Network Models

Background

This chapter is a summary of chapter 7 from a book by Peter Dayan and Laurence F. Abbott called "Theoretical Neuroscience: Computational and Mathematical Modeling of Neural Systems"[1]. The book gives a mathematical basis for understanding what nervous systems do such as learning, memory, vision, neuron function and much more. While the book is divided into 3 parts, this is specifically a summary and explanation of the seventh chapter in the second part of the book.

- **Context/Biological Phenomenan Under Consideration**

While the book covers many different aspects including plasticity, memory ecoding and decoding, this chapter in particular covers the basics of neural networks and neural modeling and has many different applications both in neuroscience research and in computational aspects such as artificial intelligence and machine learning. The summary is specifically considering some of the models that were shown and discussed in chapter 7. While the book went into much deeper length about all of these and included other models, the first three of these are built upon each other while the other model, the stochastic neural model, can be looked at as similar in many regardds but offers a much different way to look at these neural networks.

History Of Neural Network Models

Neural networks have been studdied for quite a long time now but just recently are begining to be looked at from a quanitative perspective. The first attempt at using mathematics as a basis for these models was in the 1940's and was a simple mathematical model that used electrical circuits. The first multilayer model was made in the 1970s but not much research was done during this time due to computational limitations and various asundry reasons. Bidirectional connections between neurons and multilayered neuron models in the 1980's sparked much interest in the field and with the increasing availability of computers, people began to research increasingly complex networks which helped push multiple other fields from neuroscience to artificial intelligence.

Types of Models

- **Firing Rate Models**

This firing-rate model consists of two basic parts. The first is that its total synaptic input to a neuron (neuron u) depends on the firing rates of its presynaptic input neurons. The second part of this model is how the firing rate of post synaptic neuron (output v) is affected by the total synaptic input.

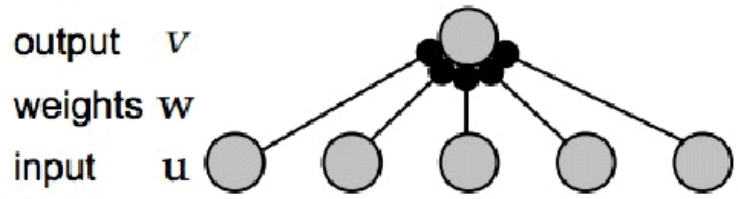

Figure 1.This Figure is an example of feed forward inputs u to the output v with synaptic weights w. Taken from Theoretical Neuroscience by Dayan & Abbott.

- **Parameters:**

$I_s$is the total synaptic current

N_u is the number of synaptic inputs (input u

Synaptic inputs labeled by b=1, 2, … , N_u

Firing rate of input b is u_b (input vector **u**)

synaptic weight is denoted as w_b (note that for excitatory synapses $w_b > 0$ and for inhibitory synapses $w_b < 0$)

synaptic kernel $K_s(t)$

Current in output neuron at time t is $w_b\, K_s(t)$

the neural response function $\rho_b(\tau) = \Sigma_i\, \delta(\tau - t_i)$

$F(I_s)$ is the firing rate function.

- **Assumptions:**

Effects of synaptic spikes sum linearly with large enough sample.

The synaptic kernel most often used is an exponential:

$$K_s(t) = \exp(-t/\tau_s)/\tau_s$$

- **Derivation of Firing-Rate Model:**

Total current at time t with spikes at presynaptic input b at times t_i is given by:

$$w_b \sum_{t_i<t} K_s(t - t_i) = w_b \int_{-\infty}^{t} d\tau\, K_s(t - \tau)\, \rho_b(\tau)$$

with no nonlinear interaction between different synaptic currents the equation becomes

$$I_s = \sum_{b=1}^{N_u} w_b \int_{-\infty}^{t} d\tau\, K_s(t - \tau)\, \rho_b(\tau)$$

We must now replace the neural response function $\rho_b(\tau)$ in equation (3) with, the firing rate of input b, $u_b(\tau)$. Therefore (3) becomes:

$$I_s = \sum_{b=1}^{N_u} w_b \int_{-\infty}^{t} d\tau\, K_s(t - \tau)\, u_b(\tau)$$

Taking the derivative of (4) with respect to t, and rearranging such that we are using the synaptic kernel equation from (1), we get:

$$\tau_s \frac{dI_s}{dt} = -I_s + \sum_{b=1}^{N_u} w_b\, u_b = -I_s + \boldsymbol{w}\cdot\boldsymbol{u}$$

with sum of $w_b\, u_b$ expressed as a dot product. Since (4) determines the current entering a post synaptic neuron, we must now determine the post synaptic firing rate from I_s.

