BCI practical course : P300 Visual/Matrix Speller

Jason Farquhar (coordinator)
<J.Farquhar@donders.ru.nl>
Loukianos Spyrou (Teaching Assistant)
<l.spyrou@donders.ru.nl>

Learning Goals – Last Time: Imagined Movement

- Know how to:
  - Build a multi-block experiement
  - Collect labelled data during the callibration phase of an experiment for later classifier training
  - Train an Event Related Spectral Pattern (ERSP) classifier using the saved data, and the example ERSP classifier training code
  - Build a continuous feedback testing block, with
    - Feedback display
    - Data signal processing and classifier application
  - Speed-up Matlab drawing by making a plot and drawing objects once, and thereafter setting properties on a handle to the drawn object

Learning Goals : Visual Speller

- Know how to:
  - Present complex parallel stimuli
  - Perform sequence decoding for multiple different sequences

Today's Plan

- Brain Test: ERP viewer & IM BCI
- Hands-on : Visual Speller 1 – Calibration Block

break

- Hands-on : Visual Speller 2 – Classifer training
- Hands-on : Visual Speller 3 – Testing Block

break

- Brain-test : Visual Speller

Hands on : Visual Matrix Speller

Experiment Task:

- Build a complete visual matrix speller based BCI experiment consisting of 3 blocks:
  - 1) Training/Calibration Block : where the user is presented with matrix speller stimuli and an instruction on which target to attend to
  - 2) Classifier Training Blocks : where the saved labelled data from the calibration block is used to train an ERP classifier
  - 3) Testing Block : where the trained classifier is used predict which symbol the user is attending to and at the end of the sequence this prediction is used to generate feedback

Hands-on : Matrix Speller Calibration Block

Experiment Task - stimuli

- In 5 sequences of with 5 repeititions epochs
  - (N.B. A repetition is one complete set of stimulating all rows and all columns)
- Start a sequence by displaying the symbol matrix with the target symbol highlighted in green for 2 seconds.
- Clear the cue and display the matrix
- Loop over epochs in the sequence and
  - Highlight the indicated **row** or **column** for 100ms as determined by the epoch count and this sequences stimulus code
  - display the unhighlighted matrix for 100ms before moving to the next epoch
- Initially: highlight all rows for 5 reps, then columns for 5 reps
- Wait for user key press to move to the next sequence
- After the last sequence, display a thankyou message

Hands-on : Matrix Speller Calibration Block

Experiment Task – signal processing

- For every **epoch**, i.e. point where a row/col is highlighed, record 600ms of data annotated with whether the current sequences target symbol was highlighted at this time or not
- When the block is finished save the saved annotated training data

Useful Functions : `initGrid`

- *`hdls`*`=initGrid(`*`symbols`*`)`
  - Create a figure with the strings contained in the cell-array of strings *symbols* are displayed in the same shape as that of *symbols,*
    - i.e. If symbols={4 x 3} them the figure has a matrix of 4 rows by 3 columns etc.
  - Return the handles to the text objects in hdls.
    - Note: hdls has the same size as symbols, so *hdls*(i,j) refers to the text object containing *symbols*(i,j)
  - Note – to change the text color use:

    set(hdls(i),'color',[r g b])

Extra Credit

1)interleave row and col stimulus

- but ensure you always do all rows then all cols, i.e. Don't do row1, col3, row5 etc.
- this ensures the target-to-target interval is large which maximises the strength of the generated ERP

2)Test the timing quality of your stimulus, i.e.

- Are the flashes exactly 200ms apart?

- Modify your code to reduce this timing jitter
  - Hint : allow for Matlab run time when sleeping...
  - Useful function : getwTime() -- get current time in seconds with ns accurate clock

Hands on : Classifier Training Block

Experiment Task

- Load the calibration data saved previously
- Pre-process and train an ERP classifier
- Save the trained classifier for later

Key functions

`clsfr=buffer_train_erp_clsfr(data,devents,hdr,...)`

- train a linear classifier on the frequency power spectrum of the data
- `data, devents` – are data and associated events as output by buffer_waitdata.
- devents.type is used as the unique label for the class of data

Useful Options to change the signal-processing pipeline:

- capFile – cap file to use, e.g. 1010
- spatialfilter – type of reference to use, e.g. 'CAR','slap'
- freqband – frequency range used for classifier training
- badchrm – do we do bad channel removal?
- badtrrm – do we do bad trial removal?

