

# A Primer on Computational Modeling & Connectionist Networks

## Modeling

Modeling is the stuff of science. You observe a phenomenon in reality, you abstract it by creating a model of it, your model makes predictions which you then test against reality and come back to refine your model. So, a model is a depiction of reality that can make predictions.

Take physics, for example. You are sitting under a tree and an apple falls on your head. If you were so inclined, you'd wonder about how the apple's velocity at the moment it hit you, the time it took to fall, and so on. You might come up with a formula, say:  $v = g * t$  ( $v$  is velocity,  $g$  is a constant for acceleration and  $t$  is time) to describe what you observed. That formula is an abstraction, and a model. There is no real time, no real gravity and nothing really falling when you use this formula, but it is useful because if you plug numbers into it, it spits numbers out – it makes predictions. Armed with these numbers you can go outside again, drop a bunch of apples from various different distances and figure out that for things to make sense,  $g=9.8\text{m/s}^2$ .

## Connectionist Models

Connectionist systems and neural networks are computational models. Connectionist networks are made out of many many simple elements (**nodes**) that are connected among them (through **weights**) and thus affect each other's behavior (**activation**). These networks of simple elements are, in reality, a bunch of formulas. The formulas governing how each element of a connectionist system works are very simple (addition, multiplication), however, when you put together hundreds (or even tens!) of these simple elements, things get complicated. To figure out what a connectionist network predicts, you have to build it and "run" it. This is typically done by using a computer program to simulate the network (calculate all the formulas for all the elements), which is why it is called *computational* modeling.

Connectionist networks are used to model learning/adaptive behavior because they change with time, that is, they *learn*. Being neurally inspired, these systems learn by changing the weights that connect the nodes. This will become clear with an example, but before that, a brief summary about the components of neural networks.

Connectionist networks are typically made out of:

**Nodes** – this is the simplest element in the network. Each node has an associated number called "activation" which corresponds, loosely speaking, to the firing rate in a neuron.

**Weights** – Nodes are connected to each other through weights. Weights are, you guessed it, a number. The activation of a node at any time depends, in part, on the activation of the nodes connected to it and the weights with which they are connected.

To describe a network you need mainly two formulas: one describing how activation is computed from the activations of the nodes connected to it, and one describing how weights change. To calculate the activation value of a node, a common activation formula consists of adding all the activation of the nodes to which *that* node is connected, and weighing this sum using the connection weights. That is, each of the nodes gets to contribute some of its activation.

A typical learning formula (or weight update rule) is Hebbian learning, which I expect most of you are familiar with. This formula can be pithily expressed as “Nodes that fire together, wire together”. That is, if two nodes are active at the same time, the weight connecting them increases; if they are not, the weight decreases. If you think about it you will see that what this rule does is capture the correlations (co-occurrences, statistics) in the environment.

