BCI practical course : P300 Visual/Matrix Speller

Jason Farquhar <J.Farquhar@donders.ru.nl>

Radboud University Nijmegen

Learning Goals – Last Time: Imagined Movement

- Know how to:
  - Build a multi-phase 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

- Hands-on : Scanning Visual Speller – Calibration Block
- Hands-on : Scanning Visual Speller – Classifer training

break

- Hands-on : Scanning Visual Speller – Testing Block

Break

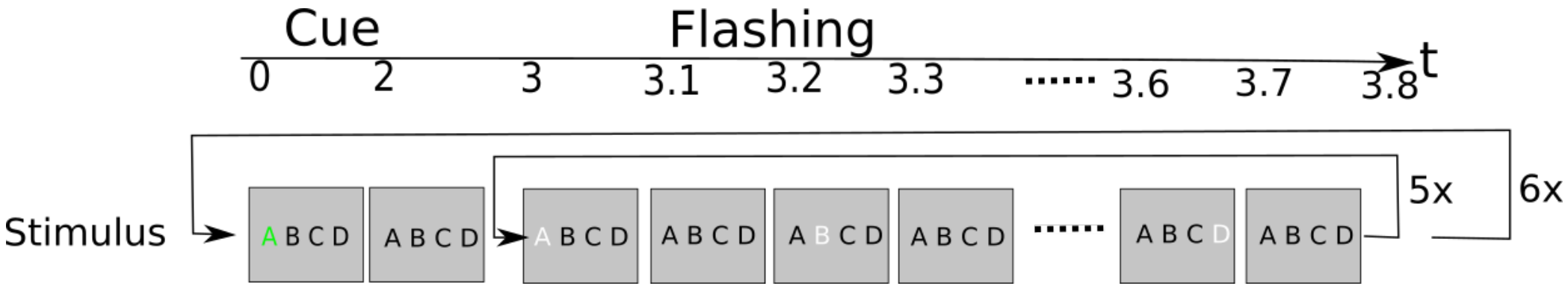
- Hands-on : Row/Col Visual Speller – Calibration
- Hands-on : Row/Col Visual Speller – Training
- Hands-on : Row/Col Visual Speller – Testing

Hands on : Visual Matrix Speller

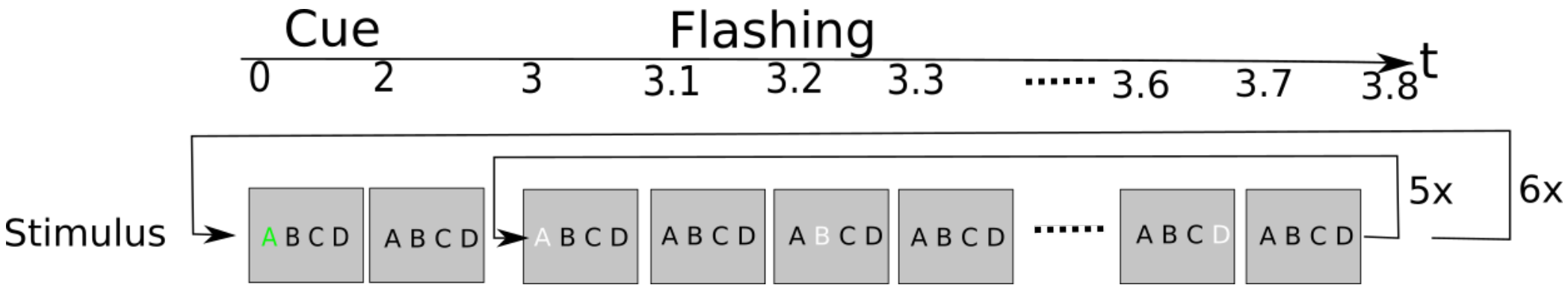
Experiment Task:

- 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 : Scanning Visual Speller Calibration Block Stimuli



Hands-on : Scanning Visual Speller Calibration Block Stimuli



- In 5 sequences of with 5 repeititions epochs
  - (N.B. A repetition is one complete set of stimulating all symbols)
- 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 **symbol** for 100ms
  - display the unhighlighted matrix for 100ms
  - move to the next epoch
- Wait for user key press to move to the next sequence
- After the last sequence, display a thankyou message