Hands on : Matrix Speller Feedback

Experiment Task -- stimuli

- Display some instructions for the user
- Run the same type of stimulus generation as for the calibration block, **except**
  - No initial target symbol display
  - Prediction display at the end of the sequence
    - Highlight the predicted target letter in green for 2 seconds

Hands on : Matrix Speller Feedback

Experiment Task – signal processing

- Get 0-.6s data every time a stimulus event happens
- Apply the trained classifier to this to get a classifier prediction
- At the end of the sequence identify the most likely target symbol from the set of classifier predictions and the knowledge of the stimulus sequence

Notes: Decoding sequences

- For each epoch;
  - the stimulus sequence says if that symbol was stimulated or not
  - the classifier gives a predicts if that epoch was an attended stimulus event or not
- If the classifier was perfect then **for the target symbol** these 2 sequences would be the same
- For an error prone classifer, then the symbol with the stimulus sequence most similar to the classifier predicted sequence is most likely the target symbol

Notes: Decoding sequences (2)

- Thus to identify the likely target letter:
  - Compute the **similarity** between the classifiers sequence of predictions (f) and each symbols highlight sequence;
    - Similarity could be: correlation, inv-distance etc.
    - Suggest use a simple **inner-product;**
      - im(symb) = Σ_t highlight(symb,t)*f(t) = highlight(symb,:)***f**
    - as this can be shown (for expionential family classifiers) to give the same prediction as gives the same result as correlation, i.e.
  - The symbol with the highest similarity is the most likely target symbol

Notes : Process architecture

- Decoding the correct letter requires
  1) Knowledge of the sequence of symbol highlights
     - Readily availabe in the stimulus process
  2) Knowledge of the sequence of classifier predictions
     - Readily available in the signal processing process
- We have 2 choices where this combination takes place
  1) Collect the per-stimulus classifier predictions in the stimulus code
  2) Collect the symbol stimulus sequence information in the signal-processing code, combine and send the symbol prediction back to the stimulus code
- The first is simplier, so you should probably use it.
- The second is useful if someone else is writing the stimulus code (they don't need to understand **anything** about how the signal processing works)

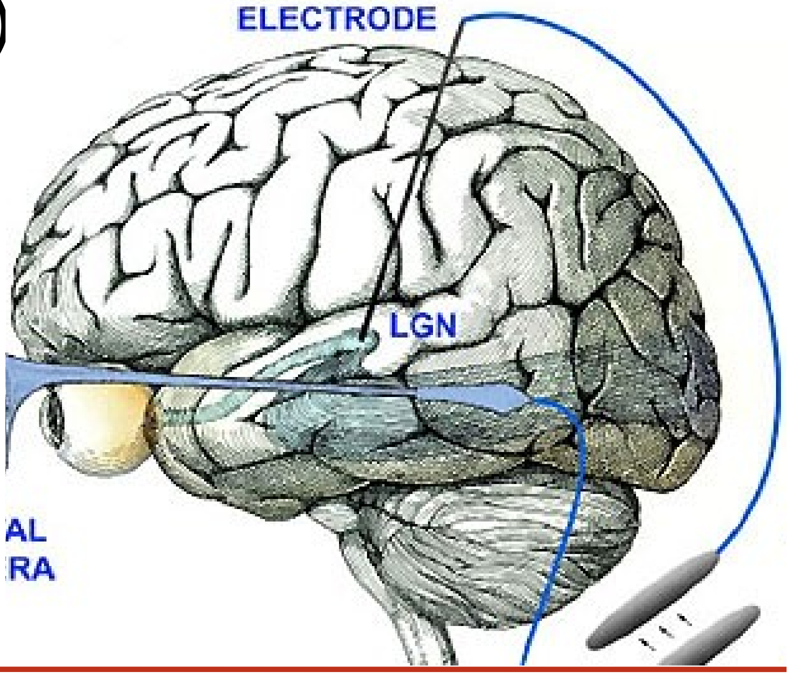
Key functions

[*f*,*fraw*,*p*]=buffer_apply_erp_clsfr(*X*,*clsfr*)

- Apply the ERP pre-processing and trained classifier stored in *clsfr* to the input [channels x time] data in *X*
- *f* is the classifiers output decision value
  - decision value is a real number where f<0 predicts class -1, f>0 predicts class +1
  - combine decision values from different data by simply adding them, e.g. $f(X_a \& X_b) = f(X_a) + f(X_b)$
- *p=Pr(+|X)* is the estimated probability of the positive class
  - Pr(+|X) = 1/(1+exp(-f(X))) for **logistic regression**

Brain Test

- Test you system using a real participant
- Hint: For a quick test that everything is working, you can:
  - use a slow stimulus (400ms ISI)
  - 'blink' for each target event
  - Use a single repetition



Summary

- BCI is fun!
- Evoked experiments are;
  - Fiddly – because you need to get the stimulus right!
  - Fiddly – because you need to get the timing right!
  - Fiddly – because you have to do sequence decoding..