Neural Decoding

Naureen Ghani

February 17, 2018

Spike-Triggered Average

We live in a complex sensory environment and receive many signals. Each of these signals activate distinct pathways. The basic premise of sensory neuroscience is that, as we move from the periphery to the internal workings of the brain, individual neurons become increasingly selective for particular features in the sensory world. In other words, *neurons are feature-selective and this helps us form our perceptions of the outside world.* Hubel and Wiesel first identified feature-selectivity in the visual pathway. Since their findings, the race has been on to identify complex feature selectivity in sensory pathways.

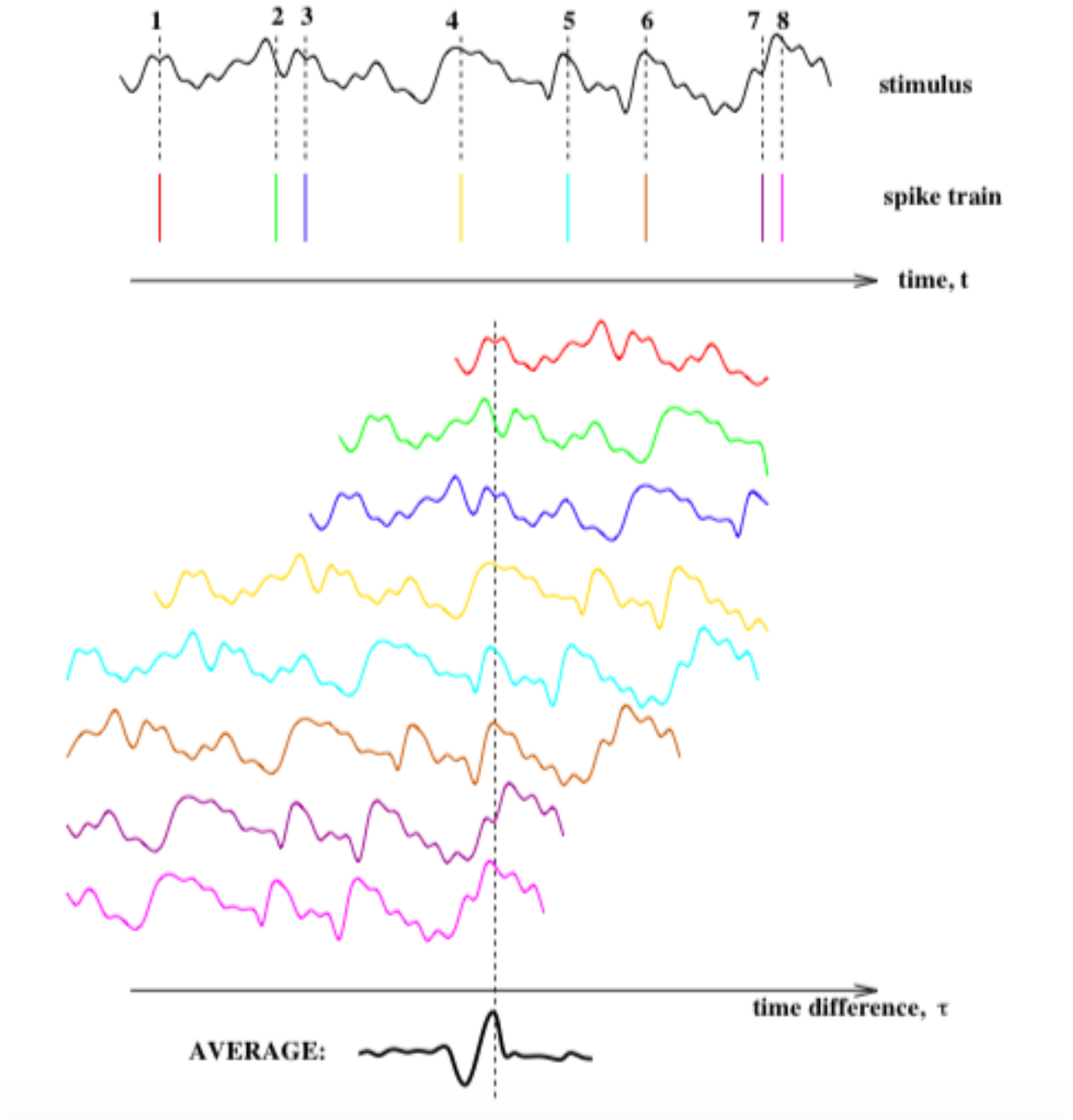
Event-triggered averages are used to try and determine what caused an event to happen. They look at what preceded a number of events and take the average. In neuroscience, events are often *spikes* recorded from a neuron in response to a sensory stimulus. For example, let's say the stimulus is white noise. We would then extract a small piece of data before the spike (***time window)*** and average it with the actual spike. This *average waveform* represents the average stimulus that caused the neuron to fire.