The firing rate equation is derived from this using, a function F(**u**·**w**) where this is a function using a threshold and a half-wave rectification (i.e. the neuron must reach some activation potential for it to go above x-axis) and is the steady state output firing rate, becomes:

$$\tau_r \frac{dv}{dt} = -v + F(\boldsymbol{u}\cdot\boldsymbol{w})$$

using experimental and computational results though, it can be shown that this model (6) does not provide a completely accurate prediction of firing rate for all levels of current input and and thus a more complicated model is needed. An example of a more complicated model would be either a Feedforward Network or a Recurrent Network which is shown graphically by:

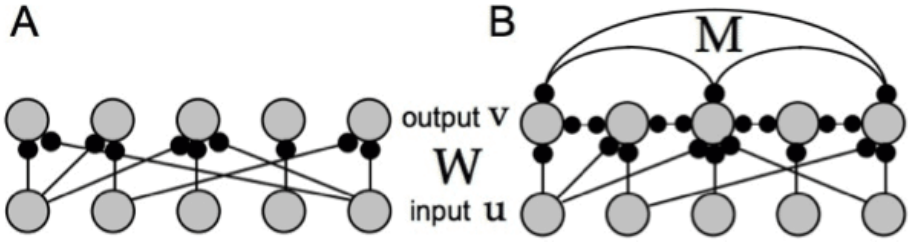

Figure 2Figure A is a feedforward network. Figure B is a recurrent Network. Taken from Theoretical Neuroscience by Dayan & Abbott.

- **Feedforward Networks**

A feedforward network has two layers of neurons and are connected by a matrix W. A feedforward model is only slithly more complicated (in that it is basically built upon the basic firing-rate model) and is arguably one of the most simple neural networks in that it does not include any loops or "backwords" directed synapses. Because of this, it is often not very practical in of itself to model anything but is useful as a basis to build other more complicated models upon that include loops and cycles such as with recurrent networks and excitatory-inhibitory networks. Using the formulation as before but with a matrix W rather than vector w to account for multiple outputs, the feed forward model can be written as:

$$\tau_r \frac{dv_a}{dt} = -v + F\left(\sum_{b=1}^{N_u} W_{ab}\, u_b\right) \text{ or } \tau_r \frac{dv}{dt} = -v + F(W\cdot u)$$

Now W represents the matrix of synaptic weights and has components W_{ab} which represent the strength from some input b to some output a.

- **Recurrent Networks**

In a recurrent network, there are two layers of neurons as well but the neurons in the output are interconnected with a matrix M. An output neuron a connected to some other element a' is denoted as matrix element $M_{aa'}$.

The model for the recurrent model is:

$$\tau_r \frac{dv}{dt} = -v + F(W\cdot u + M\cdot v)$$

These recurrent networks are also capable of forming patterns when they receive a complex Input which is not as possible with feedforward networks. These patterns can further be interpreted in a number of ways such as memory patterns and memory capacity. These different applications of recurrent neural networks have far reaching implications.

Under Dale's law, it is stated that a neuron cannot excite some postsynaptic neurons while inhibiting others, which means that the sign of all $M_{aa'}$ must be the same. Because of this, excitatory and inhibitory must be described differently and the book gives a set of equations which are respectively:

$$\tau_E \frac{dv_E}{dt} = -v_E + F_E(h_E + M_{EE} \cdot v_E + M_{EI} \cdot v_I)$$

$$\tau_I \frac{dv_I}{dt} = -v_I + F_I(h_I + M_{IE} \cdot v_E + M_{II} \cdot v_I)$$

- **Network Stability**

For a network that is experiencing constant input and in a steady state with dv/dt = 0, then the network is said to be exhibiting fixed-point behavior. While this is mostly what we have been dealing with so far, there are many different other physical events such as network oscillations and chaotic networks. Since under most conditions, a steady state will inevitably be met, a Lyapunov Function[2] can be used to prove the stability of an ordinary differential equation. The Lyapunov function is increadily useful and essential to certain aspects of control and stability theory.