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])

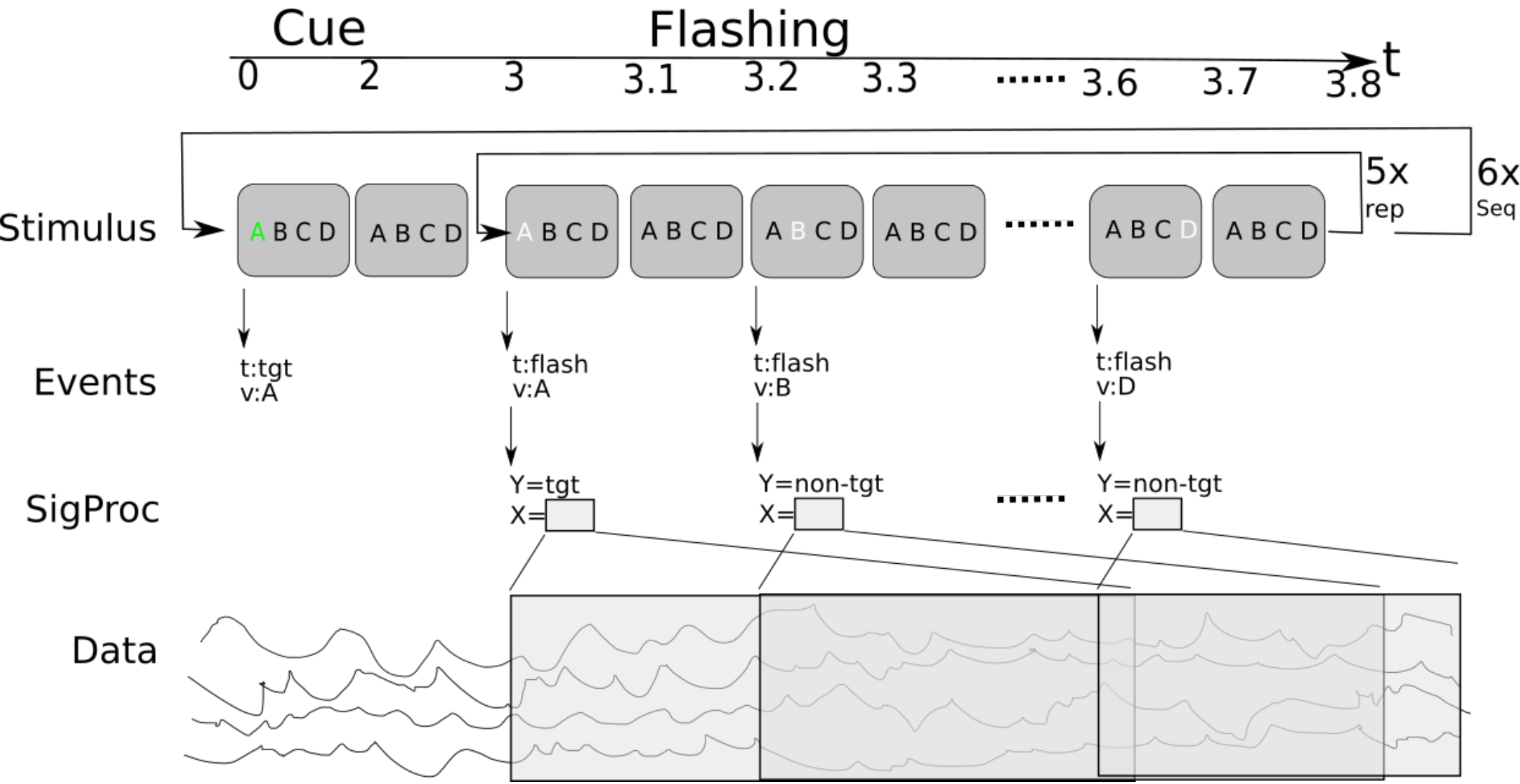
Hands-on : Scanning Visual Speller Calibration Block Signal Processing

- Question: What information does the signal processor need to gather calibration data to train the classifier?