This technique is also used in finance, where the event is price change. Event-triggered averages can identify if a consistent pattern occurs before similar price changes in order to react to them in the future.

Method

To compute the spike-triggered average:

1. Record the stimulus over complete spike train.
2. Record all spike times.
3. For every spike, add the stimulus values surrounding the spike into the spike-triggered average array. For example, a spike at 10.12s, the stimulus at 10.02s goes into the -0.10s bin, the stimulus at 10.03s goes into the -0.09s bin. Do this for a given time before and after the spike. Repeat for all spikes.
4. Once all the values are added into the array, divide by number of spikes to get ***spike-triggered average.***



Data

Let's generate synthetic neural data as an example. In our experiment, we record the activity of a single neuron in response to a white noise stimulus. These signals are often noisy, and require pre-processing. This involves *filtering* and *event (spike) detection.* Event detection is done by using a *threshold* of some value, often the *root mean square of the signal.* Data from a population of cells requires *spike sorting.* These techniques will be covered in a separate lecture.

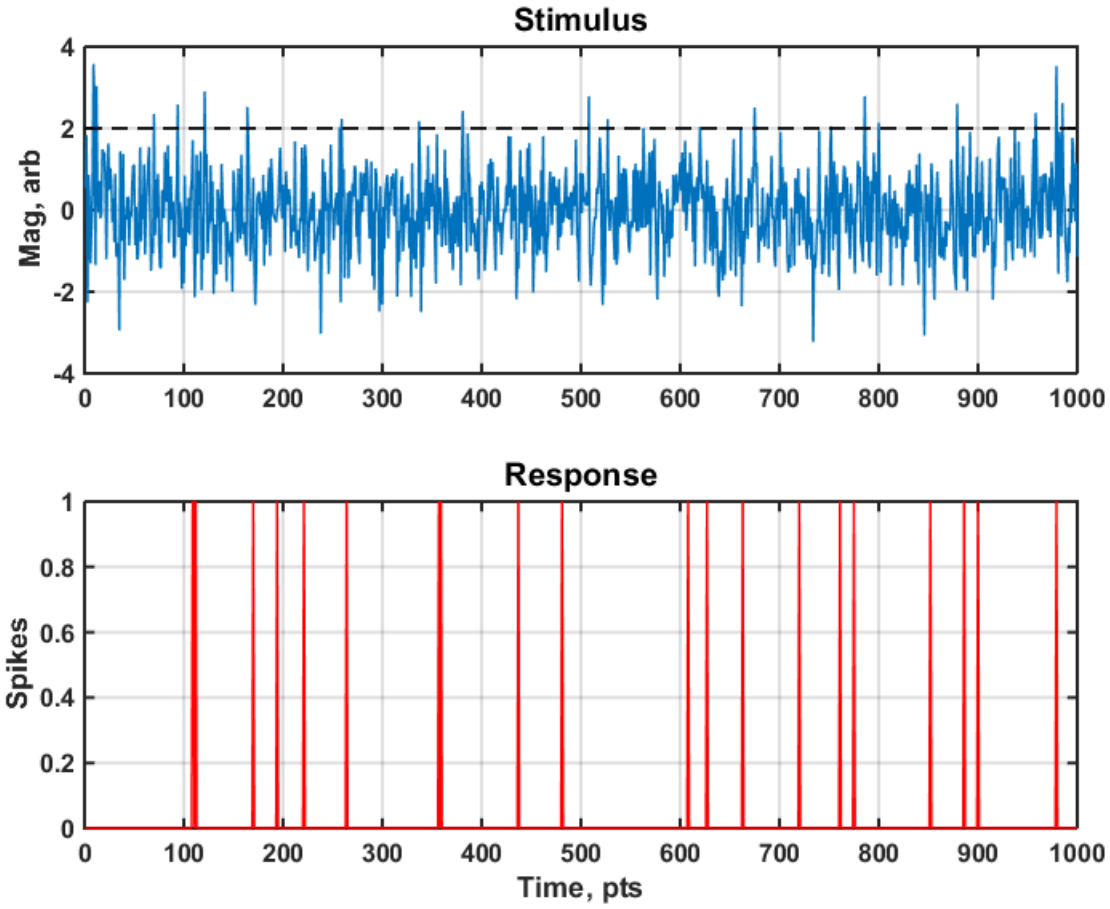
For spike-triggered averaging, we would like to obtain a binary vector representation of spike firing. We want to know what property of the stimulus caused each event.

```
nPoints = 10000; % number of points in stimulus (stim) and response (resp)
stim = randn(1, nPoints); % stim is random noise drawn from a normal distribution
resp = stim > 2; % vector of spikes (when stim is > 2)
```

```
resp = [zeros(1,100), resp(1:end-100)]; % added 100 zeros in front of resp, to
    mimic latency that would exist in a real system (spikes are not instant)
```

