Motor Cortex EEG Signal Decoding for Playing Simplified Pong

BCI Practical - Final Report

Tim Bergman (s3012050), Josh Ring (s4597222), Farhad G. Zanjani (s4454251)

Statement of Contribution

Project

Farhad & Josh:

- Dataset evaluation (4 in total, 2 in final analysis)
- Algorithm evaluation (Discriminant Classifier family, Random Forest)

Tim:

- Experiment Implementation, Data Collection
- Demo Implementation

Report

- Section 1: Farhad, Josh & Tim
- Section 2: Tim (2.1 - 2.3), Farhad (2.4), editing by Josh
- Section 3: Farhad, editing by Josh
- Section 4: Josh
- Section 5: Farhad
- Section 6: Tim

Each member contributed, to some extent, to all aspects of the project/report.
Please note: All members feel there was equal contribution by all.

1. Introduction

We present the design and development of a Brain-Computer-Interface (BCI) system, allowing for the control of a simplified computer Pong game using motor cortex EEG signals. This game was designed to demonstrate real-time interaction between a human and a computer via the decoding of brain signals, the primary goal of any BCI system. Furthermore, an attempt at implementing a 'zero-training' classifier was made. As training and calibration take a long time, and must generally be performed each time the BCI is to be used, it would be advantageous if a classifier could use data from previous sessions or even previous participants and apply it to a current session, drastically reducing the training/calibration phase or even removing it entirely.

Previous work in this direction includes importance-weighted cross-validation [Sugiyama et al.], whereby training exemplars that most closely match the exemplars encountered during testing are assigned greater importance during classifier training, which in a practical setting requires online comparison and reweighting of training data. Another, very different approach, involves so-called 'reinforcement signals', whereby a classifier is given feedback as to the accuracy of its output, resulting in a form of online supervised training [Llera et al. 2012]. The authors recommend the use of the error-related negativity, a signal produced by the subject in response to committing an error, as the reinforcement signal; this, however, raises other challenges, such as the robust, subject-independent identification of this signal, or alternatively requires that detection thereof be trained, bringing us back to square one. In the current study, we investigated the use of large datasets; by training a classifier on sufficient data, generalization to other subjects or sessions is desired. We investigated this possibility two-fold: first, we studied the classification of real movements into left and right hands on publicly available datasets for a large number of subjects, and compared our results with other analyses of the same dataset found in the literature. Second, we compared the performance of classifiers detecting non/movement of the hands on data collected from one subject, across several sessions spread over one day.

In this project, we look for a rich feature extraction method and a robust classification method for getting the highest performance, comparing the results obtained from the aforementioned experiments. We used a common and widely used family of discriminative classifiers, namely Linear Discriminant Analysis (LDA), Quadratic Discriminant Analysis (QDA), and Kernel Discriminant Analysis (KDA), and evaluate which one can perform the task with 'zero training', relying only on data recorded from previous subjects or in previous sessions. In implementing a complete BCI system, we obtained an online classification accuracy of approximately 65% in distinguishing non/movement of a player's hands during interaction with a pong game. Furthermore, we are able to distinguish movement between the left and right hands between 29

subjects with around 62% accuracy, showing higher accuracy than some recent papers on the same datasets, although this accuracy still is not high enough for embedding in a real time game.

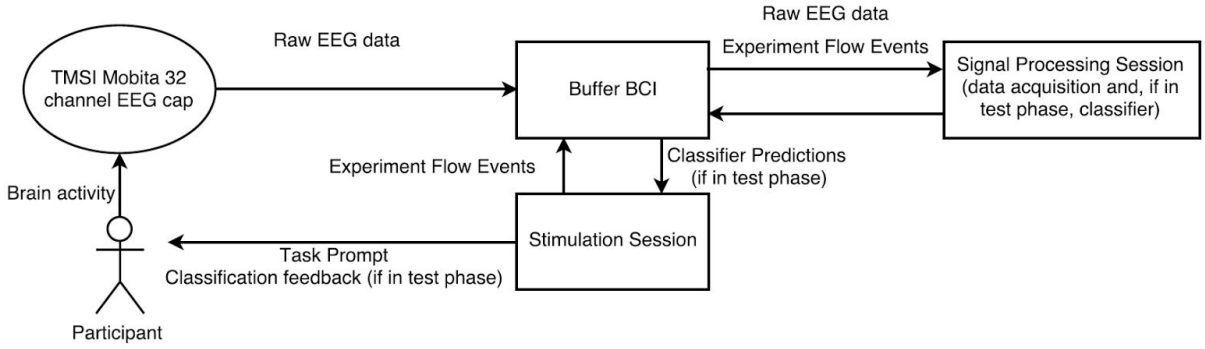
2. Methods

In order to gather and process our own data, we built the BCI system outlined below. We presented the participant with tasks (move hands or not), and took 6 seconds of raw EEG data centered on the onset of the participant's movement (or non-movement, depending on the task). After being pre-processed, a classifier was then trained on this data to distinguish between the two classes (movement vs. no-movement), after which we could use the same BCI architecture to send classification feedback to the user and play pong on-line.

Furthermore, we examined the classifier's applicability in zero-training scenarios by looking at performance between sessions on our own datasets, as well as between subjects using third-party datasets available online, in an offline analysis using multiple classification methods.

2.1 The BCI System

The BCI system we used was set up as can be seen in the following figure:



A water-based 32-channel EEG system was used to capture the participant's brain activity, chunks of which were subsequently sent to and distributed via a BCI buffer. Two Matlab sessions were used for conducting the experiment and gathering data. The first was used for applying stimulation to the participant and creating the experimental flow, with events denoting the start and end of stimulation phases being exchanged across the buffer as well. The other Matlab session handled, depending on the phase of the experiment, the data gathering, signal processing, or training and application of the classifier, when signalled to do so by experimental flow events from the stimulation client. During the test-phase, the signal processing session would send prediction responses from the applied classifier across the buffer to the stimulation session as well, which would, in turn, give feedback to the user.

Materials Used

- **Buffer BCI**: a Matlab-based wrapper for the FieldTrip buffer as written and maintained by Jason Farquhar was used to circulate the data samples and events to and from the various clients necessary for the BCI. See also: https://github.com/jadref/buffer_bci
- **TMSI Mobita water-based 32-channel EEG system**: the Mobita 32-channel water-based EEG device from TMSI was used to gather brain data for these experiments.
- **Matlab R2015b with Statistics and Machine Learning Toolbox**: the Matlab environment had been used for the signal processing and stimulation side. Because we used an LDA as an online classifier, as well as several dimensionality-reduction methods during the exploration of the data, the Statistics and Machine Learning Toolbox was also required.
- **Lindo Ouseph's Pong game**: we adapted a ready-made pong game written by Lindo Ouseph (Research Scholar, Department of Electronics, CUSAT, Kochi, India). See also: http://www.mathworks.com/matlabcentral/fileexchange/45619-pingpong-game
- **Scikit-Learn Machine Learning Module**: a Python-based module we used for offline exploration of the structure of the data.