- **Stochastic Networks**

This section will deal with a Boltzmann machine in which the input-output relationship is stochastic.

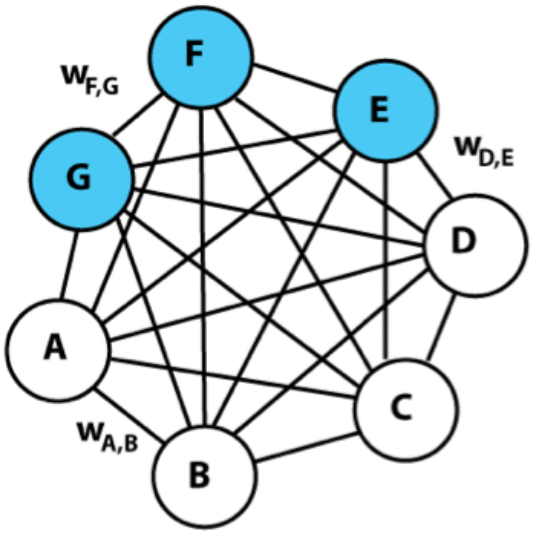

Figure 3.Image of a Boltzmann Machine[4] with some of the edges Labeled. Taken from Wikipedia page for Boltzmann machine[5].

Neurons are treated as binary for this such that, $v_a(t) = 1$ when unit a is active at time t and likewise $v_a(t) = 0$ when unit a is inactive for time t. This binary ability, while it does not account for total synaptic current, it is possible to have incredibly interesting stochastic models that can return patters from somewhat chaotic inputs.

The each discrete time, a unit is randomly chosen to be probabilistically updated to 1 with probability:

$$P[v_a(t + \Delta t) = 1] = F(I_a(t)) = \frac{1}{1 + \exp(-I_a(t))}$$

Likewise,

$$P[v_a(t + \Delta t) = 0] = 1 - F(I_a(t))$$

Here F has the property that as I_a gets larger, it is more likely to take the value 1. Because of this, the state of this network becomes a Markov chain, which means that as the network evolves, there will be random sequences of variables that emerge, v(t + Δt) that only rely

on v(t). In other words, the previous history of the model is somewhat irrelevant when looking at the model at some precise moment in time.

From equation (11), we can use Glauber dynamics to show that v does not converge to fixed point but can be described using a probability distribution function from an energy function, the energy function being: $E(v) = -h \cdot v - \frac{1}{2} v \cdot M \cdot v$. Once this stochastic network has reached its equilibrium state, the probability distribution becomes:

$$P[v] = \frac{\exp(-E(v))}{Z} \text{ with } Z = \sum_{v} \exp(-E(v))$$

This is called the Boltzmann distribution and the precise derivation can be followed along on wikipedia here[3].

- **Boltzmann Distribution / Boltzmann Machine**

Boltzmann machine is a stochastic recurrent neural network created by G. Hinton and T. Sejnowski and is named after Boltzmann distribution. The counterpart to the Boltzmann distribution is a Hopfield network which is another form of a recurrent neural network. All of these different stochastic neural network models are frequently used in computational applications of machine learning.

Toy Model of a Recurrent Network

Creating a Toy Model of a Neural Network in Matlab, we are able to see the possibilities that even a simple Neural network:

External Links

Wikipedia Pages to Recurrent and Feedforward Neural Networks:

http://en.wikipedia.org/wiki/Recurrent_neural_network & http://en.wikipedia.org/wiki/Feedforward_neural_networks

Great in depth Discussion of Neural Netoworks: http://www-cs-faculty.stanford.edu/~eroberts/courses/soco/projects/neural-networks/History/history1.html

1 Chapter 7: http://www.gatsby.ucl.ac.uk/~dayan/book/ch7.pdf by Peter Dayan, Laurence F. Abbott.

2 http://en.wikipedia.org/wiki/Lyapunov_function

3 http://en.wikipedia.org/wiki/Maxwell%E2%80%93Boltzmann_statistics#A_derivation_of_the_Maxwell.E2.80.93Boltzmann_distribution

4 http://en.wikipedia.org/wiki/File:Boltzmannexamplev2.png

5 http://en.wikipedia.org/wiki/Boltzmann_machine

Created with Wolfram *Mathematica* 9.0