Hannah Machado

March 5th, 2019

CSC 550

Mid-term assignment

**MATLAB code:**

%% Computing the spike-triggered average stimulus of real data

% This first part is accomplished by taking the 150 milliseconds of stimulus immediately preceding each spike and adding them together. This sum is then divided by the total number of spikes fired over the course of the entire experiment to determine the average stimulus preceding a spike.

% This spike-triggered average is, in a sense, a template for what the neuron

% is "looking for" in the stimulus.

kernelSize = 150;

totalCount=sum(rho);

a=zeros(kernelSize, 1);

for i=kernelSize+1:length(rho)

a = a + rho(i)\*stim((i-kernelSize+1):i);

end

a=a/totalCount;

%% We'll first look at the answer; then unpack what we did

figure(5);

clf;

h=plot(a, 'o-');

set(h, 'linewidth', 2);

set(gca, 'fontsize', 14);

xlabel('time'); ylabel('STA');

A screenshot of a cell phone

Description automatically generated

%% Now we simulate the spike trains

% At each frame, the intensity of the uniform screen changes:

% it is drawn randomly, here from a Gaussian distribution.

numsamples = 600000;

stimulus=(1/3\*randn(numsamples,1));

%% Plot the stimulus

figure(1);

plot(stimulus(1:1000))

set(gca, 'fontsize', 14);

xlabel('Time (msec)')

ylabel('Stimulus strength')

title('First 1 second of stimulus');

A screenshot of a cell phone

Description automatically generated

%% Now simulate a model neuron with function getLinear1 that computes the linear

% response of our model neuron.

filterLen = 150;

[linearResp, filter] = getLinear1(stimulus, filterLen);

%% The linear response smooths (averages) the stimulus

% over time. The top panel of the figure shows

% the first second of the stimulus, the small middle

% panel shows the linear filter (which we usually would

% not know), and the third panel shows the first second

% of the linear response.

figure(2);

subplot(2,1,1);

plot(stimulus(1:1000))

set(gca, 'fontsize', 14);

title('Stimulus');

set(gca,'XTIck',[])

subplot(2,1,2);

plot(linearResp(1:1000))

set(gca, 'fontsize', 14);

title('Linear response');

xlabel('Time (ms)');

axes('position',[.13 .47 .2 .1])

plot(filter,'m-')

axis([0 length(filter) 1.1\*min(filter) 1.1\*max(filter)]);

set(gca,'XTick',[],'YTick',[],'Box','on','Units','Normalized')

A screenshot of a cell phone

Description automatically generated

%% We would like to unpack this and see what the linear

% filter is doing

% Choose a random starting point, and examine the stimulus

% starting from the random position and with a length equal

% to the filter length. We'll plot both this stimulus sequence

% and the filter.

thestart = round(rand\*100+1);

thelen = length(filter)

thestim = stimulus(thestart:thelen+thestart-1)

subplot(3,1,1); plot(thestim);

set(gca, 'fontsize', 14); title('stimulus')

subplot(3,1,2); plot(filter);

set(gca, 'fontsize', 14); title('filter')

A screenshot of a cell phone

Description automatically generated

%% Plot the point by point multiplication of the filter

% with the stimulus

subplot(3,1,3); plot(filter.\*thestim);

set(gca, 'fontsize', 14); title('stimulus times filter');

A screenshot of a cell phone

Description automatically generated

%% The linear filter response to the stimulus, is the sum

% of this point by point multiplication. This results in a

% single number. This is also known as inner product or dot product.

sum(filter.\*thestim)

linearResp(thelen+thestart-1)

%% OUT OF CURIOSITY: What if the input was exactly the linear filter

% or variant of (toggle these)

%thestim = filter;

% thestim = -filter;

thestim = flipud(filter);

subplot(3,1,1); plot(thestim);

set(gca, 'fontsize', 14); title('stimulus')

subplot(3,1,2); plot(filter);

set(gca, 'fontsize', 14); title('filter')

subplot(3,1,3); plot(filter.\*thestim);

set(gca, 'fontsize', 14); title('stimulus times filter');

sum(filter.\*thestim)

A screenshot of a cell phone

Description automatically generated

%% Under a simple non-linear model, the firing rate of a neuron is

% a single-valued non-linear function of an underlying linear response.

% Here, getNonlinear1 is used.

% Here we're applying this non-linear transformation on the linear response of

% our simulated neuron.

nonlinearResp = zeros(size(linearResp));

theind = find(linearResp>0);

nonlinearResp(theind) = linearResp(theind).^2;

%% We can plot together the linear and nonlinear response

h = figure(3); clf;

h = subplot(3,1,1);

plot(linearResp(1:150),'o-')

title('Linear response')

set(h,'XTick',[],'XTickLabel',[])

h = subplot(3,1,2);

plot(nonlinearResp(1:150),'o-')

title('Nonlinear function of linear response')

set(h,'XTick',[],'XTickLabel',[])

A picture containing screenshot

Description automatically generated

% Now we compare the spike-triggered average to the filter we used.