Let us plot our first 1000 points of artificial data:

```
subplot(2,1,1), plot(stim(1:1000))
line([1, 1000], [2, 2], 'LineStyle', '--', ...
    'color', 'k')
ylabel('Mag, arb'), title('Stimulus')
subplot(2,1,2), plot(resp(1:1000), 'r')
ylabel('Spikes'), title('Response')
xlabel('Time, pts')
```



With a stimulus and response, we can take the spike-triggered average. For each spike, we want to look at the **stim vector** before the spike occurred for a specific time window. We then want to take an average of this window. In MATLAB, we will plan to:

- Discard spikes in the first time window– This is because we cannot access the data before the spike (not physically present).
- Count the total number of events
- Find the index of each event so we can identify the indexes of the time window preceding each spike
- For each spike,
  - Work out the index range of its preceding time window
  - Collect the data in **stim** from this time window

- Average the data collected from all the time windows

Let us use a time window of 300 points:

```
windowSize = 300;
resp(1:windowSize)=0; % discard spikes in first window
nEvs = sum(resp); % binary vector, we can simply add to find number of events
evIdx = find(resp); % gives indexes of spikes (where a 1 was found)
```

At this point, there are two ways to take the average. We can collect all the time windows and take the mean after the ***for loop.*** The alternative is to keep a sum of the data in the windows that is added to in each iteration of the ***for loop,*** and then divide by the number of interations after the ***for loop.*** The second way is superior because our average needs to be only 1 x windowSize rather than nEvs x windowSize. This is not too important for our needs but space is at a premium for real neural data sets.

```
avg = zeros(1,windowSize); % pre-allocation of space

% For each event
for w = 1: nEvs
        % Find the indexes of the time window preceding the event
        wIdx = evIdx(w)-windowSize : evIdx(w)-1;
        % Add the stim from this window to the average
        avg = avg + stim(wIdx);
end
```

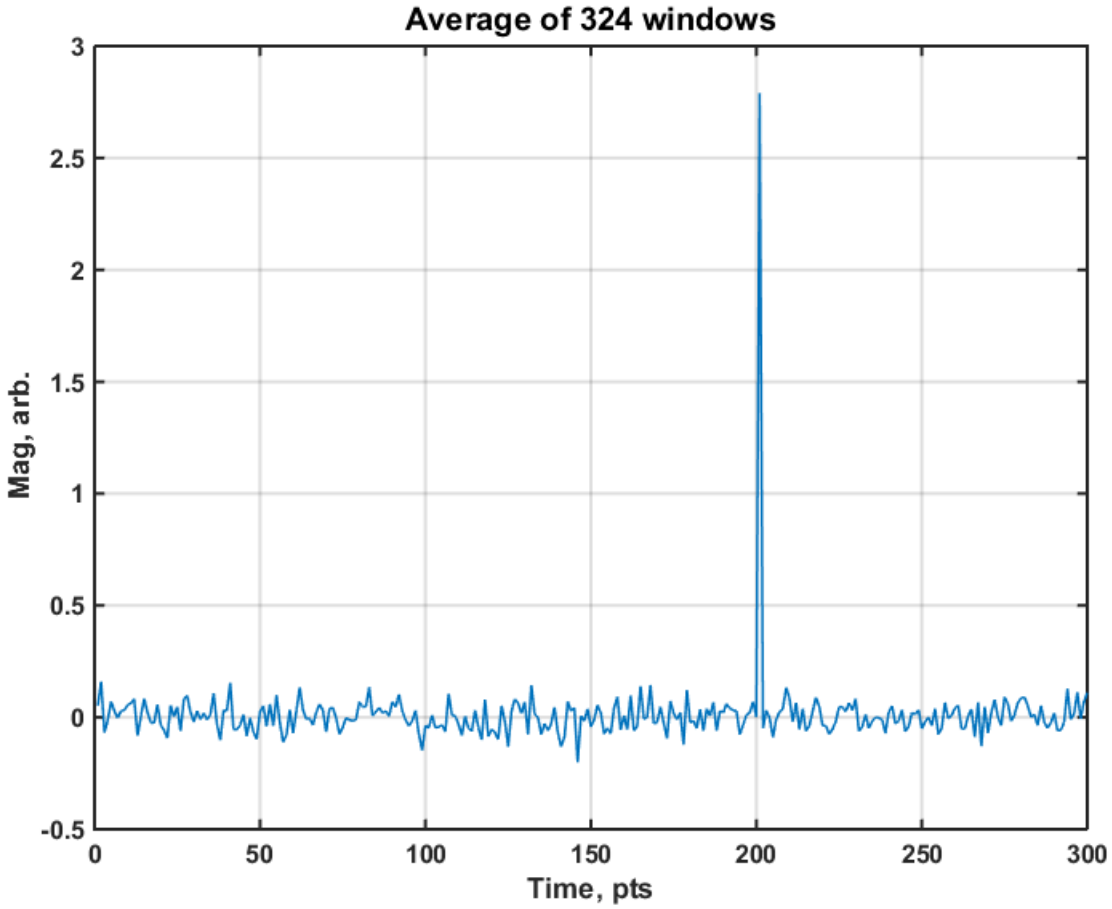
Now we know the number of events and their indexes, so we can collect data from the time window from **stim.** This loop runs from 1:nEvs and after the first iteration (w = 1), it creates a vector of indexes for the time window based on the first event index in **evIdx.** The window is 300 points, so if we have a an event **evIdx(w)** = 400, we extract data from 100:399. We do this iteratively for all events.

