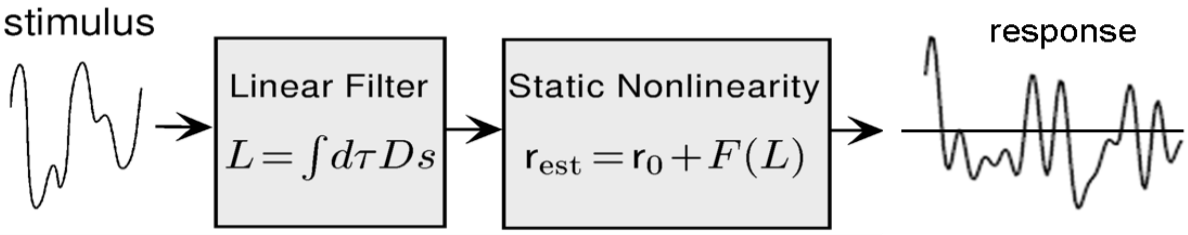
McGill University, Department of Neurology & Neurosurgery
NEUR 503

Introduction to Computational Neuroscience

Lab: Reverse correlation and visual receptive fields

Part 1: Reverse correlation with a one-dimensional input

The function *oned.m* simulates the response of a neuron to a stimulus sequence at a single point in space: It takes a one-dimensional temporal input and returns a one-dimensional temporal output. Specifically, the neuron filters the input, applies a static nonlinearity, and returns the result. You should be able to recover both the temporal filter and the nonlinearity, using the methods described in the lecture and in more detail in Chapter 2 of Dayan & Abbott.



Recovering the filter

- Measure the temporal impulse response (filter), by using an impulse (i.e. a vector like [0 0 0 1 0 0 0]) as input to the function *oned*. Plot the output.
- Measure the temporal impulse response with random noise as the input. You can create a white noise input, using the *rand* function in Matlab. The input should be a vector of length 1000 with all values between -1 and 1.
- Feed this input into the model neuron by calling the function *oned* with the noise input.
- Recover the linear temporal filter by computing the cross-correlation of the input and the output. This is described in the lecture and in equation 1.21 of the textbook. The *xcorr* function in Matlab will be helpful here.
- You will notice that the output of *xcorr* is very long and symmetrical. However, the actual filter is a vector of length 50, so you will need to extract the relevant part from the full cross-correlation. (Here it might be useful to carefully read the Matlab reference on *xcorr*). Plot the resulting filter. How does the filter compare to the one recovered with an impulse as input? Why are the two filters different?
- Note that *xcorr* does not perform the normalization necessary to recover the kernel accurately. That is, if you recompute the filter using random inputs between -10 and 10, the filter increases in amplitude, which doesn't make sense. Something similar happens if you use an input of length 10000 instead of 1000. You can solve this problem by normalizing the filter by the length of the input and by the standard deviation of the input (squared), as in equation 2.6 from Dayan & Abbott.
- Use convolution to calculate the predicted output of your normalized filter for the white noise stimulus used in the previous step. Superimpose the predicted output upon the real output obtained from *oned.m*.
- Visualize the static nonlinearity by plotting the output values predicted by the linear filter with the actual observed output values for the same input, as shown in the lecture and in Figure 2.2 of Dayan & Abbott.

Testing the filter

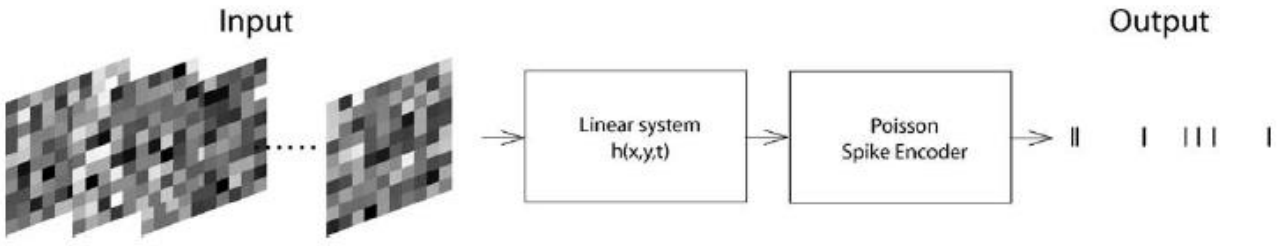
- Compute the mean squared error between the estimated and observed responses (equation 2.3 of Dayan & Abbott) for various lengths of the input vector. Try input vector lengths of 100, 500, 1000,

2000, 5000. (For better results, compute the mean squared error several times for each input vector length and take the average.)

- Find the parameters of the static nonlinearity (equation 2.10 of Dayan & Abbott) that best fit the static nonlinearity in the model. You can do this by eye or alternatively by function optimization. Apply the static nonlinearity to the predicted output and show how it affects the mean squared error for the same range of input vector lengths used above.

Part 2: Spike-triggered averaging with a three-dimensional input

The function *threed.m* contains a simulated V1 simple cell. It has a two-dimensional spatial receptive field, and a temporal response that is the same as the one-dimensional filter obtained in Part 1. You should be able to recover the spatial structure of the receptive field using spike-triggered averaging, as described in Chapter 1 of Dayan & Abbott.



Recovering the filter

- Create a white noise input, using the *rand* function in Matlab. The input should be a three-dimensional matrix, in which the first two indices refer to space and the third to time. The point is to stimulate the neuron with a series of random images. The function *threed* expects each spatial image to be square (same number of rows and columns) for this assignment. Thus your white noise matrix should have size (s,s,len), where len is the length. Try an image size of s = 20 and a length of len = 12000.
- Feed this input into the model neuron by calling function *threed* with the noise input.
- The output will be a one-dimensional vector that contains ones and zeros. The ones correspond to spikes. Calculate the spike-triggered average of the spatial receptive field at the peak of the temporal response, using equation 1.19 from Dayan & Abbott. What is the preferred orientation of the neuron?
- Evaluate the spatial receptive field at the trough of the temporal response. What has changed compared to the peak of the response?