Advanced brain imaging methods: Tutorial 1

Remi Gau

24th February 2014

1 GLM: 2 samples t-test

1.1 Having a look at the data

The file 'T-Test.mat' contains simulated data from 2 different groups A and B presented with two different condition from an experiment. You can load the content of this file into matlab by double cliking on it or by typing:

```
load T-Test.mat
```

You should now see two new variables, Y_A and Y_B, in the matlab workspace. You can plot the distribution of the group A by typing `hist(Y_A,x)` where x is the number of bin you want your distribution to have.

1.2 Design Matrix

If you want to peform a two-sample t-test on this data using a GLM, what should the design matrix (X) be? Create this design matrix in matlab. For this you can use the functions:

- `ones(a,b)` which will create a matrix with a lines and b columns containing only ones,
- `zeros(a,b)` which will do the same but with zeros,
- `repmat(x,a,b)`which will repeat the matrix x, a times vertically and b times horizontally,
- you can concatenate vertically 2 vectors A and B into a new vector C like this `C=[A;B]` ,
- you can concatenate horizontally 2 vectors A and B into a new vector C like this `C=[A,B]` ,

You can display your design matrix using the `imagesc` function. If you want to have the design matric in white and black type `colormap('Gray')` before displaying it. You can add a color bar to your figure with `colorbar` .

1.3 Estimating Beta

The ordinary least squares method gives the following unbiased estimator of β:

$$\hat{\beta} = (X^T * X)^{-1} * X^T * Y$$

where X is the design matrix and Y the concatenated data.

What is the matlab code equivalent to this equation that will compute β? Remember that in matlab:

- to get the transpose X^T of a matrix X, you can simply type `X'`,
- to get the inverse X^{-1} of a matrix X, you can simply type `inv(X)`.

1.4 Residual and statistics

You can now compute $\hat{Y}$ the predicted value for each subject with the following equation:

$$\hat{Y} = X\beta$$

The residual for each subject (i.e the errors of your model) can then be computed with this equation:

$$\varepsilon = Y - \hat{Y}$$

You can have a look at the errors to make sure they are normally distributed using the `hist` command or, even better if you have acces to matlab statistical toolbox, the `normplot` command.

Using the function `std` , compute the σ_ε, the standard deviation of the errors, and then you should be able to find the t-value with the following equation:

$$t = \frac{c^T\beta}{\sqrt{\sigma_\varepsilon^2 c^T inv(X' * X)c}}$$

where c is a vertical vector that defines the contrast you want to test.

To get the square root of a number a, you can just do `sqrt(a)` and to get the square of a, typing `a^2` will do the job.

Once you have the t-value the following code should give you the p-value associated to it:

```
% degrees of freedom: depends only on
% the rank of the design matrix (X)
df       = rank(X)-1;

% degrees of freedom of the error ;
% Y is a vector containing all the data
% 'length' tells us how many values Y contains
dferror  = length(Y) - df - 1;

% the p value is found via the cummulative distribution
% function (cdf) of a student distribution
% requires access to matlab statistical toolbox
p = tcdf(-t, dferror)
```

2 GLM: compare 2 regression models

2.1 Having a look at the data

The file 'Regression.mat' contains simulated data of a subject's reaction times (Y) when exposed to a 100 different levels of our experimental VI (x) was. You can load the content of this file into matlab by double cliking on it or by typing:

```
load Regression.mat
```

You can plot the distribution of the data by typing `scatter(x,Y)` .

2.2 Design Matrix

We want to fit two linear regression lines through this data, to compare two different GLM to explain those data: The first one has no constant term and can be written:

$$Y = \beta_1 x$$

And the second is:

$$Y = \beta_1 x + \beta_0$$

What are the design matrices X and X_{NoCst} of these two GLM ? Create and display the 2 design matrices with matlab.

2.3 Betas, residuals and RSS

Once you have the 2 matrices, compute for each model, just like for the exercice on the 2 samples t-test:

- the *beta* (β and β_{NoCst}),
- the predicted values ($\hat{Y}$ and $\hat{Y}_{NoCst}$),
- the residuals (ε and ε_{NoCst}).

Comparing the normplot of the residuals of each model should already give you an idea which model is better. It can be even more obvious if you plot the data and the predicted value of each model on the same figure, like this:

```
figure('name', 'Model with no constant: data and predicted')
hold on
scatter(x,Y,'b')
scatter(x,Y_hat_NoCst,'r')
```

You can compare the two models by performing an F-test using the extra-sum-of-squares principle. First, compute for each model the residual sum of squares:

$$RSS = \sum_i^N (\varepsilon_i^2$$

where $\bar{\varepsilon}$ is the mean of the residuals.