2.2 Experimental Design

Tasks and Goals

The goal of our experiment was to distinguish between two classes from the participant's EEG data where actual movement was concerned. The two classes we considered were no-movement and movement, with a more fine-grained distinction between the left and right side. The movements we considered were big hand movements (or more precisely: finger movements), as a big part of the motor cortex is dedicated to controlling the hand, our assumption was that we should be able to gain a reasonable brain signal from even the water-based EEG system because of this.

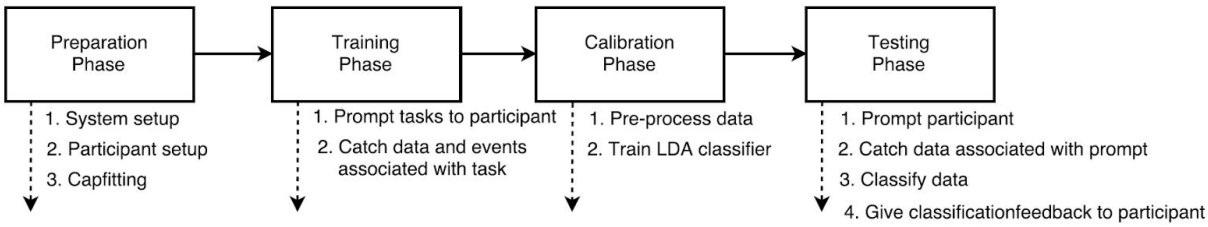
As such, our participants were asked to either engage in exaggerated hand movements (spreading and closing the fingers) for the movement-condition, or to not engage in any movement at all for the no-movement condition. The movement condition was split up between the left and right hands.

Participants

For our own experiments, we considered one participant, a student from the Radboud University, male, aged 24, who provided 6 datasets acquired across several sessions spanning a single afternoon, to investigate within-subject, between-session transferability.

Procedure

The full experiment was broken up over multiple phases, as outlined in the following figure.



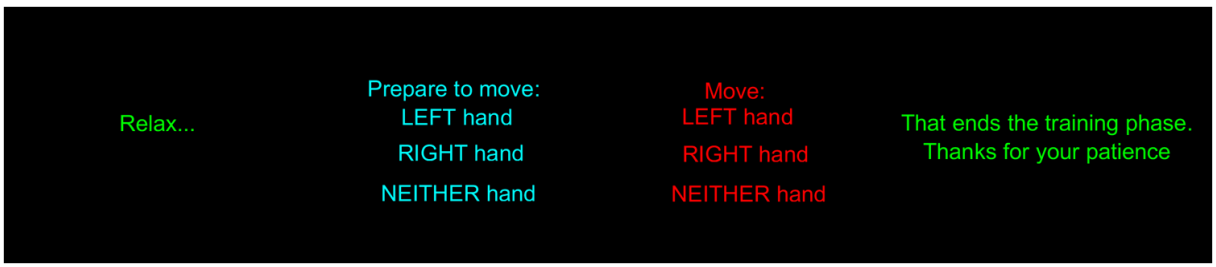
Preparatory phase

The preparatory phase involved setting up the required buffer-BCI architecture (buffer, mobita, event viewer, signal viewer), and included capfitting (setting up the EEG cap's electrodes, fitting the wet sponges, putting it on the participant and making sure we get an adequate signal).

Training phase

The training phase involved the participant being prompted 80 times, 40 times for each class (i.e. no-movement and movement), where the movement class was split into 20 times left and 20 times right. These prompts were presented randomly.

The prompts themselves were structured as follows: first, a 3-second rest phase; second, a 'prepare to move' phase with the required task was presented for 3 seconds as well; finally, a 'move' phase in which the participant was to move as per the prompt was presented for 3 seconds as well. After all the prompts had been completed, the participant was thanked for his or her participation, and the training phase ends. A data acquisition session captured the EEG data across the full 6 second duration of each prepare-to-move plus actually-move epoch, with the corresponding class label (left hand, right hand, no-movement).



Calibration phase

The calibration phase involved the pre-processing of the signals (where signals were cropped to the region of time were were interested in, then detrended, re-referenced, had bad epochs

removed, had a spatial filter applied and had its spectral features extracted. See also: Pre-processing of the Signals below). Afterwards, the LDA classifier was trained to the data with their corresponding labels.

Testing phase

In the testing phase, a similar set-up to the training phase was developed, where a 3-second 'prepare to move' phase was followed with a 3-second 'move' phase. The participant could choose to either move or not. During each of these test-epochs, an identical pre-processing pipeline was used as the one used in the calibration phase, after which the data was passed to the classifier and a class prediction would be given. This prediction would be passed back to the stimulation-session, which would display the prediction.

Three test-phase stimulation sessions were developed:

- **Task-based Evaluation Session**: the first was very similar to the train-phase, in which the participant was prompted with a task he or she had to perform. This would allow for contrasting the intended class with the predicted class and give an estimate of classifier performance.
- **Free-Decision Session**: the second was a free-decision task in which the user would be prompted, and the user him/herself could decide on whether to move or not. This did not allow for measured performance checking, but did implement the free-decision architecture the intended end-product (i.e. the pong game) would have.
- **Pong Game**: the third was the actual single-player pong game, in which a ball would move slowly across the screen, bouncing against the far wall, as well as the top and bottom walls. The goal of the game is to use a slow moving paddle on one side of the screen to intercept and bounce the ball back as often as possible. When a movement prediction was given by the classifier, the direction in which the paddle travelled would invert, giving the player a rudimentary 'switch'-style control scheme over the paddle. If no movement was detected, the paddle would not switch direction.

2.3 Signal Processing

Examined Signals

For BCI, in general two kinds of signals can be used [Wolpaw et al. 2002]:

- **Evoked Responses**: in which an external stimulus evokes an Event Related Potential (such as the P300), and
- **Induced responses**: in which an internal mental state as chosen by the user induces an Event-Related Spectral Perturbation (i.e. a change in the power within a frequency range) (such as the RP).

For our experiment, we decided on using an induced response, namely the mu-band activation of the motor cortex, in order to classify whether the user was (going to) move their hands, and by extension the pong paddle, or not. Due to increases in classification accuracy, we ultimately elected to use a wider spectrum of frequencies (8 - 45 Hz) during the final demo.

Preprocessing

Prior to classifier training, the data underwent several preprocessing steps.

- **Detrending:** During the detrending procedure, slow drifts and arbitrary offsets in the EEG data are removed by computing and subtracting the linear trends for each channel and epoch.
- **Bad channel removal:** In order to achieve between-session and between-subject transferability, we decided not to remove bad channels. For this reason, the capfitting procedure beforehand becomes more important. A disadvantage of this decision is that any incidental bad channels can muddy the signals.
- **Re-referencing:** Because EEG contains a lot of signals from external noise sources, we re-referenced the signal, removing common external signals by subtracting from all channels the average signal over all channels.
- **Bad epoch removal:** In this stage, bad epochs are identified and removed. Some trials might have artifacts such as eye-blinks that have time points with excessively high power. Putting these through a classifier can lead to output mistakes, which is why these should be removed. Specifically, any trials with a variance in power that are 3 standard deviations greater than the mean are considered outliers and are removed. Do note that we did not use bad epoch removal during online classification and testing, only for offline analysis.
- **Spectral feature selection:** The ERSP signal is a change in power over a particular frequency range, such that the raw time-domain features must be converted to a frequency-domain representation. Using Welch's method, a high quality estimate of the signal's power spectrum density for each epoch is computed, after which we only consider the region of interest in a particular frequency range. Any frequency bins that lie outside of our region of interest (8 Hz – 45 Hz, as determined by classifier accuracy) are then discarded.