The last step is to take the summed signals and divide by the number of observations that contributed to it:

```
avg = avg./sum(resp);
```

The **. / operation** denotes element-wise division where each value in a vector (or m atrix) is divided by a corresponding element. Now let's plot the result:

```
figure
plot(avg);
title(['Average of', num2str(nEvs), 'windows']);
xlabel('Time, points');
ylabel('Mag, arb');
```



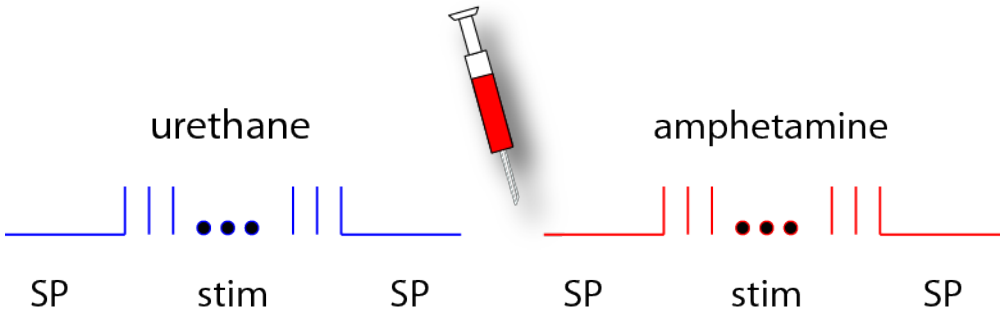
Neurophysiology Dataset

Let us now analyze with a real neurophysiology dataset. This is kindly provided by the University of Lethbridge (Canada) for educational purposes. In this dataset, silicon probes with recording sites (called ***tetrodes***) were inserted into two places in the rat brain (***S1 and mPFC***). **S1** is the **rat "barrel" cortex** and **mPFC** is the **medial pre-frontal cortex.**

The experiment consisted in recording neuronal data in anesthetized rats receiving tactile stimulation under two conditions:

- only urethane
- urethane + amphetamine

Neural data was recorded during *periods* of spontaneous (no stimulation) and evoked activity (where tactile stimulation was given) for both conditions.



Description

This dataset has been saved as a struct and contains:

- SpkCnt : neurons x time
- stim: times of tactile stimulation
- SpkEle: neurons x tetrode

Stimulus-Triggered Activity

In order to have an idea about how neurons in the different areas (S1 and mPFC) respond to stimulation, we can display a ***stimulus-triggered response (LFP and PSTH):***

```
pfc_nns = find(SpkEle<=8); % pfc neurons
s1_nns = find(SpkEle > 8); % s1 neurons
nr_pfc_nns = length( pfc_nns ); % number of pfc neurons
```

Extracting useful information in order to target our analysis, we divide and the label the data to have a better control. For example, we need to know:

- the times of both conditions
- when stimulation was given
- how many times stimulation was given and take data around it

```
% Firing rate for S1 neurons during each stage
fr1 = mean(SpkCnt1(:,s1_nns));
fr2 = mean(SpkCnt2(:,s1_nns));
fr3 = mean(SpkCnt3(:,s1_nns));

% Firing rate for PFC neurons during each stage
fr4 = mean(SpkCnt1(:,pfc_nns));
fr5 = mean(SpkCnt2(:,pfc_nns));
fr6 = mean(SpkCnt3(:,pfc_nns));
```

