Introduction to Neural Networks
U. Minn. Psy 5038
Spring, 1999
Daniel Kersten

Lecture 2b

Models and neural computation

Neural Models

What is a model? A simplification of something complicated to help our understanding.

What is a good model? Reduces complexity but still preserves essential features of the phenomenon. Should go beyond description, and allow us to make predictions.

Computational Neuroscience - levels of abstraction in neural models

Structure-less ("point") models

■ Discrete (binary) signals--discrete time

The action potential is the key characteristic in these models. Signals are discrete (on or off), and time is discrete.

At each time unit, the neuron sums its (excitatory and inhibitory) inputs, and turns on the output if the sum exceeds a threshold.

e.g. McCulloch-Pitts, elements of the Perceptron, Hopfield discrete nets.

A gross simplification.

...but the collective computational power of a large network of these simple model neurons can be great.

And when the model neuron is made more realistic (inputs are graded, last on the order of milliseconds or more, output is delayed), the computational properties of these networks is preserved (Hopfield, 1984).

■ Continuous signals -- discrete time

Action potential responses are interpreted in terms of a single scalar continuous value--the spike frequency--at ith time t[i] .

Both of the above two classes are useful for large scale models (thousands of neurons).

These discrete time models are the standard "**connectionist models**" that provide the basic buidling blocks for the networks modeled in this course. We will show later how the continuous signal model is an approximation of the structure-less continuous-time "leaky integrate and fire" model.

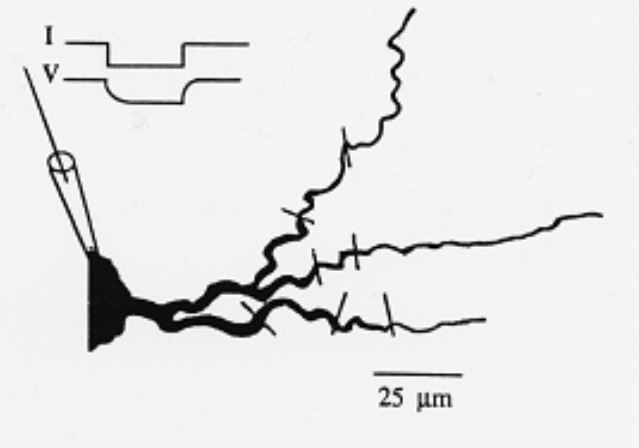
■ Structure-less continuous-time

Analog. More realistic than discrete models. Emphasizes nonlinear dynamics, dynamic threshold, refractory period, membrane voltage oscillations. Behavior represented by differential equations.

- "integrate and fire" model -- takes into account membrane capacitance. threshold is a free parameter
- Hodgkin-Huxley model--Realistic. Parameters defining the model have a physical interpretation (e.g. various ion currents)

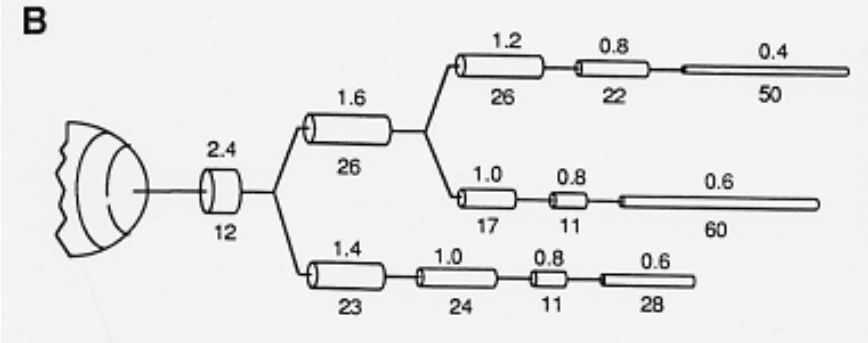
Structured models

■ Passive - cable



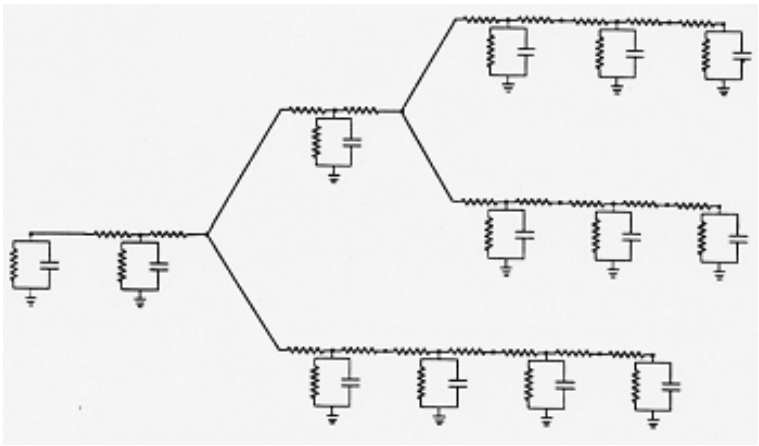
From Segev (1992).