2.4 Signal Analysis

Visualization

In designing a real-time Brain-Computer-Interface, offline analysis of brain signals is helpful in obtaining an overview of the task, giving insight into appropriate approaches for data preprocessing, feature extraction and the design or choice of classification technique. Due to the time-intensive nature of collecting EEG signals from a sufficient number of subjects, and to allow us to compare our approach to previous work, we first addressed the challenge of distinguishing left from right hand movements in EEG data using a publicly available dataset, prior to collecting our own data via the experiment detailed above.

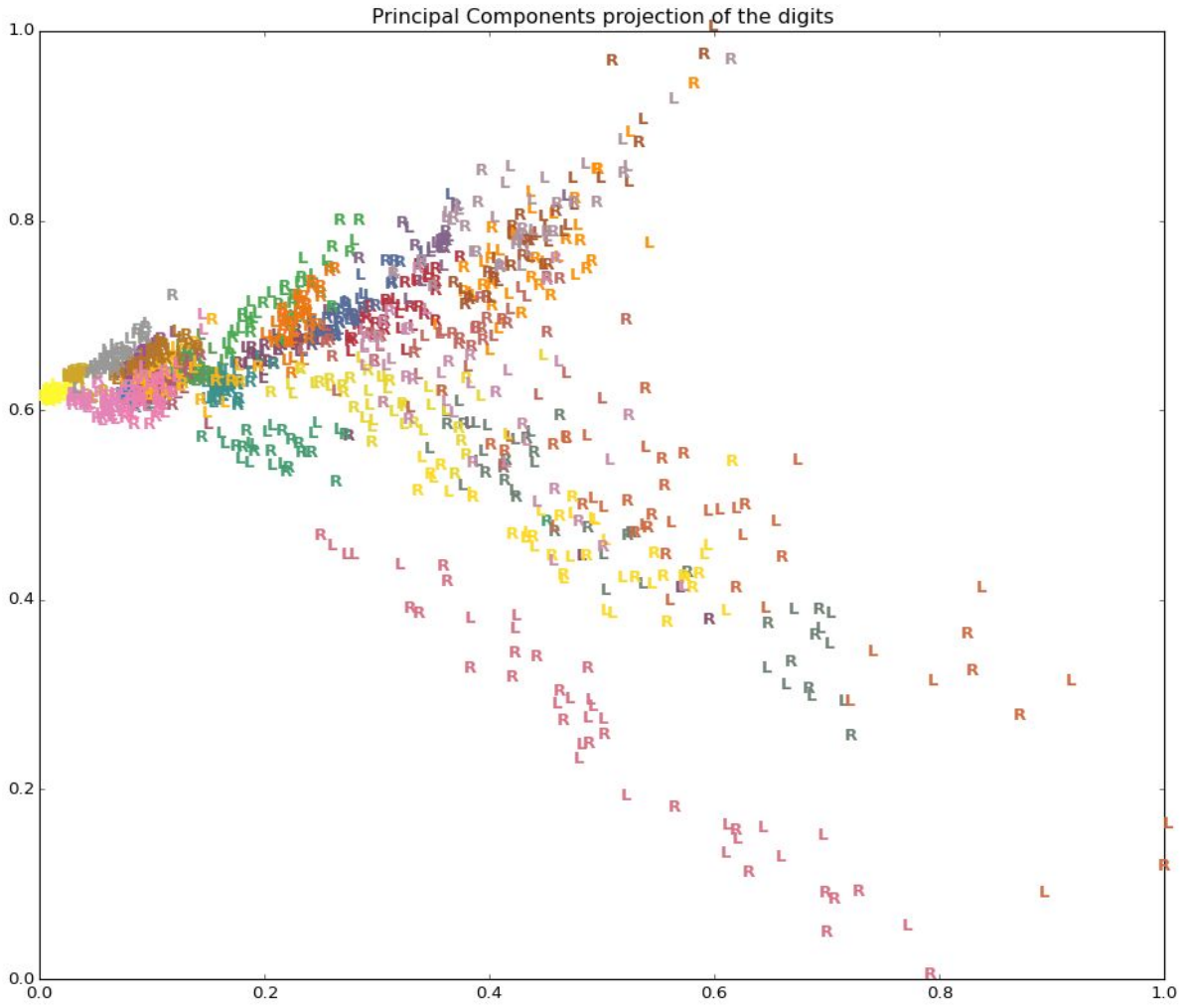


Fig 1 a. 2D visualization of PSD (power spectrum density) along Motor Cortex for 29 subjects – two first principal components – colors represent individual subjects and letters “L” and “R” indicate left and right hand, respectively.

Given the staggering number of subjects and the high quality of the data, the PhysioNet dataset [Schalk & Mellinger, 2010], [Goldberger et al. 2000], consisting of EEG recordings of 109 subjects performing actual and imagined motor tasks involving both hands and feet, is an obvious choice when investigating EEG motor movement classification. In the present study we examine a subset of this data, namely the left and right hand real-movement trials of 29 subjects, graciously provided to us by Dr. Jason Farquhar. When analyzing data in general, and particularly when classifying data, it is often helpful to take a first look at the data using visualization tools, in order to gain insight into the complexity of the problem, such as the distribution of the classes in input space. To this end we used Principal Component Analysis (PCA), a standard and well-established method for reducing the dimensionality of a signal, and t-distributed Stochastic Neighbor Embedding (t-SNE), a newer addition to the data-scientist's toolbox, to project the data onto two dimensions and allow for human-readable plots.