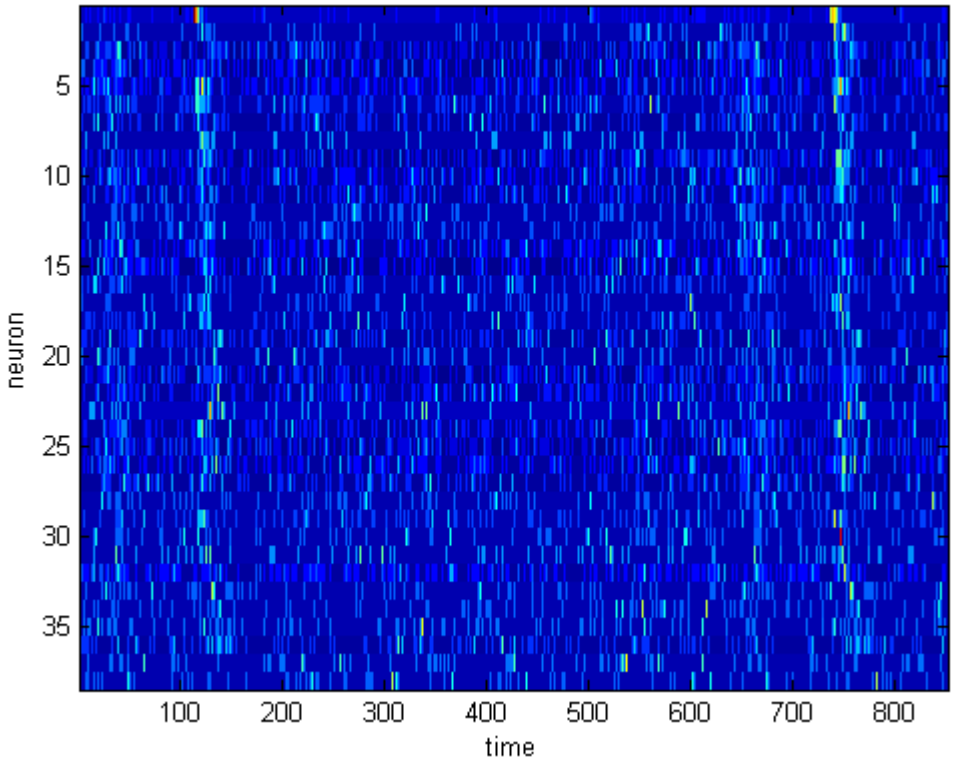
To select active S1 neurons, identify neurons with a firing rate above threshold s1 and PFC:

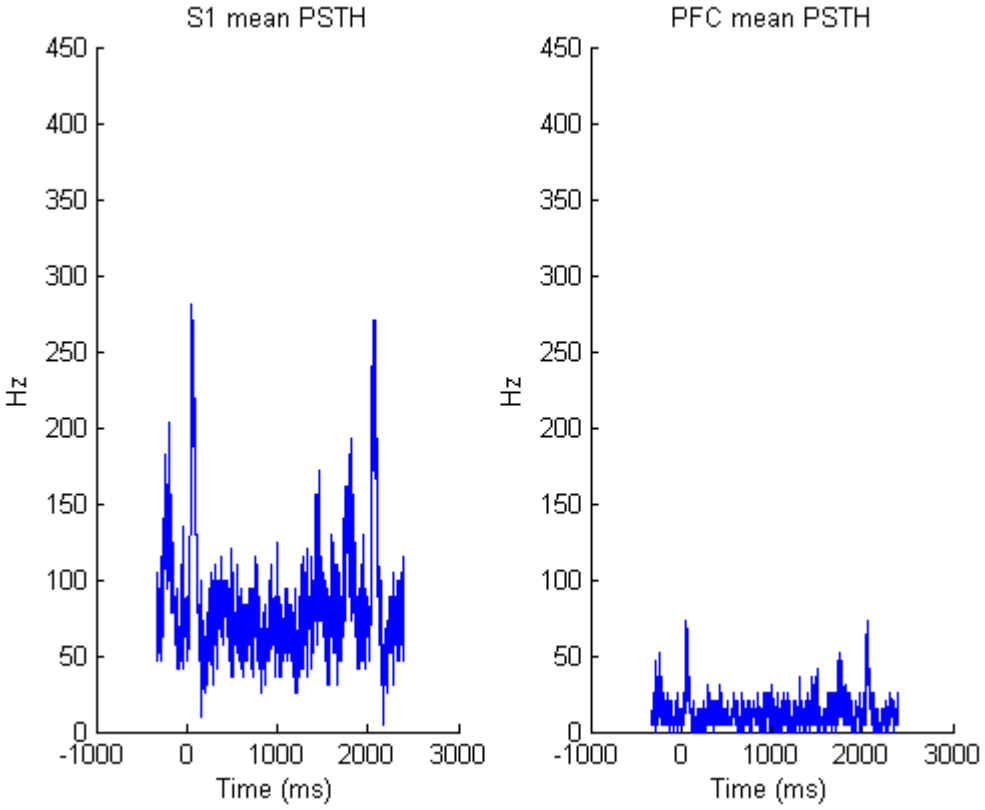
```
fr_th = 0.002;
act_s1nns = find( fr1 > fr_th & fr2 > fr_th & fr3 > fr_th);
act_pfcnns = find( fr4 > fr_th & fr5 > fr_th & fr6 > fr_th);
```