% They are similar shape up to a constant multiplication factor

figure(6);

clf;

h=plot([a/abs(sum(a)),filter/abs(sum(filter))], 'o-');

set(h, 'linewidth', 2);

set(gca, 'fontsize', 14);

legend('estimated STA', 'actual filter', 'Location', 'NorthWest')

A screenshot of a cell phone

Description automatically generated

%% So far, we constructed a model neuron and its response to experimental

% stimuli. Here's the first second of each step:

thelen = min(numsamples, 1000);

h = figure(4); clf;

h = subplot(3,1,1);

plot(linearResp(1:thelen),'b-')

title('Linear response')

set(h,'XTick',[],'XTickLabel',[])

h = subplot(3,1,2);

plot(nonlinearResp(1:thelen),'r-')

title('Nonlinear function of linear response')

set(h,'XTick',[],'XTickLabel',[])

h = subplot(3,1,3);

stem(neuralResponse(1:thelen))

set(gca,'Ylim',[0 2]);

title('# of Spikes (1 ms bins)');

set(h,'XTick',[],'XTickLabel',[])

A screenshot of a cell phone

Description automatically generated

%% Here we have a plot of the spike sequences from the real neuron data (rho), and the spike sequences from your synthetically generated spikes (neuralResp)

h = figure(5); clf;

h = subplot(3,1,1);

plot(linearResp(1:150),'b-')

title('Synthetic Linear response')

set(h,'XTick',[],'XTickLabel',[])

h = subplot(3,1,2);

stem(neuralResponse(1:150))

set(gca,'Ylim',[0 2]);

title('Synthetic Spikes');

set(h,'XTick',[],'XTickLabel',[])

h = subplot(3,1,3);

stem(rho(1:150))

set(gca,'Ylim',[0 2]);

title('Real spikes');

set(h,'XTick',[],'XTickLabel',[])

A close up of a logo

Description automatically generated



A screenshot of a cell phone

Description automatically generated



A close up of a device

Description automatically generated

From these two images above we can see that the firing rates of the synthetic spikes differ from the real spikes. The synthetic spikes are much less consistent than the real data in the period of time. There is a cluster of spikes in the middle and inconsistent spike-less patterns throughout.

%% Now perform the autocorrelation using the real and the synthetic neuron.

% Autocorrelation for 10 milliseconds (5 time frames apart) is given by taking the

%correlation between the spike sequence from the 1st time point to the end minus 5th

%time point, with the spike sequence from the 6th time point to the end

matrixC = zeros(50);

for i= 0:50

matrixC(i+1) = corr(rho(i+1:end),rho(1:end-i));

end

figure(12);

clf;

plot (matrixC, 'o-');

set(gca,'Ylim',[0 1]);

title('Correlation real spikes');

set(h,'XTick',[],'XTickLabel',[])

matrixS = zeros(50);

for i= 0:50

matrixS = corr(linearResp(i+1:end),linearResp(1:end-i));

end

figure(13); clf;

plot (matrixS, 'o-');

set(gca,'Ylim',[0 1]);

title('Correlation of synthetic spikes');

set(h,'XTick',[],'XTickLabel',[])

A close up of a map

Description automatically generated

Time steps

The dip at 2 milliseconds on the real neuronal correlation reflects the depolarization of the neuron when the threshold is met upon a stimulus excitation. The correlation suddenly lowers because the spikes that moment are much more negative when compared to the initial spikes from a resting state neuron or a refractive period. After the initial action potential, the stimulus passes along and the correlation begins to increase again

A screenshot of a cell phone

Description automatically generated

Time steps

**MATLAB code for Linear filter used:**

function [linearResp,filter] = getLinear1(stimulus, kernelSize)

tau = 10;

% Compute the linear response using a single exponential

% lowpass filter. You could substitute other linear filters here if you

% wanted to, but this one is pretty simple. The 3rd row of y is the linear

% response (refer to diffEqTutorial). This takes couple of minutes to calculate

linearResp=zeros(length(stimulus),3);

for i=1:length(stimulus)-1

linearResp(i+1,1) = linearResp(i,1)+(1/tau)\*(stimulus(i) - linearResp(i,1));

end

% Getting rid of the first- and second-order filtered signals, we only

% want the third one.

linearResp = linearResp(:,1);

impulse = zeros(1000,1); impulse(1) = 1;

impulseResp=zeros(length(impulse),3);

for i=1:length(impulse)-1

impulseResp(i+1,1) = impulseResp(i,1)+(1/tau)\*(impulse(i) - impulseResp(i,1));

end

impulseResp = impulseResp(:,1);

impulseResp = impulseResp(1:1:1\*kernelSize);

impulseResp = flipud(impulseResp);

filter = impulseResp;

**MATLAB code for Non-linear filter used:**

function nonlinearResp = getNonlinear1(linearResp);

nonlinearResp = zeros(size(linearResp));

theind = find(linearResp>0);

nonlinearResp(theind) = linearResp(theind).^2;