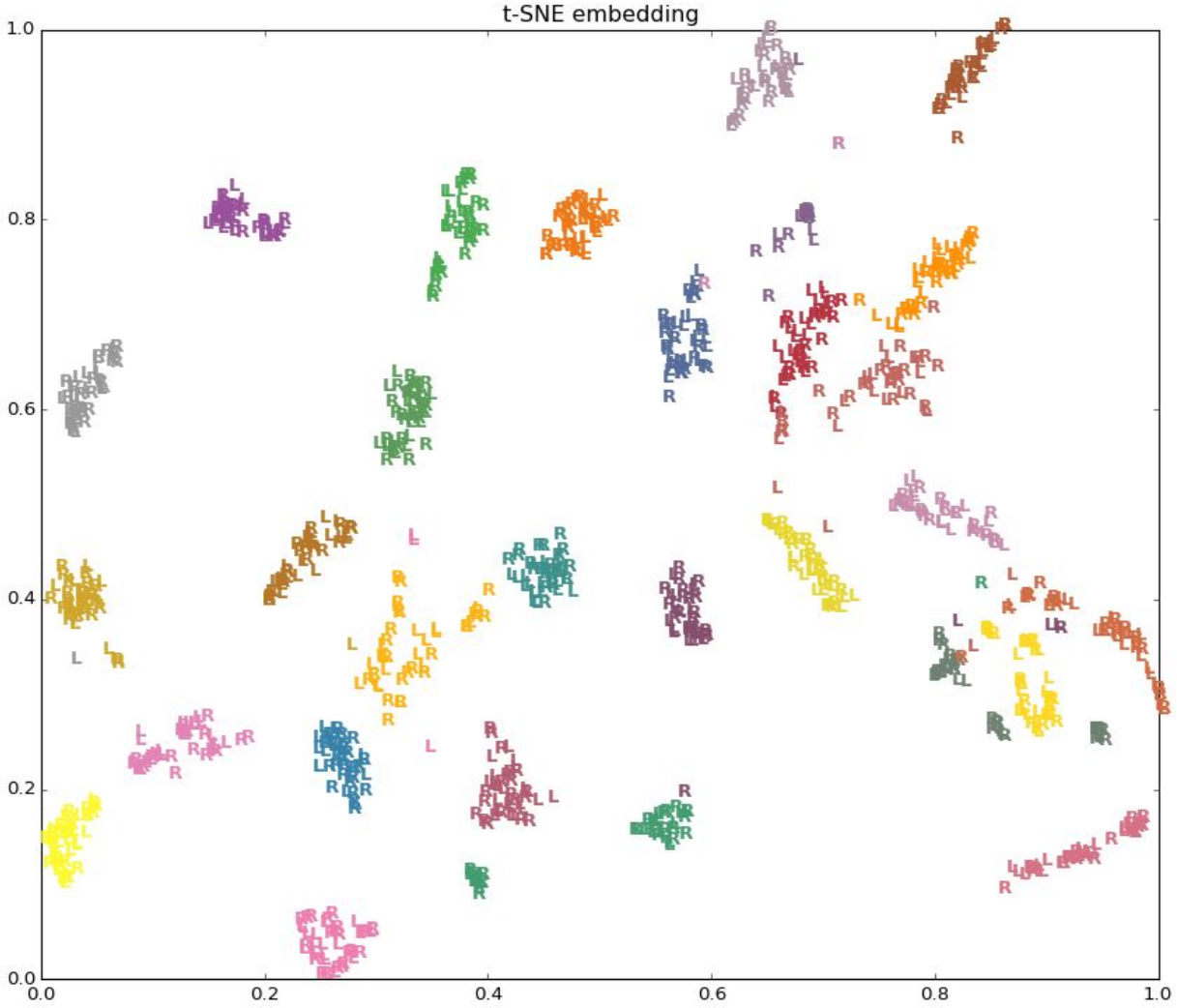


Fig 1 b. 2D visualization of PSD along Motor Cortex for 29 subjects – t-SNE subspace – colors represent individual subjects and letters "L" and "R" indicate Left and Right hand, respectively.

Figure 1 depicts the two-dimensional projections obtained by each method, of the EEG power spectrum decomposition of the data. The different colors denote individual subjects, and each trial is reduced to a single point in the 2D subspace, marked as “L” or “R” according to the hand that was moved. The top plot in Fig. 1 depicts the PCA subspace, consisting of the first and second component, with the t-SNE subspace depicted in the bottom plot. As can be deduced from both these subspaces, somewhat more clearly in t-SNE projection, EEG signals related to motor tasks are very specific to individual subjects. Points are grouped into clusters of like colors, indicating that there are great differences between the data of each subject, but no great differences common to recordings of left and right hand movements. As such, we would expect that global classification of left/right hand movements across all subjects does not achieve brilliant performance, in agreement with results reported by Jason Sleight et al. [Sleight et al. 2009] and Morimoto et al. [Morimoto et al. 2014].

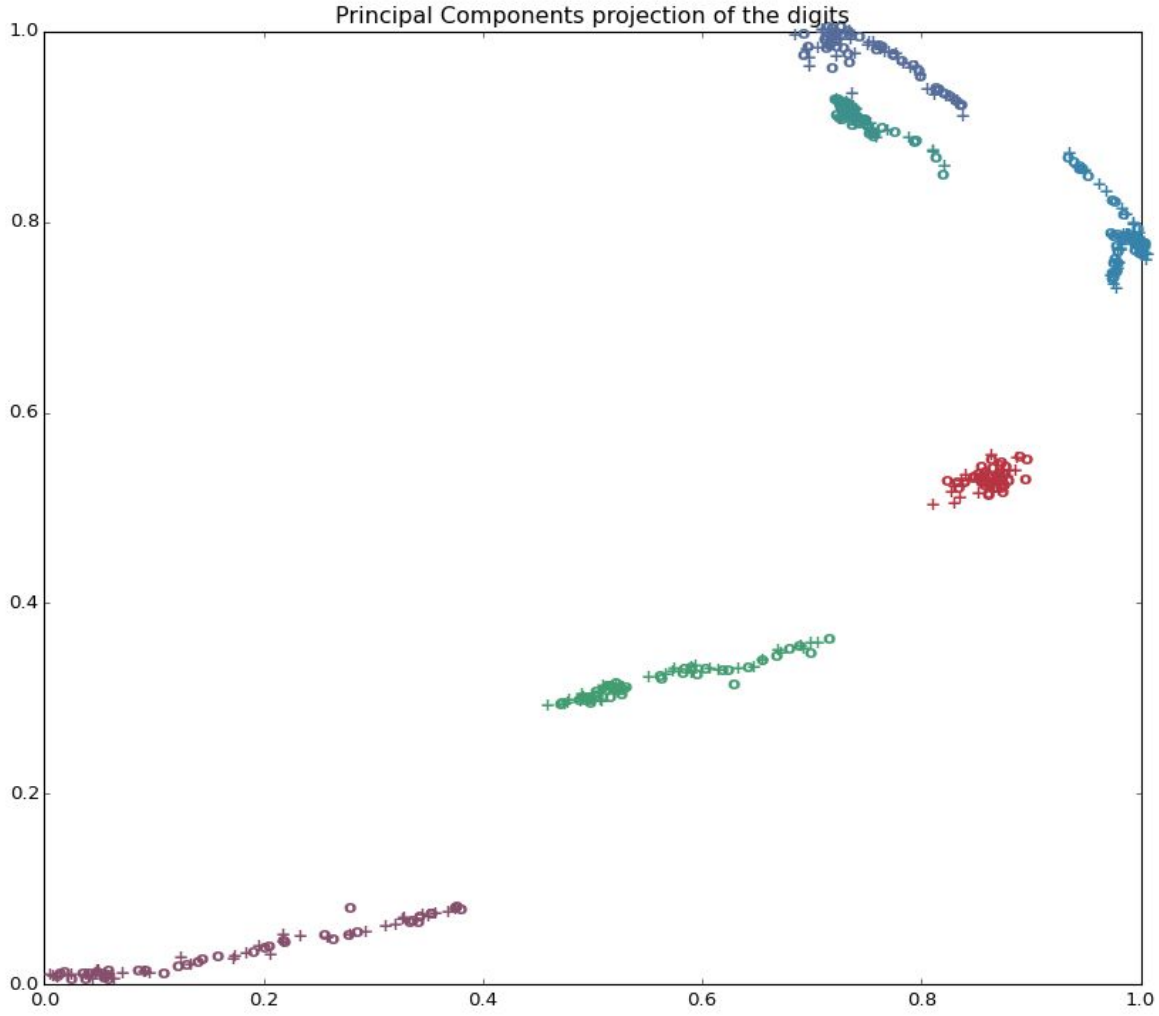


Fig 2 a. 2D visualization of PSD along Motor Cortex of recorded subject – two first principal components – colors represent individual sessions and symbols “+” and “o” indicate non/movement of hands, respectively.