Hands-on : Scanning Visual Speller Calibration Block Signal Processing

- Question: What information does the signal processor need to gather calibration data to train the classifier?
- Answer:
  - 1) When a ‘flash’ started
  - 2) How long the flash brain response should last
  - 3) For each flash what ‘class’ of brain response it should be.
    - Specificially, for the visual speller if this was a ‘target’ flash or not

Hands-on : Scanning Visual Speller Calibration Block Signal Processing



Hands-on : Scanning Visual Speller Calibration Block Signal Processing

- For every **epoch**, i.e. point where a symbol is highlighed:
- In the Stimulus Presentation:
  - Send a target event which says if the current flash is a target-symbol flash or not
- In the Signal Processor
  - Catch the target event which says when the flash started
  - Record 600ms of data (expected P300 duration)
  - When the block is finished save the saved annotated training data

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

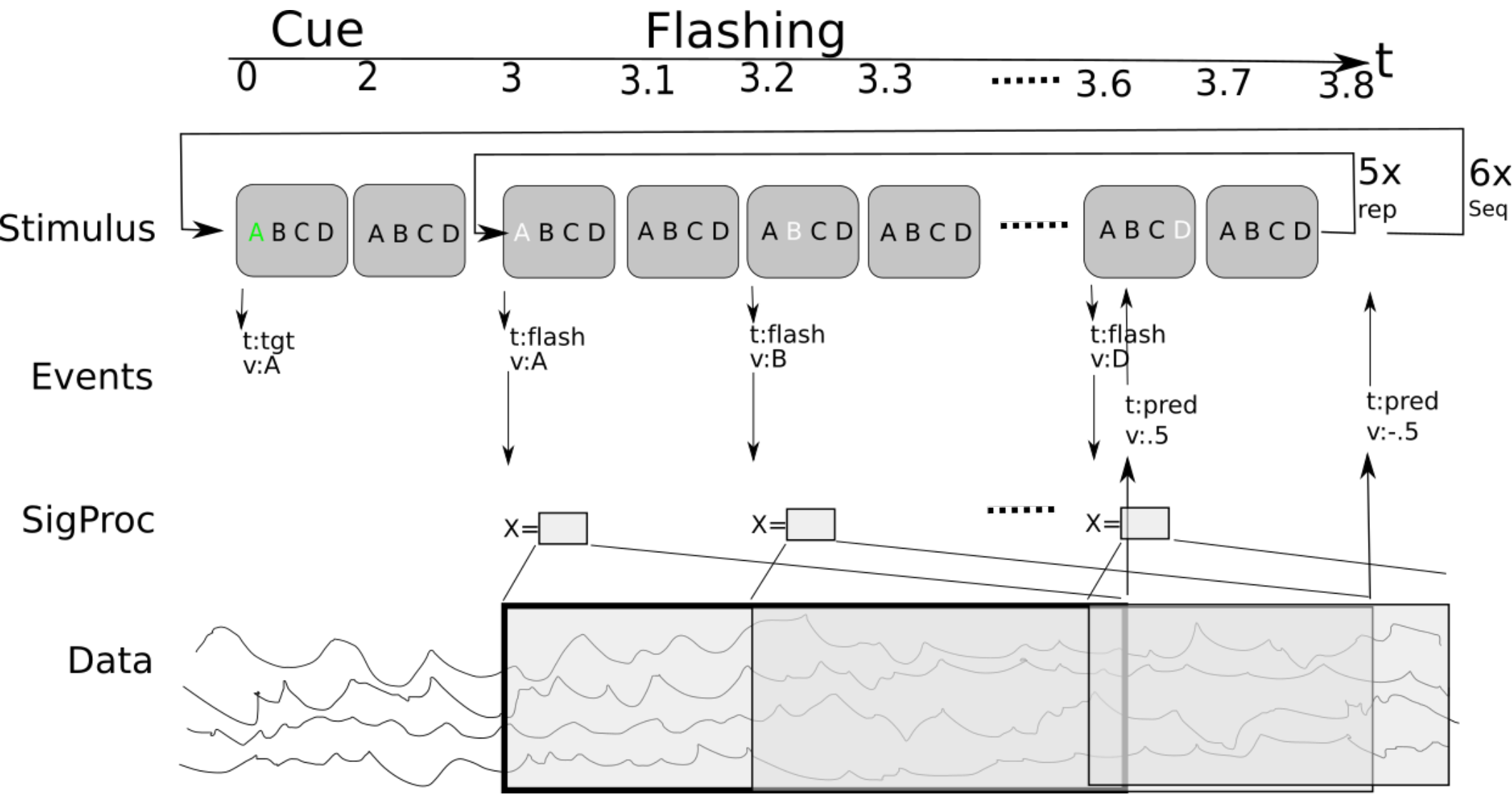
Useful Options to change the signal-processing pipeline:

- 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 : Scanning Visual Speller Feedback Stimulus

- As for 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 : Scanning Visual Speller
Feedback Signal Processing



Hands on : Scanning Visual Speller Feedback Signal Processing

- Get 0-.6s data every time a stimulus event happens
- Apply the trained classifier to this to get a classifier prediction
- Send an event with the classifier prediction
  - (Be sure to use the same sample number so can be connected to the orginal flash)

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**

Hands on : Scanning Visual Speller Feedback Decoding

- Now we have stimuli running and the classifier generating flash predictions.
- How do we combine these to identify the most-likely letter?

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)

- Compute the **similarity** between the classifiers sequence of predictions (f) and each symbols highlight sequence;
- The symbol with the highest similarity is the most likely target symbol

- Many possible similarity measures:
  - correlation, inv-distance...
  - Suggest use a simple **inner-product;**

$$ip(symb) = \sum_t highlight(symb,t)*f(t) = highlight(symb,:)*\mathbf{f}$$

  - as this can be shown (for exponential family classifiers) equivalent finding the maximum likelihood solution

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)

Hands On: Visual Speller Feedback Stimulus

Modify your FeedbackStimulusPresentation to give a correct feedback based on the classifier outputs.

You will need to:

1) Record information about what symbols were flashed in what order.
2) Catch the classifier predictions sent by the signal-processor (hint: buffer_newEvents)
3) Align the symbol-flash information, with the classifier prediction information.
4) Compute the similarity betwen the vector of classifier predictions.
   - If you have the classifier predictions in a vector: `f`=[nFlash x 1],
   - And the true flash sequence in a matrix, `stimSeq` =[nFlash x nSymbols]
   - Do this with: `sim = f'*stimSeq;`
5) Identify the most similar symbol. (hint: [ans,simIndex]=max(sim));
6) Give the feedback to the user.

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..

Hands-on : row/column Visual Speller Calibration Stimulus

- Normal matrix-speller shows stimuli on a grid and highlights the letters in groups of same-row or same-column.
- Modify your Stimulus presentation to work in this way also:
  - That is, instead of flashing a single character flash a whole row or column.
    - (hint: set row 2 to green: `set(h(2,:),'color','g');`
  - Modify your 'flash' event code so it's value is true if the target is in the highlighted row or column
- Also: normal matrix-speller randomly choose which row-column to highlight – as this tends to generate stronger brain responses.
  - Modify your stimulus presentation such that the row/col order is randomized.

Hands-on : row/column Visual Speller Calibration Signals

- This should be identical as for the scanning mode if you have set your events correctly.

Hands-on : row/column Visual Speller Classifier Training Signals

- This should be identical as for the scanning mode if you have set your events correctly.

Hands-on : row/column Visual Speller Feedback Signals

- This should be identical as for the scanning mode if you have set your events correctly.

Hands-on : row/column Visual Speller Feedback Signals

- Modify your FeedbackStimulus code in a similar way as your calibration code to:
  - ‘flash’ symbols in groups of same row or column
  - Randomly choose which row/column to flash at any time

- If you have sent your events correctly, and recorded the ‘flash’ sequence of each character correctly then:
  - the decoding component of the stimulus should be identical as in the scanning mode (i.e. catching prediction events, computing the similarities, and showing the feedback)

Congratulations!

You have just implemented all the major components of a matrix speller BCI

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..