Cable theory - passive trees. Assume membrane is passive. Take into account dendritic morphology or structure. (Rall, 1964). Uses cable equations on segments of dendrites.



From Segev (1992).

■ Passive - compartmental



Segev (1992).

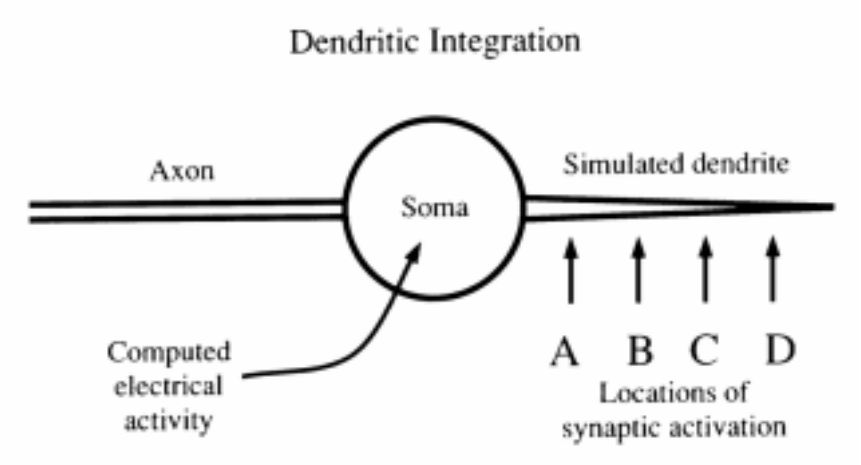
■ Dynamic - compartmental

Complex non-linear trees to model non-linear dynamical properties. Computer simulations necessary. Theory difficult.

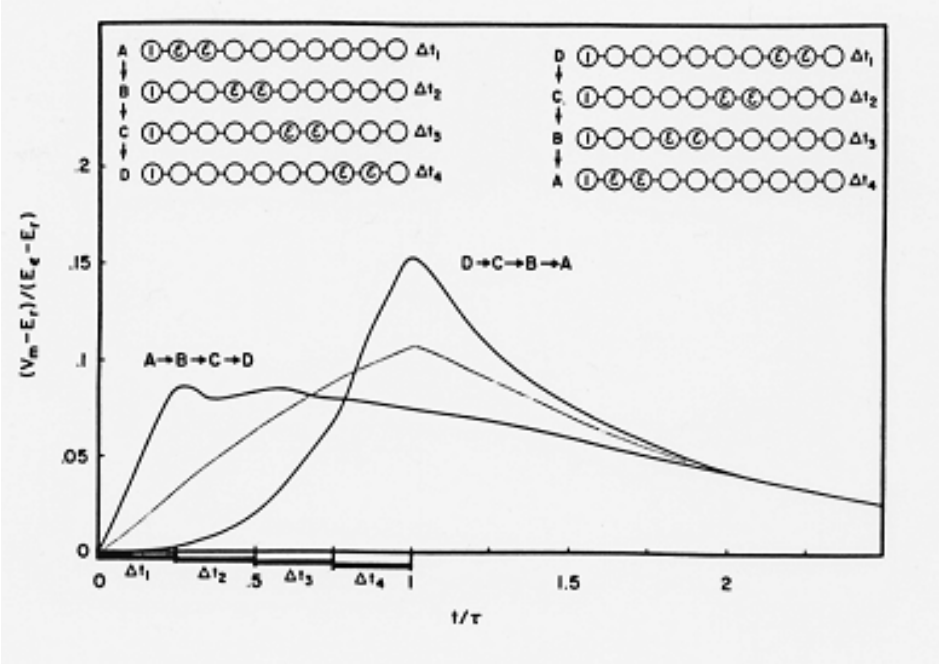
One reason dendritic structure is important because it can show what a single neuron can compute.

■ Dendritic structure shows what a single neuron can compute--Rall's motion selectivity example

Example: **"motion selective" neuron**



Consider the sequential stimulation of the dendrite from left to right (ABCD) vs. right to left (DCBA). (Recall that information flow in the neuron as a whole is from the dendritic arbor towards the axon.) (Anderson, 1995).



Dependence on location on dendrite. Rall's example.

Basis for visual motion selectivity?

Caveat: It is worthwhile pointing out that although this model of motion selectivity has been around for several decades, it has yet to be established that this is right model for motion selectivity of visual neurons. A major problem has been that dendritic transmission is actually too fast to account for the slow velocities that can be detected by animals (Barlow, 1996).

For the purposes of this course, dendritic morphology and its potential for increased computational power will unfortunately largely be ignored. We should remember that simple phenomena such as our sensitivity to motion direction differences may be computed on a single neuron rather than requiring a collection.

Preview: McCulloch-Pitts

The next lecture will provide an overview of the first formal neural model--the McCulloch-Pitts model. This was developed in the 1940's and is famous and important because it showed that with a few simple assumptions, networks of of neurons may be capable of computing the full scale of logical operations.