Remember that `mean(A)` will compute the mean of the elements of the vector A, similarly `sum(A)` will give the sum of the elements of A and finally you can get the a vector B where each element is the square of the corresponding elemenet in A by doing `B=A.^2` .

The statistics used to compare the two models (to know if adding a constant to the model makes a difference) is the following:

$$F = \frac{RSS_{NoCst} - RSS}{RSS};$$

You can calculate the p-value associated to this F value with the following code:

```
p  = 1 - fcdf(F, rank(X)-rank(X_NoCst), length(Y)-rank(X))
```

3 HRF: creating it

SPM uses the function `spm_hrf(dt)` to create an HRF, with *dt* being the temporal resolution. Uses this function to create an HRF with a 0.01 second temporal resolution.

The HRF is usually modelled by a weighted sum of gamma function. A gamma function can be described by the following equation:

$$h(t) = \frac{t^{n-1}}{\lambda^n (n-1)!} e^{-t/\lambda}$$

A gamma function is given by the following code:

```
g = ((t).^(n-1)) / (lambda^n*factorial(n-1)) .* exp(-(t)/lambda)
```

First, create a 'time' vector, t containing all the values from 0 to 32 seconds in steps of 0.01 second (e.g the code `0:1:10` would give you all the values from 0 to 10 in steps of 1). Then create a first gamma function h_1 with $n = 4$ and $lambda = 1$, a second one h_2 with $n = 8$ and $lambda = 1$ and a third one h_3 with $n = 16$ and $lambda = 1$. Finally get the HRF by making the weighted sums of those functions as follow:

$$HRF = c_1 h_1 + c_2 h_2 + c_3 h_3;$$

Try different values of c_1, c_2 and c_3 and plot the results to try to find some of the values that SPM uses to create a typical HRF (hint: start with $c_1 = 1$ and a negative c_2).

4 More on the HRF: basis set

Most of the time you want to model the HRF but also its temporal and dispersion derivatives. The 3 of them together can then be used as an informed basis function set to model the BOLD signal. The following code will produce one of those basis set:

```
% Temporal resolution in seconds of the
% informed basis set you are going to create
xBF.dt = .1;
xBF.name = 'hrf (with time and dispersion derivatives)';

% Length of the HRF in seconds
xBF.length = 33;
xBF.order = 1;

% Creates the informed basis set
xBF = spm_get_bf(xBF);
```

The structure `xBF` contains the shape of the different basis function under the field `bf` . For example `xBF.bf(:,1)` contains the shape of the canonical HRF and `xBF.bf(:,2)` the temporal derivative. Using the `plot` function plot the 3 different functions.

Then you can try different weighted sum or difference of those functions to see how you can model different BOLD responses with this basis set. For example:

```
2*xBF.bf(:,1)-0.5*xBF.bf(:,2)
```

5 Concolving with the HRF

Create a basis function set with a temporal resolution of 1 second.

Then, using the functions `ones` , `zeros` and `repmat` , create a 'time' vector made of 8 repetitions of 16 zeros followed by one 1. This is the onset vector: a value of 1 means that at this instant in time a stimulus was presented. You can view this vector using the function `plot` or the function `stem` .

You can then convolve this vector with the HRF and its temporal derivative using the `conv` function. For example.

```
ConvolvedRegressor = conv(HRF, OnsetVector);
```

Plot the results of both convolution.

6 Doing a GLM on real fMRI data

Load the file TimeCourseA1.mat which contains the time course of the BOLD signal acquired from the left and right primary auditory cortices during an experiment where subjects where passively listening to words. The TR for this experiment was 7 seconds. 96 acquisitions were made in blocks of 6. Successive blocks alternated between rest and auditory stimulation, starting with rest. Auditory stimulation was bi-syllabic words presented binaurally at a rate of 60 per minute.

Do a GLM analysis for each of those time courses:

It is always good to have a look at the raw data.

The following code will give you the onset vector for this experiment:

```
Onsets = 6:12:84;
OnsetVector = zeros(length(TimeCourseLeft),1);
for i=1:length(Onsets)
    OnsetVector([Onsets(i):Onsets(i)+5])=1;
end
```

Convolve this onset vector with the canonical HRF from the basis set.

Create a design matrix with this convolved regressor. Should you add a column to that matrix to model a constant?

Compute the β, the predicted values and the residuals for this model.

Plot the predicted and the raw data on the same figure.

Plot the normplot of the residuals.

You can also check if there is any autocorrelation in the residual with the following code:

```
[AutoCorrFunc, lags] = xcorr(Residuals, 'coeff');
plot(lags, AutoCorrFunc)
ylabel('Autocorrelation')
```

Given those different graphs, can you think of simple ways to improve your model? You can then do an F test to see if those changes improved the model.