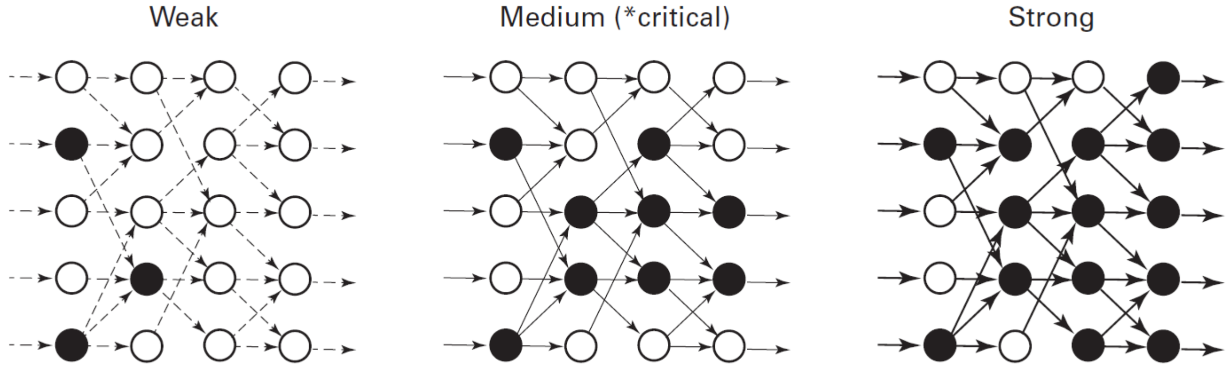
Exercises for Chapter 1, The Main Idea

The main idea conveyed in Chapter 1 is that neural networks process information best when they operate near the critical point. Because this chapter was based on intuitions, these exercises will be brief and not very quantitative. More quantitative exercises will come in later chapters.

Figure 1.4 in the book shows a feed-forward network, where the first layer of neurons on the left can be activated by an input. After the activity is successively propagated, it arrives at the final output layer on the right.



Broadly speaking, the network can be tuned into three different regimes by setting the transmission probabilities between the neurons. When these probabilities are weak, the network will be subcritical. When they are strong, the network will be supercritical. When they are at some intermediate value, the network can be critical, where it will have special information processing capabilities.

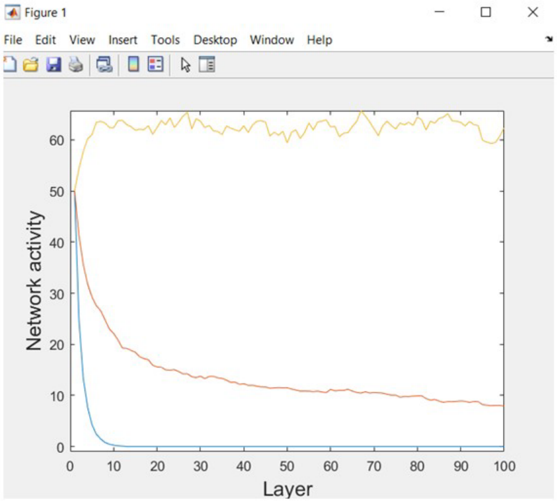
We will explore three aspects of this model. First, we will see how *activity* propagates differently in each (subcritical, ~critical, supercritical). Second, we will see how *information* is transmitted differently in each regime. Third, we will look for a connection between criticality and power laws.

To prepare for these exercises, be sure to download the Matlab programs for this chapter (FeedForwardDecayDemo, FeedForwardActivity) and place them in your Matlab path.