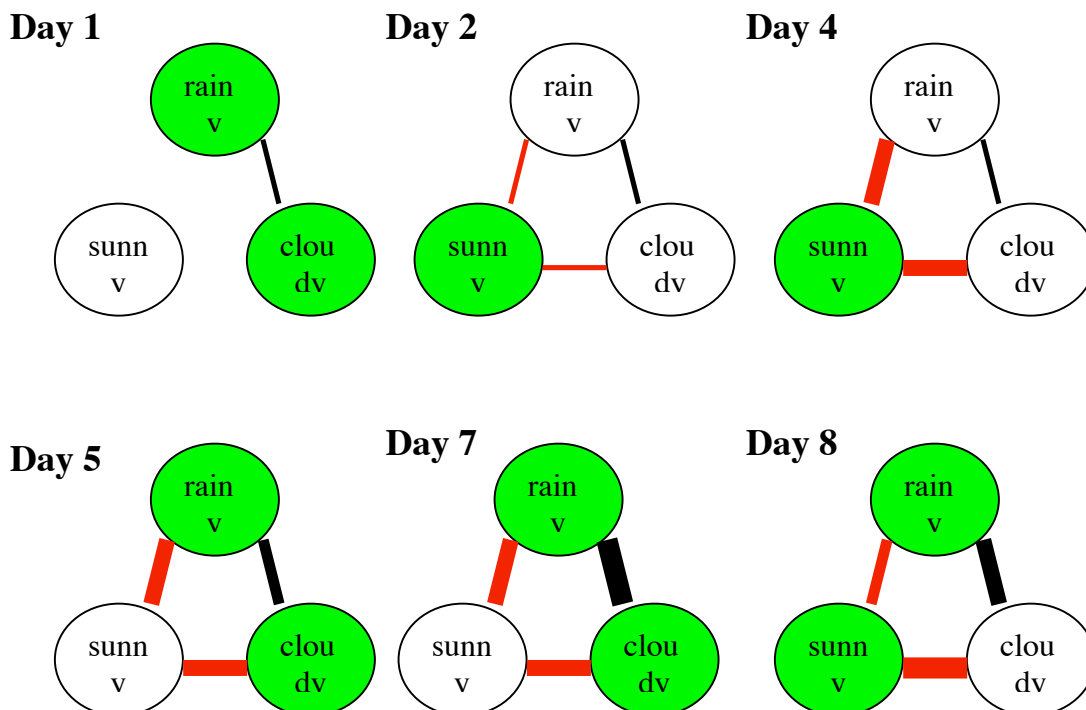
### An example

Say you have a tiny network designed to capture how weather works. In this network we have nodes that represent “rainy”, “sunny” and “cloudy”. If it is cloudy, the “cloudy” node is activated, if it is sunny, the “sunny” node is activated, and so on. Now, this network is going to be trained on what happens in the world. For example:

Day 1: cloudy and rainy  
 Day 2: sunny  
 Day 3: sunny  
 Day 4: sunny

Day 5: cloudy and rainy  
 Day 6: cloudy and rainy  
 Day 7: cloudy and rainy  
 Day 8: sunny and rainy

In the beginning, cloudy and rainy are activated at the same time, so the weight between



them will increase; sunny is not activated, so the weight between it and the other two nodes will decrease. In the figure, you can see how the weights are updated according to what happens each day (activation in the nodes is shown as green, positive weights are black lines and negative weights are red lines). After the network has seen all these examples, it will have learned that cloudy and rainy are highly positively correlated with each other (connected with a thick black line). It will also have learned that sunny is negatively correlated with both cloudy and rainy, though more so with cloudy (thick red line) than with rainy (thin red line).

Now that the network has captured the statistical regularities embedded in those 8 days, it can be “ran”. So, if I know that it is cloudy for sure, I can activate the node for cloudy. Cloudy, in turn, will activate the node for rainy (because positive activation times the positive weight = a positive number) and decrease any activation sunny may have (because positive activation times a negative weight = a negative number). So, if you know it is cloudy, you can predict it is likely to be rainy and unlikely to be sunny.

### **About the model we saw in class**

Here’s what you need to know about the model we talked about in class. Very young children know a lot about how words work, how words go with categories and how categories are organized. If they didn’t they would be unable to learn words because they would spend all eternity contemplating the infinite number of possible meanings for any given word (the indeterminacy problem). For example, young children know that solid things are named by their shape, non-solid things by their material, things with eyes by their shape and texture, and so on. There are two opposing ideas to explain this. On the one hand, the nativists say children come to the task of language learning with knowledge specific to language in the form of biases. These biases constrain word learning and make it possible. On the other hand, connectionists say children *learn* how words work by *learning words*. The first few words are a chore, but after learning a good amount of nouns (about 50 according to some studies), children figure out that names for solid things refer to the thing’s shape, and names for non-solid things refer to the thing’s material.

What the model does is take the first couple hundred words children learn – the ones children typically know by the time they are 30 months old – and train a network on them. The network will extract the regularities embedded in children’s vocabularies, in a way very similar to the little weather example above. That a connectionist network can learn the words is not surprising, what is surprising (at least to some, especially the nativists) is that this network exhibits the same kind of general knowledge children do. That is, the networks not only learn that ball is ball-shaped and that milk is milk-material, but given *any* new solid or non-solid, they will highlight its shape or its material appropriately, like children. The networks also make predictions about how children will name things under different circumstances, and when tested in children, these predictions have been shown to be true, which is further evidence that the model is capturing something fundamental about the phenomenon it is trying to model.