The same dimensionality-reduction techniques were applied to the data we collected using the experimental procedure outlined above. The results of this transformation can be seen in the figures 2a and 2b; once again, individual sessions are easily discriminable, whereas no systematic differences between movement and non-movement trials were identified. This serves as a good illustration of the challenge faced when attempting between-session transfer of a classifier, even tailored to a single subject.

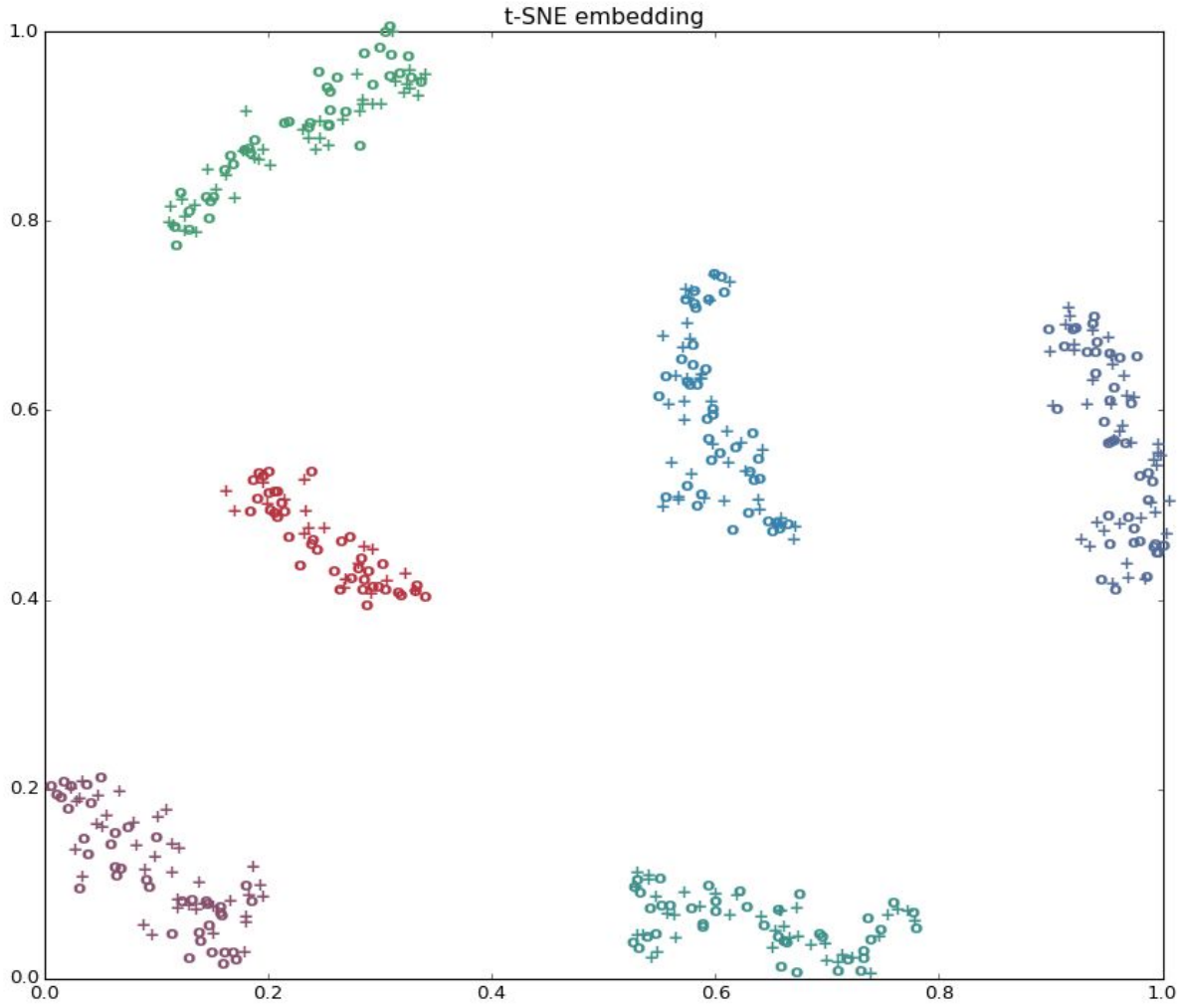


Fig 2 b. 2D visualization of PSD along Motor Cortex of recorded subject – t-SNE subspace – colors represent individual sessions and symbols "+" and "o" indicate non/movement of hands, respectively.

In some respects, a recent report on classifying left and right hand movement [Alomari et al. 2013] using recordings from the PhysioNet database, contradicts our approach and findings in tackling this dataset. The first conflict is that Alomari et al., somewhat inexplicably, band-pass filtered the EEG signal between 0.5 and 90 Hz; given that the sampling frequency of this dataset is 160 Hz, the highest frequency that should be considered is 80 Hz, i.e. half of the sampling frequency. Secondly, Alomari et al. included just 8 electrodes (FC3, FCZ, FC4, C3, C1, CZ, C2,

and C4) as an optimal set, whereas our findings suggest that adding FC1, FC2, CP3, CP4, CP1, CP2, and CPz will improve the classification performance significantly. This subset of 15 electrodes for classifying left and right hand real movements is in agreement with the approach found in the [Morimoto et al. 2014] report. However, it is worth mentioning that Alomari et al. consider only 6 subjects in their paper, while in our analysis we used a larger subset of 29 subjects. This difference in sample size may be the source of conflicting findings in terms of the optimal subset of electrodes. Furthermore, in our analysis we processed only the signals collected during movement onset and completion, discarding recordings from before or after the action, whereas Alomari et al. [2013] also considered features extracted from two seconds of data prior and subsequent to the movement.

Fig. 3 shows the power spectrum of C3 electrode for different trials of left and right hand of subject number 7 of PhysioNet dataset. This subject had the best performance in prediction (shown in Fig. 4) among other and this can be seen by different power spectrum patterns around middle of band width (around frequencies 11~26 Hz).

