Advanced brain imaging methods: Practical 3

Remi Gau

14th March 2016

1 Convolution and frequency spectrum

In this first exercise, we are going to see how the hemodynamic response function (HRF) acts as a low-pass filter on the neuronal activation.

1.1 Model neuronal response, HRF and convolution

Create a vector that models the neuronal activation of a block designed experiment where you have 8 repeats of 6 seconds of rest followed by 6 seconds of activation. A rest are modelled by and an activation is modelled by 1. To keep things simple, let's say that the temporal resolution is 1 Hz (1 data point per second).

For this part, the only matlab functions you need are 'ones', 'zeros' and 'repmat'.

Once you have done this, use the appropriate SPM function to generate a canonical HRF with the same temporal resolution. See the practical 2 on how to do that.

Finally convolve (with the function 'conv') the model of the neuronal activation with the HRF to get model BOLD response corresponding to to your block design: typically this model BOLD response would be one of the regressor of your design matrix.

Plot those results (function 'plot').

1.2 Frequency spectrum

Now let's have a look at what frequencies are present in those two models and the HRF. To do this we use the Fourrier transform, which is implemented in matab by the function 'fft' (fast Fourrier transform). Besides we are only interested by the real part of the result of the Fourrier transform, we can take only take the absolute value of the result (function 'abs'). Finally because we are interested in the power spectrum, we need to take the square of each value.

```
[Freq] = abs(fft(Signal)).^2;
```

Because in the end we will want to compare different power spectrums, you might want to normalize them so that the area under each spectrum is equal to 1.

You can then plot each spectrum. Though you will realize that each spectrum is symmetrical so you only need to plot the the first half of it.

1.3 Slow event-related experiment

Now do the same with a vector that models the neuronal activation of a slow event-related designed experiment where you have 8 repeats of 1 second of activation followed by 16 seconds of rest.

2 Design efficiency

Now we are going to connect what we have just seen with the efficiency of a design. The efficiency is linked to the formula of a t-test:

$$t = \frac{c\beta}{\sqrt{\sigma^2 c(X'X)^{-1}c}}$$

Where c is the contrast of interest. β are the estimates of the weight associated to the design matrix X. σ^2 is the error variance and therefore the only term that is data dependent. Therefore the efficiency of $c\beta$ is proportional to:

$$\frac{1}{c^T(X^TX)^{-1}c}$$

Knowing this, we are going to compare several designs to see how efficient each of them is at measuring an activation.

2.1 Comparing designs

Creates vectors that model the neuronal activation of:

- a block design,
- a slow event-related design,
- a fast event-related design,
- a fast event-related design with jittering.

Use a temporal resolution of 1 Hz. All models must have 16 seconds of activation. For the slow and fast event-related designs, find appropriate stimulus onset asynchrony so that the former but not the later leaves enough time to the modelled BOLD response to come back to baseline.

A quick and dirty way to add jittering to the fast event-related design is to take the vector that models the fast event-related design and randomly permutate all its values (the function 'randperm' might help you to do this).

Once you have done that compute the efficiency of each design using the formula above. For this you will have to figure out what the appropriate values of c and X.

Finally you can plot the power spectrum of design and compare their relative efficiency.

3 Non-parametric test

Non-parametric test are usually used when one or more of the assumptions that underlie parametric tests are violated. In this example we are going to see of compute the p-value of associated the mean difference between 2 non-gaussian ditributions.

3.1 Creating the samples

Matlab can generate i values sampled from a gaussian distribution of mean μ and standard deviation σ like this:

```
X = randn(i,1)*sigma + mu;
```

Create 2 sets of data where each set is mixture of 2 gaussian distribution. You can choose the values you want for i, μ, σ for each of the 4 gaussians.

Once you have done that compute, the t-value of the difference between those two sets of data:

$$t_{ini} = \frac{\mu_1 - \mu_2}{\sqrt{\frac{\sigma_1}{n_1 - 1} + \frac{\sigma_2}{n_2 - 1}}}$$

where μ, σ and n are the mean, standard deviation and sample size of each data set.

3.2 Null distribution

We are then going to generate a distribution of what the t-values should look like if we assume that there is no difference between the 2 sets of data (also called the null distribution of the distribution under H_0).

To do that we need to:

- Pool the data of the 2 groups.
- Randomly assign the data to 2 temporary groups.
- Compute and store the t-value associated to the difference between those 2 temporary groups

This process is then repeated a large number of time i.e at least a 1000 times (use a for loop).

You can then compute the p-value associated to your initial t-value by counting the proportion of t-value from the null distribution that are bigger than your initial t-value.