Now let us slice the activity into trial locked time windows and plot the response (raster plot):

```
% tactile stim trig activity before drug
f_tac1 = find(stim_sm==3);
f_tac1( find(diff( f_tac1 ) < 10 | diff( f_tac1 ) > 700)+1) = [];
w1=100;
w2=750;
spk1=zeros(length(f_tac1),w1+w2+1,length(act_s1nns)); spk2=zeros(length(f_tac1),
   w1+w2+1,length(act_pfcnns));

for tr=2:length(f_tac1)-1
      spk1(tr,:,:)=SpkCnt2(f_tac1(tr)-w1:f_tac1(tr)+w2,s1_nns(act_s1nns));
          spk2(tr,:,:)=SpkCnt2(f_tac1(tr)-w1:f_tac1(tr)+w2,pfc_nns(act_pfcnns));
end

% range of the reponse plots in ms
range = (-100:750)*3.2;
```

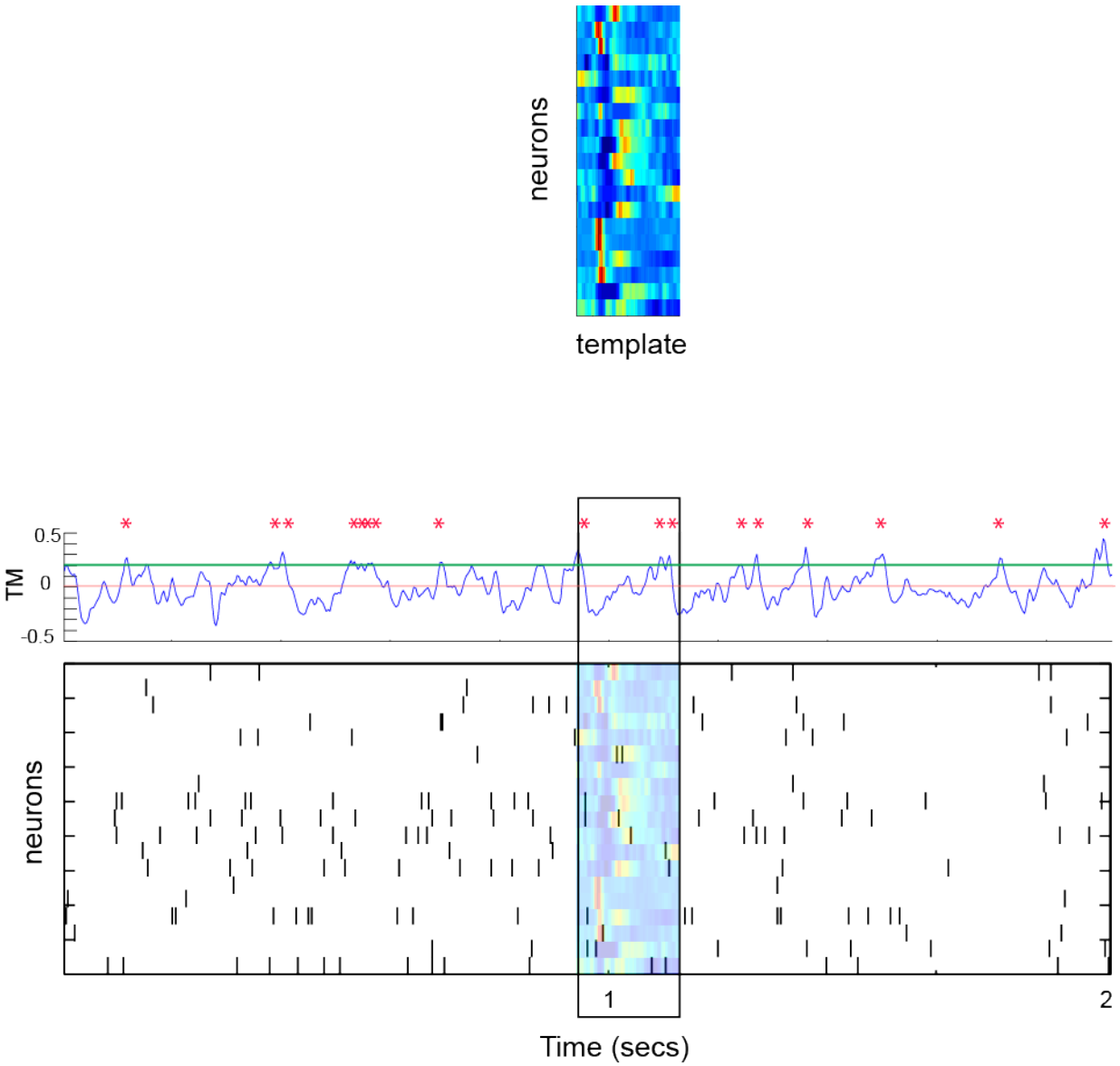
```
% Plotting the response
figure();
title(['Neural response in urethane condition']);
imagesc(zscore(squeeze(mean(spk1)))');
xlabel('time');
ylabel('neuron');
figure();
subplot(1,2,1); hold on;

% PSTH Diagrams
title(['S1 mean PSTH']);
plot(range, 312.5*squeeze(mean(sum(spk1(:,:,:),3))));
ylim([0 450]);
xlabel('Time (ms)');
ylabel(' Hz ');
subplot(1,2,2); hold on;
title(['PFC mean PSTH']);
plot(range, 312.5*squeeze(mean(sum(spk2(:,:,:),3)))); ylim([0 450]);
xlabel('Time (ms)');
ylabel(' Hz ');
```