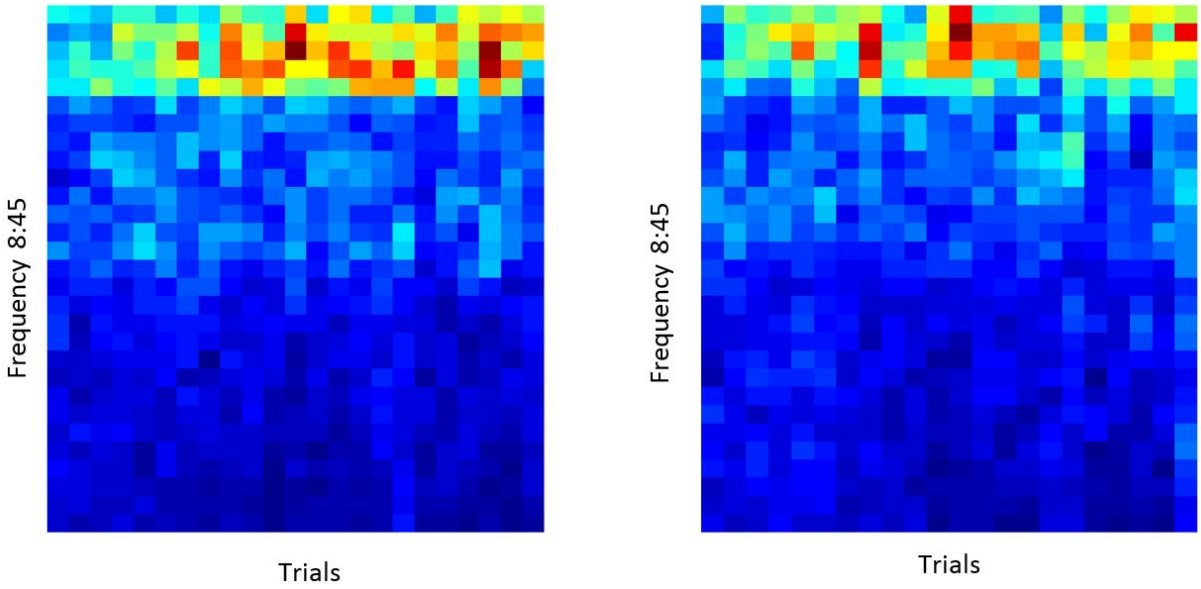


Fig. 3. Power Spectrum visualization of electrode C3 of left hand (left plot) and right hand (right plot) for subject number 7 from the PhysioNet dataset. The difference around 11~26 Hz (rows 11~15 of figures from top) can be seen clearly

The PhysioNet and our own data were preprocessed using the pipeline outlined above, and passed to the classifier as a vector of feature*electrode (15*29).

We investigated three discriminative methods for classification of the EEG signal; Linear Discriminant Analysis (LDA), widely utilized among the BCI community [Lotte et al. 2007], as well as two extensions thereof, Quadratic Discriminant Analysis (QDA) and Kernel Discriminant

Analysis (KDA) [Baudat et al. 2000]. In the following, we give a general outline of the principles underlying these approaches.

Classification

A linear classification model can be formulated in terms of dimensionality reduction; in the case of two classes, if we project the D-dimensional input vector $\boldsymbol{x}$ down to a one-dimensional representation $\boldsymbol{y}$ using weight-matrix $\boldsymbol{w}$, as in $\boldsymbol{y} = \boldsymbol{w}^{\mathrm{T}}\boldsymbol{x}$, and place a threshold w_0 on $\boldsymbol{y}$ such that $\boldsymbol{y}$ belongs to class C1 if $\boldsymbol{y} \geq -w_0$, and to class C2 otherwise, we obtain a linear classifier. Although projecting high dimensional data onto one dimension will in the general case lead to a considerable loss of information, with the possible result that classes which are well separated in input space become strongly overlapped, it is possible to adjust the components of weight-matrix w such that we can select a projection which maximizes class separation in one dimension. In LDA, this adjustment is done by maximizing the Fisher criteria, defined as the ratio of between-class variance to within-class variance.

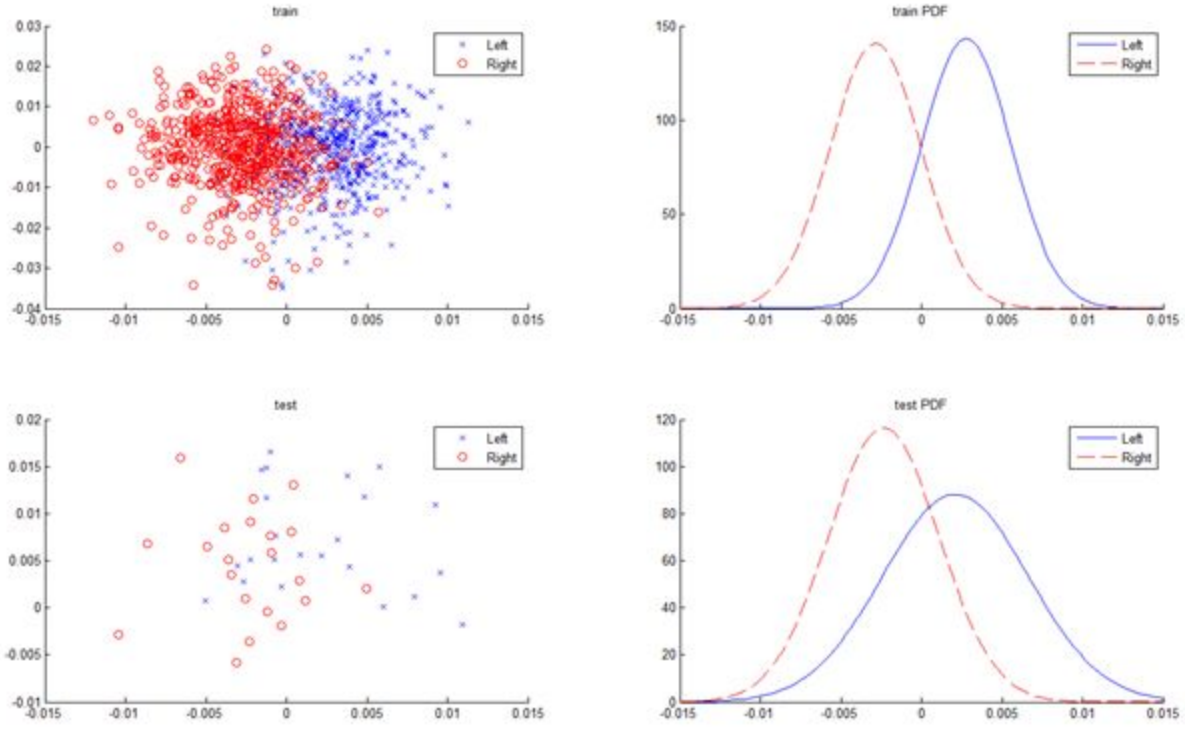


Fig. 4. (Left) Projection of train (top) and test (bottom) data into 2D LDA subspace – (Right) normal distribution of features in one-dimensional subspace

LDA assumes the data in each class is distributed according to homogeneous covariance matrices, but in not all cases does the data come from such ideal distributions. QDA allows for heterogeneous covariance matrices in calculating the class boundary by considering a quadratic

score function. Comparing the superiority of LDA against QDA is dependent on the class distributions and their covariance matrices. Generally, if classes have multivariate normal densities, QDA is the optimal classifier, whereas if classes have multivariate normal densities with equal covariance matrices, LDA is the optimal classifier [Narsky et al. 2014]. Any linear model can be turned into a nonlinear model by applying the so-called "kernel trick" to the model, replacing its predictors with the kernel function. Such nonlinear models usually obtain better classification performance, because of their sensitivity to a nonlinear decision border. In search of a superior classifier, we apply Kernel Discriminant Analysis (KDA) with a Radial Basis Function (RBF) kernel to the task of classifying EEG data.