1. **Propagating activity**
   Run the FeedForwardDecayDemo function by giving it an input of 50 (here 50 neurons being stimulated – this can vary from 0 to 100) by typing this in the Matlab command line:

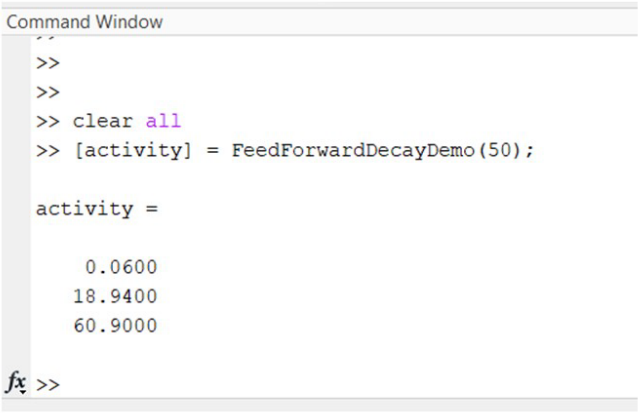
   ```
   [activity] = FeedForwardDecayDemo(50);
   ```

You should see an output graph something like this:



The output of the function FeedForwardDecayDemo when it is given an input of 50 (neurons to be stimulated). This branching model network has a layered, feed-forward structure and has 100 neurons per layer, 10 connections per neuron, and 100 layers. Activity in each layer is plotted. The upper curve is for the supercritical model; middle for the nearly critical; lower for subcritical.

The network has 100 neurons in each layer, so you can stimulate anywhere from 0 to 100 of them. The program should take about 15 – 30 s to run. It should also output some numbers in the command line, like this:



```
>>
>>
>> clear all
>> [activity] = FeedForwardDecayDemo(50);

activity =

    0.0600
   18.9400
   60.9000

fx >>
```

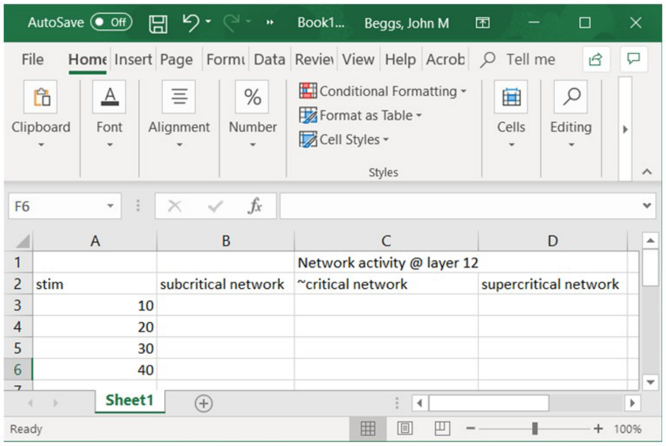
The output of the function FeedForwardDecayDemo when it is given an input of 50. Note that the variable "activity" has three output values, one for each network. The top one is from the subcritical, the middle from the critical, and the last is from the supercritical.

Exercise: Try different input values so you can get a sense of how the three different networks respond. If you stimulate with larger numbers, can you get the lower curve for the subcritical network to decay more slowly? If you stimulate with smaller numbers, can you get the upper curve for the supercritical network to rise less rapidly? What happens to the middle curve as you stimulate with different numbers?

2. Information transmission

Here we will try to understand which of the three networks is best at transmitting information. Note that information is not the same thing as activity.

Exercise: To perform this experiment, run the program FeedForwardDecayDemo four times with these values of inputs [10, 20, 30, 40]. Next, look at the values of the variable "activity" that are printed as outputs. These are the average levels of activity in each network in layer 12. Create a table like the one shown below and enter the outputs appropriately.



	A	B	C	D
1			Network activity @ layer 12	
2	stim	subcritical network	~critical network	supercritical network
3	10			
4	20			
5	30			
6	40			

The variable "activity" that is output from FeedForwardDecayDemo can be placed into a spreadsheet like the one shown above. Input different values of stim [10, 20, 30, 40] to the function and enter into the table the average activity from layer 12 that is output for each network. If you were only able to view the output from one network (subcritical, ~critical, supercritical), which one would be most informative in telling you what the amount of stimulation was? This guessing game provides intuition about mutual information between the input and output of the network.