S1 mean PSTH
PFC mean PSTH
Hz
Time (ms)
450
400
350
300
250
200
150
100
50
0
-1000
0
1000
2000
3000

Template Matching

Template matching quantifies replay of temporal sequences of neuronal activity.



Let us first build the template. To do so, we define it as the (summed evoked act - mean) / std average stim triggered activity of the S1 neurons. Here is the code:

```
t_len = 60;
t_window = 101:100+t_len;
X = squeeze(sum(spk1(:,t_window,:),1))';
zs = zscore(X')';
figure('name','Template');
subplot(1,2,1);
imagesc(zs);

% In order to have a less specific template, we smooth the activity pattern
% for every neuron
g_sigma = 20;
gs = normpdf(-g_sigma*2:g_sigma*2,0, g_sigma/2 )';
for n=1:size(zs,1)
        temp = zs( n, :);
        tmplt(n ,:) = conv2(  temp', gs, 'same');
```

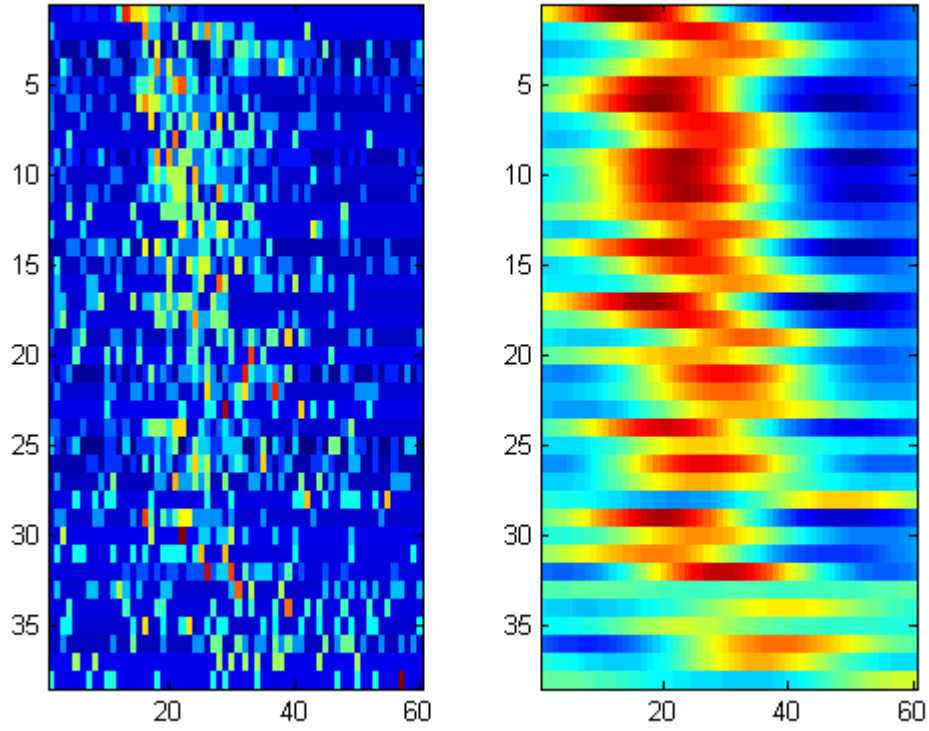
```
end

template = tmplt(:,:);
subplot(1,2,2);
imagesc(template);
```