The model ignores some of the very properties we just looked at that might be important for certain kinds of neural processing (e.g. motion direction selectivity through dendritic cable transmission properties). The model abstracts properties that at the time seemed the most essential. Although some of the basic facts about the physiology were wrong, the notion of neurons as computational devices remains with us.

In preparation for next time, here is a review of two basic logic operations, and how to compute logical functions in Mathematiica.

Example 1: Inclusive OR.

a	b	c
0	0	0
0	1	1
1	0	1
1	1	1

Example 2: AND

a	b	c
0	0	0
0	1	0
1	0	0
1	1	1

Next time we will see how a model neuron can be said to be doing *threshold logic*.. In the meantime, you can practice a bit more *Mathematica* by constructing simple logical functions for OR, AND, exclusive OR, and NAND. *Mathematica* uses similar notation to the C programming language. Alternatively, you can use **And**, **Or**, **Not.**

■ **Logic in *Mathematica***

```
a = True; b = False;

a || b
Or[a,b]
a && b
Xor[a,b]
```

```
True
```

```
True
```

```
False
```

```
True
```

```
MyNAND[a_,b_] := And[Not[a],b]
MyNAND[True,False]
```

```
False
```

More notes on getting started with *Mathematica*

■ **Numerical Calculations.** Last time you saw how you can do arithmetic. Try other operations, 5^3, 4*3 (note that 4 3, where a space separates the digits is also interpreted as multiplication). Note that if you try division, e.g. 2/3, you get the exact answer back.

```
N[2 3]
```

```
0.666667
```

You can go back and select an expression by clicking on the brackets on the far right. These brackets are features of the Macintosh interface and serve to organize text and calculations into a Notebook with outlining features. You can group or ungroup cells for text, graphs, and expressions in various ways to present your calculations. Explore these options under Cell in the menu. You can see the possible cell types under the **Style** menu.

The most recent result of a calculation is given by **%**, the one before by **%%**, and so forth. By ending an expression with **;** you can suppress the output--this is VERY useful later when the output might be a list of a 10,000 neural activity levels!

```
(3/4)/6;
(3 4)/6;
```

```
%%
```

```
1
8
```

■ **Defining functions.** In the next lecture, you will use *Mathematica* to model the generic connectionist neuron. Part of the model will require defining a function. Here is an example:

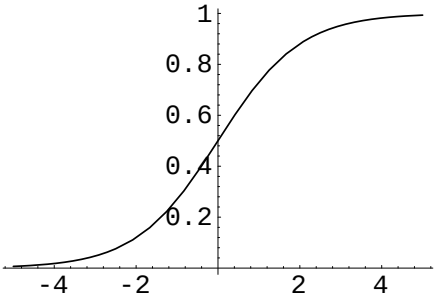
```
squash[x_] := N[1/(1 + Exp[-x])];
```

The underscore, **x_ is important** because it tells Mathematica that x represents a slot, not an expression. Note that we've used a colon followed by equals (:=) instead of just an equals sign (=). When you use an equals sign, the value is calculated immediately. When there is a colon in front of the equals, the value is calculated only when called on later. So here we use : = because we need to define the function for later use.

Also note that our squashing function was defined with **N[]**. *Mathematica* trys to keep everything exact as long as possible and thus will try to do symbol manipulation if we don't explicitly tell it that we want numerical representations and calculations.

■ **Graphics.** Plotting a graph of the squash function.

```
Plot[squash[x], {x,-5, 5}];
```



This squashing function is often used to model the small-signal compression and large signal saturation characteristics of neural output.

Optional Exercise

Modify the squash function to include a parameter, beta, that controls the steepness of the non-linearity. Make a graph that superimposes two of the plots with different lambdas. Note you can superimpose two graphs as follows:

```
Plot[{Exp[x],Sin[x]}, {x,-5, 5}];
```

Or like this:

```
g1 = Plot[Exp[x], {x,-5, 5}, DisplayFunction->Identity,
    PlotStyle->{RGBColor[0,.5,1]}];
g2 = Plot[Sin[x], {x,-5,5}, DisplayFunction->Identity,
    PlotStyle->{RGBColor[1,.5,1]}];
Show[g1,g2,DisplayFunction->$DisplayFunction];
```

The **DisplayFunction** variable effectively turns the graphics display off and on.

References

Barlow, H. (1996). Intraneuronal information processing, directional selectivity and memory for spatio-temporal sequences. Network: Computation in Neural Systems, 7, 251-259.

Hopfield, J. J. (1984). Neurons with graded response have collective computational properties like those of two-state neurons. Proc. Natl. Acad. Sci. USA, 81, 3088-3092.

McCulloch, W. S., & Pitts, W. (1943). A logical calculus of the ideas immanent in nervous activity. Bulletin of Mathematical Biophysics, 5, 115-133.

Rall, W. (1967). Distinguishing theoretical synaptic potentials computed for different soma-dendritic distributions of synaptic input. J Neurophysiol, 30(5), 1138-68.

Segev, I. (1992). Single neurone models: oversimple, complex and reduced. Trends in Neuroscience, 15(11), 414-421.