Fig. 4 illustrates a 2D projection obtained using LDA of left/right hand movement trials, split into 28 subjects in the train set (top) and subject number 29 in the test set (bottom). Of course, a one-dimensional subspace (X-axis) suffices for dividing the data into two classes, however two dimensions are used to facilitate interpretation and make the plot more interesting. The normal distribution of left and right hand along the first subspace (X-axis) are shown in the plots on the right side of the figure. As was to be anticipated, the discrimination accuracy for training data is slightly greater than that of the test data, both sets falling in roughly the same distribution. This is due to the large proportion of the data (28 out of 29 subjects) used for training, allowing for a good estimation of the data distribution in input space.

3. Results

In order to evaluate the selected classifiers as candidates in a 'zero-training' BCI system, we examined their performance on both the described subset of the PhysioNet dataset, and the data we collected from a single subject over multiple sessions using the experimental procedure outlined above. The PhysioNet dataset was used to examine between-subject transferability of the learned decision boundary, whereas our own dataset, due to the inferior quality (relative to the PhysioNet dataset) of the EEG recordings, was used to examine between-session transferability.

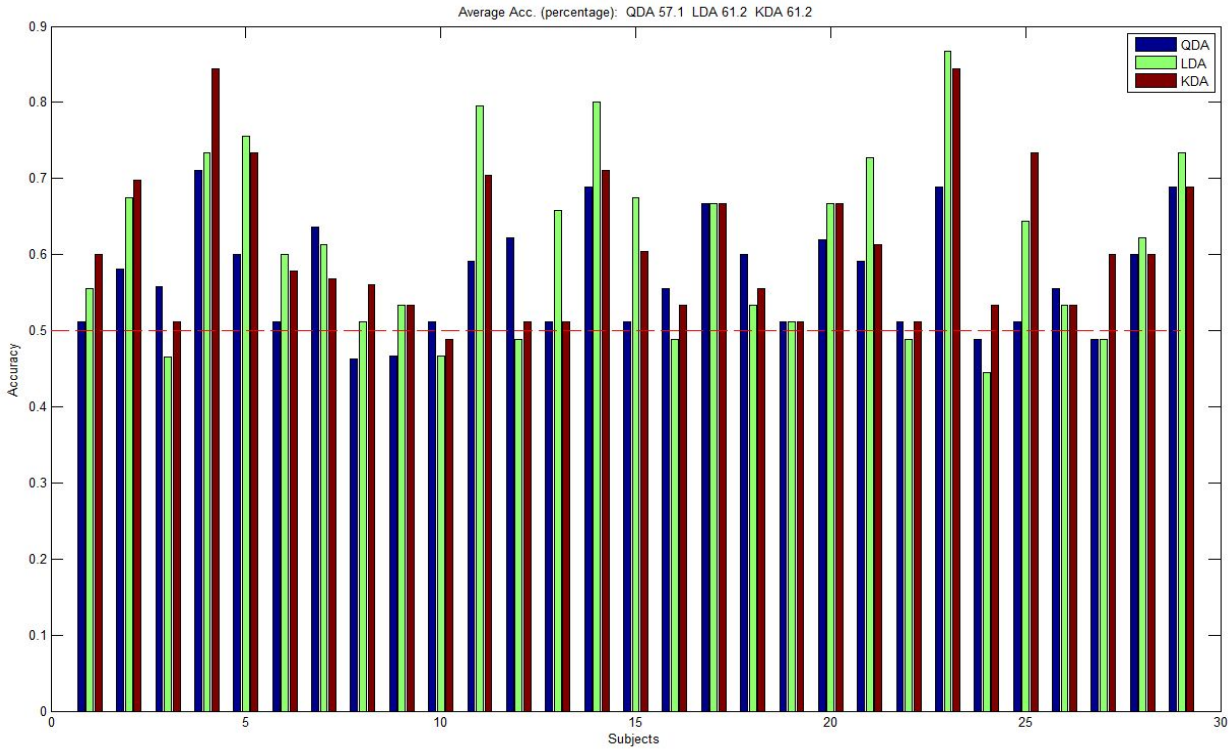


Fig 5. Between-subject classification on PhysioNet data - bars present accuracy of prediction by training on 28 subjects and testing on remaining subject (leave-one-out cross validation) for three classifiers; Mean and standard deviation of accuracies are *QDA* = 57.1∓7%, *LDA* = 61.2∓12% *and KDA* = 61.2∓11%

We used 'leave-one-subject-out' cross-validation in order to assess classifier transfer performance (Fig. 5), and compared this to the performance that would be obtained if training and testing were done within each dataset (Fig. 6). Figure 5 depicts the evaluation of LDA, QDA and KDA by leave-one-out cross-validation for all 29 subjects. In this figure, the X-axis denotes which subject was held out for testing. The performance of KDA and LDA is quite similar, with an average accuracy of 61.2% for both KDA and LDA. The QDA classifier performed poorly in comparison with the others, perhaps indicating homogeneous covariance matrices between classes as outlined above. Within-subject classification accuracy was calculated using 5-fold cross-validation; as can

be seen in Figure 6, classification accuracy did not improve significantly in this condition (KDA: 60.4%, LDA: 65.2%), with QDA even performing slightly worse at 53.7%.

