*k;

c = 1/(1+rPr);

P = P - k\*(k'\*c);

e = target(i) - z(i);

w = w + e\*k'\*c;

\*Initialize with P = I (the identity matrix).

4. FORCE:

(\*) Instead of using the desired output while learning, use the actual output . The algorithm will converge and the learning phase would bring itself to end when the output desired will be produced.