Practical 5: Mutual information in a Poisson spiker

Neural Computation 2004-2005. Mark van Rossum

3rd November 2010

1 Poisson spiker

In this practical we look at the information in a spike train generated by a neuron which fires roughly like a Poisson spiker. We apply a brief stimulus to the neuron and measure the response. First, consider just one brief output bin, which can contain a spike or not. The probability that a spike is generated in the bin is \( p \). In principle, the Poisson process can have an arbitrary number of spikes per bin (although with a low probability for high numbers). Here we assume that we are dealing with a more neuron-like behaviour: because of refractoriness there is maximally one spike per bin (an ‘amputated’ Poisson process).

Next, we introduce a longer observation window and split the response of the neuron in multiple time bins. The probability per bin remains \( p \). Possible responses are written in an array called a word. If we have four bins a possible response ‘word’ is: (0,1,1,0).

We will consider a stimulus \( s \) which for simplicity corresponds to the spike probability and runs between 0 and 1\(^{1}\). In other words \( s = p = 0 \ldots 1 \).

There are two ways to go about this practical: pen and paper, or matlab... You choose.

2 Matlab approach

2.1 Writing the code

1. Generate the modified Poisson train. You can do this using:
   \[
   \text{floor(rand(nbins,ntrials)+s)}.
   \]

2. Next, we need to uniquely label the possible response words with a number. For the example of the nbins=4 case, an easy way is to give (0,0,0,0) number 1, (1,0,0,0) number 2, and so on so that (1,1,1,1) number 16. In other words we convert a binary string into a decimal number. In matlab:
   \[
   \begin{align*}
   \text{base} = [ & ] \ % \text{ need to create the base only once} \\
   \text{for} \ i=1: \ \text{nbins} \\
   \text{base}=[\text{base} 2^{\text{i-1}}] \\
   \end{align*}
   \]

3. We need the probability \( P(r,s) \), which is the joint probability of \( r \) and \( s \). You can store \( P(r,s) \) in a \( nstim \times 2^{\text{nbins}} \) matrix (see below for \( nstim \)).

\[^{1}\text{The right choice of stimulus is an important factor in these calculations. Try to figure out why.}\]
4. We can now complete the Matlab script. The first part is to generate the data/simulate the neuron. Write a double loop:

(a) Loop over different stimuli intensities. Divide the total range of stimulus intensities into \text{nstim} parts (50 is a good value).

(b) For each stimulus intensity, loop over trials (\text{ntrials}=100 is a good starting value), calculate a response and store the responses in the matrix:

\[
P(r, s) = P(r, s) + 1/(\#\text{total entries})
\]

% given a word with \text{nbins}, eg (0,1,1,0)
index = base*word + 1; % runs from 1 to 2^\text{nbins}
prs(istim,label) = prs(istim,index) + 1/ntot;

5. To calculate the mutual information, we write another part which analyses the generated data. We need the following quantities.

- \text{P}(s): probability for stimulus \text{s}; \text{P}(r): probability for response \text{r}.
- From \text{P}(r, s) calculate \text{P}(r) and \text{P}(s) using the sum function. Check that you properly normalise the distributions; you should have \[
\sum_r P(r) = \sum_s P(s) = \sum_{r,s} P(r, s) = 1.
\]

Use \[
I_m = \sum_s P(s) \sum_r P(r|s) \log_2 \frac{P(r|s)}{P(r)}
\]
and that \[
P(r|s) = P(r, s)/P(s).
\]

(a) Loop again over all stimulus intensities

(b) Loop over the possible patterns and sum the mutual information for given stimulus.

2.2 Matlab: Properties of the mutual information

Now that we have the program start with one bin.

- What is the output entropy of the spiketrain? How many bits is the mutual information? How many bits is that per spike?
- Increase the number of bins. Do you expect the mutual information to increase? Do you expect that it goes up linearly? [Hint: What is the information in the input? The mutual information can not be larger than the stimulus information.]
- Research the parameter \text{ntrials}. Explain your results.
- Set \text{nbins}=3. Write some extra code to plot the output information, \[
H = -\sum_{r,s} P(s) P(r|s) \log_2 P(r),
\]
as a function of \text{r} (the pattern label).
- Do the same for the noise entropy \[
H_{\text{noise}} = -\sum_{r,s} P(s) P(r|s) \log_2 P(r|s).
\]
- Explain your results.
- Try different input distributions. Why does it change the information content? As simple manipulation, change \text{nstim} to much smaller values.

3 Analytical approach

First consider one bin. What is the probability for a 'one' and a 'zero' for a given \text{s}? In other words write down \text{P}(r|s).

Calculate the information by first calculating \[
\sum_r P(r|s) \log_2 \frac{P(r|s)}{P(r)}
\]
for a given \text{s}, and then integrating it over \[
\int_0^1 ds P(s).
\]

Extent to two bins. Now there are four possible responses. Calculate the information.
Also consider the case that \text{s} can only take two values.