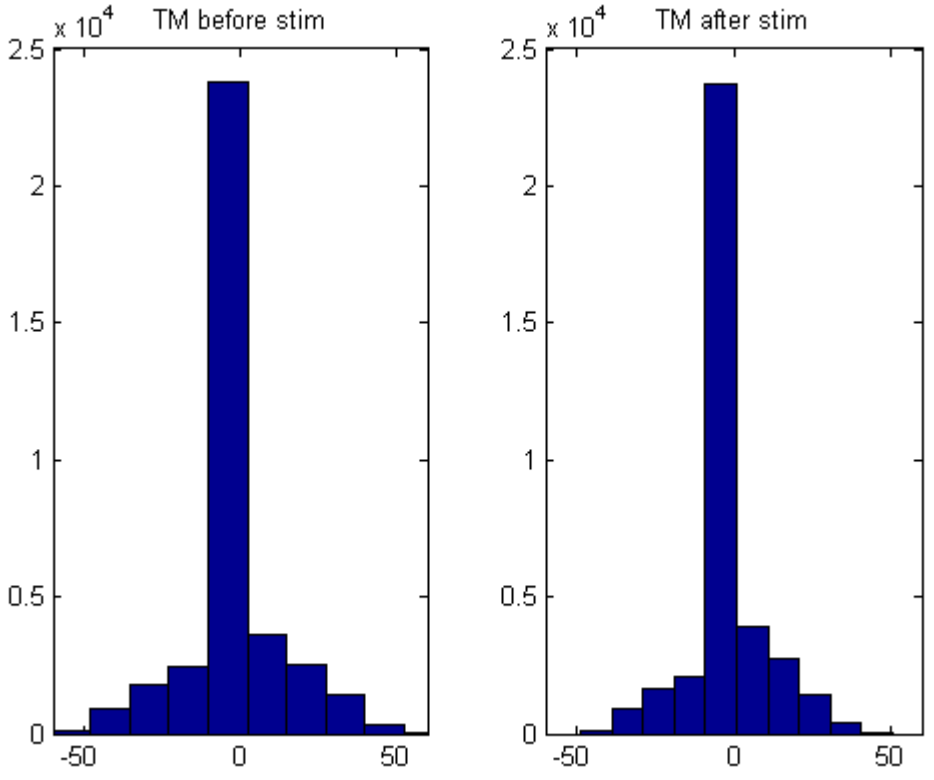
We then write code for the matching:

```
shift = 1;
num_windows = 37000;
norm_bef = zeros(1,num_windows);
norm_aft = zeros(1,num_windows);
% take windows of activity to compare against our template
for j=1:num_windows
        winanalysis = 1+((j-1)*shift) : 1+((j-1)*shift)+t_len-1;
        % activity before stimulation
        tgt = SpkCnt1(winanalysis, s1_nns(act_s1nns))';
        % we normalize in the same way such windows
        tgt_z = zscore(tgt')';
        norm_aft(j) = dot(template(:), tgt_z(:)); % similarity comparison
        % activity after stimulation
        tgt = SpkCnt3(winanalysis, s1_nns(act_s1nns))';
        tgt_z = zscore(tgt')';
        norm_bef(j) = dot(template(:), tgt_z(:)); % similarity comparison
end
```

We show the distribution of the template-matched (TM) signal before and after stimulation:

```
figure();
subplot(1,2,1);
hist(norm_bef);
title('TM before stim');
xlim([-60 60]);
subplot(1,2,2);
hist(norm_aft);
title('TM after stim');
xlim([-60 60]);
```


5
10
15
20
25
30
35
20
40
60
5
10
15
20
25
30
35
20
40
60


x 10^4
TM before stim
2.5
2
1.5
1
0.5
0
-50
0
50
x 10^4
TM after stim
2.5
2
1.5
1
0.5
0
-50
0
50