To mimic how information transmission is measured, we can play a guessing game. Look at the output column from a given network and imagine that you had only that to base your guesses on. Which column would best help you to guess what the input to the network was?

Let's measure the correlation coefficients between the input values and the output values for each network. For example, if the output values for the subcritical network were [0.2, 0.1, 0.0, 0.3], you could calculate the correlation coefficient between them and the inputs [10, 20, 30, 40] like this:

```
C = corrcoef([10 20 30 40], [0.2 0.1 0.0 0.3]);
C(2)
```

For this example, you should get C(2) = 0.200, quantifying the correlation between them. Which network's output has the highest correlation with the input?

This is a very rough way of estimating which network transmits the most information about the input.

Next, take the correlation values you obtained for the three networks (`Corr_sub, Corr_crit, Corr_super`) and now plot them, using something like this:

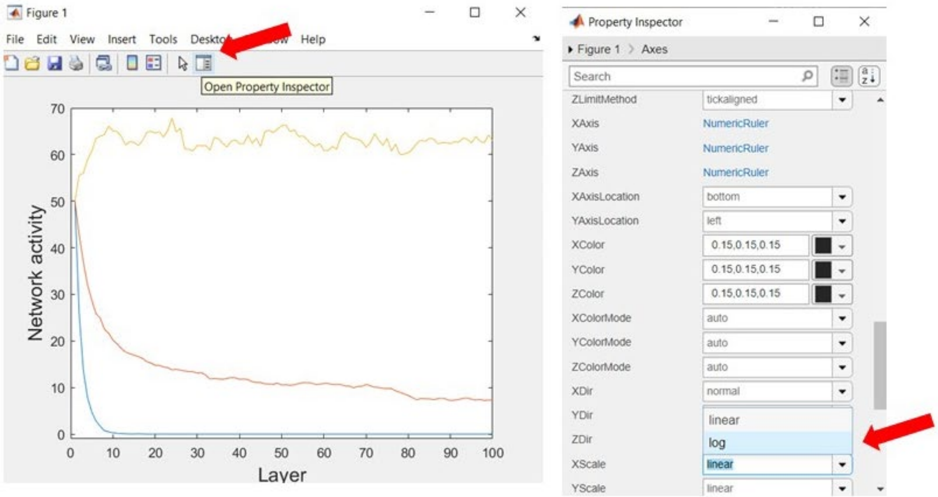
```
figure; plot([1 2 3],[Corr_sub, Corr_crit, Corr_super]);
ylim([-0.5, 1]); xlim([0.9, 3.1]);
```

How does this plot relate to Figure 1.3 in the book?

3. Criticality and power laws

Pages 13-14 of the book explain a connection between very slowly decaying functions, like that produced by the critical network, and power laws.

Exercise: To quickly investigate this, first get the plot produced by FeedForwardDecayDemo when it is given an input of 50. Second, transform the axes of the plot by clicking on the figure’s Property Inspector, under the Rulers tab, as shown below:



ZLimitMethod tickaligned
XAxis NumericRuler
YAxis NumericRuler
ZAxis NumericRuler
XAxisLocation bottom
YAxisLocation left
XColor 0.15,0.15,0.15
YColor 0.15,0.15,0.15
ZColor 0.15,0.15,0.15
XColorMode auto
YColorMode auto
ZColorMode auto
XDir normal
YDir
ZDir
XScale linear
YScale linear

To transform to a plot with log-log coordinates: Click to Open the Property Inspector, as shown by the red arrow. After the window pops up to the right of the figure, select Rulers, and then under XScale and YScale, select log, as shown by the second red arrow.

After transforming to log-log coordinates, do any of the curves now appear to look like straight lines? Are there any that do not? If you stimulate the network with more than 50 (or less), can you cause a curve that previously looked like a power law to be disrupted? In other words, does an apparent power law shape depend heavily on the magnitude of the inputs?

Matlab code used for exercises in this chapter, listed in order of use:

FeedForwardDecayDemo
FeedForwardActivity
PickWeights