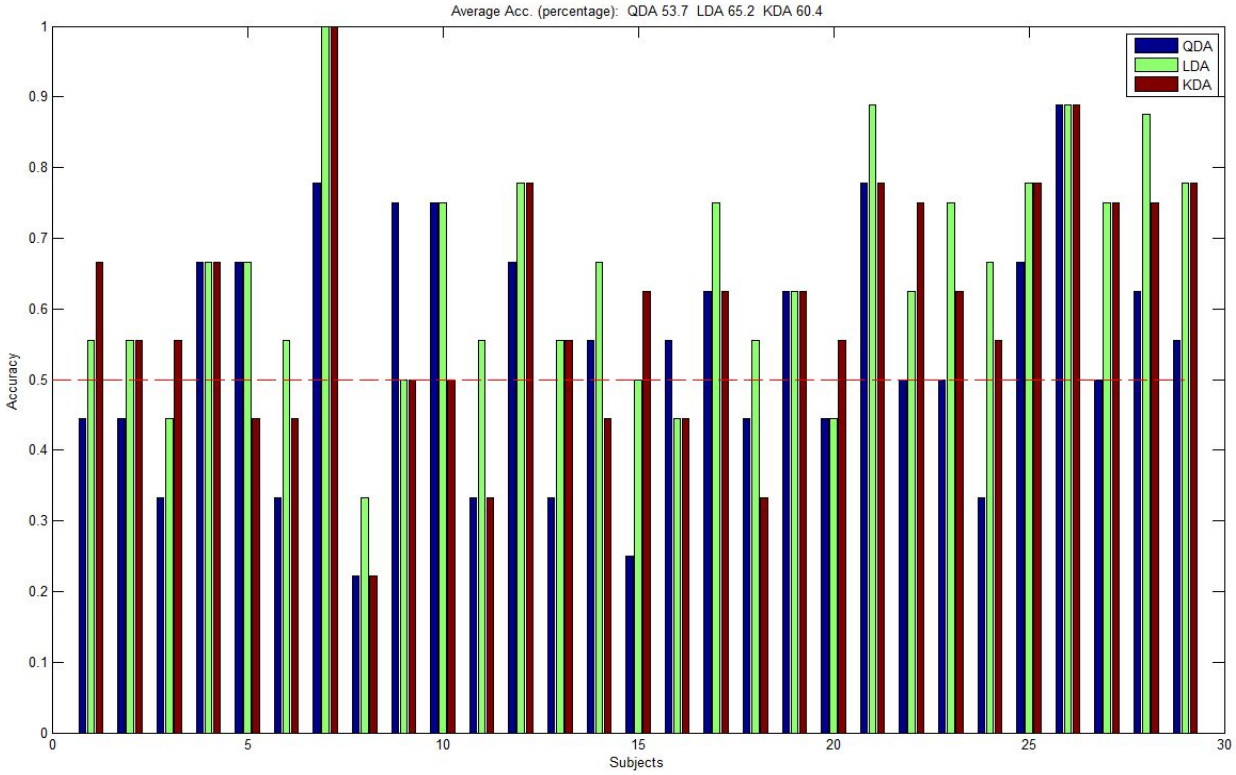


Fig 6. Within-subject classification on PhysioNet data - bars present accuracy of prediction by training on 80% of data per subject and test on rest 20%; Mean accuracies are
QDA = 53.7∓17%, *LDA* = 65.2∓15% *and KDA* = 60.4∓17%

Between-session transfer of the classifiers was evaluated using our own dataset via leave-one-session-out cross-validation (Fig. 7), and again compared with the performance that a traditional approach of re-training the classifier for each session would obtain, via 5-fold cross-validation. On average, between-session transfer performance on the water-based EEG data was comparable to the between-subject transfer performance on the gel-based EEG data; however, performance was significantly better during within-session classification, with LDA achieving an impressive mean classification score of 80%.

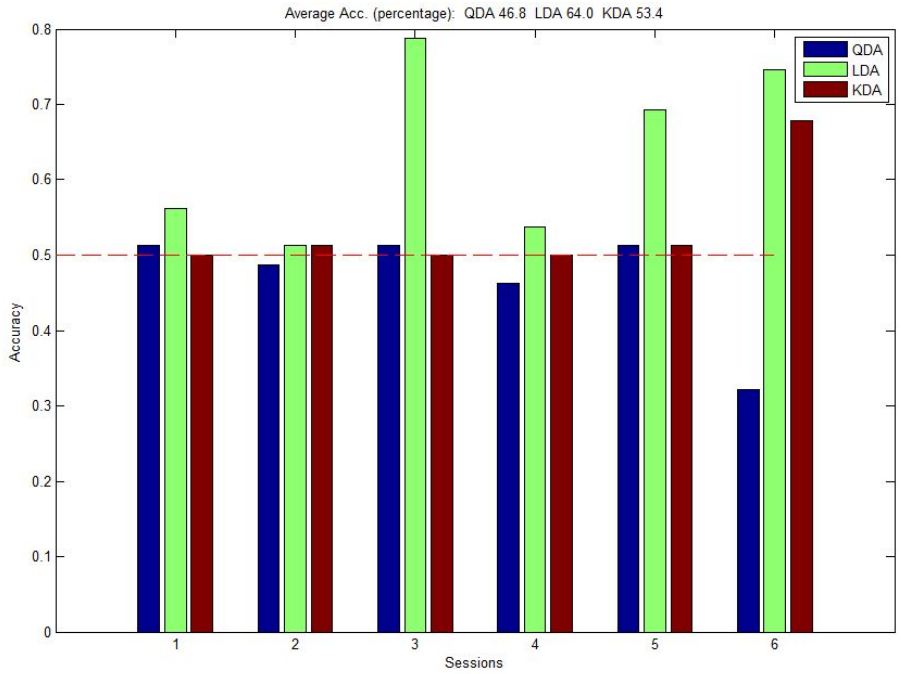


Fig. 7. Between-session transfer performance of classifiers on collected data - mean performance: QDA=46.8∓7%, LDA=64∓11%, KDA=53.4∓7%

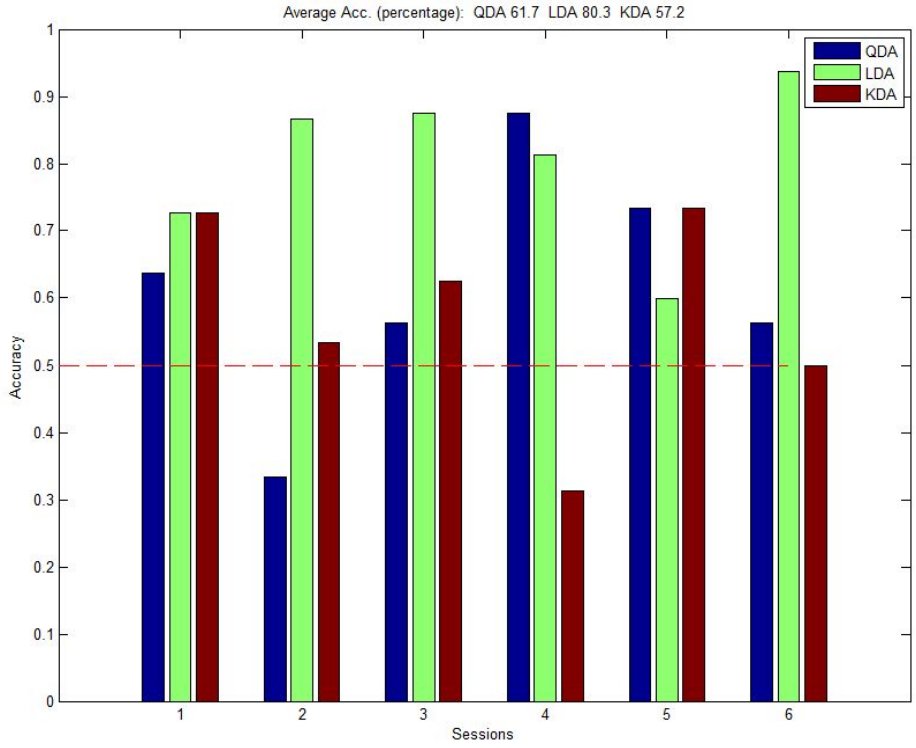


Fig. 8. Within-sessions performance of classifiers on collected data - mean performance: QDA=61.7∓18%, LDA=80.3∓12%, KDA=57.2∓16%

4. Discussion

In agreement with the findings of Lotte et al. [2007], the LDA classifier seems the safest bet, achieving the best mean accuracy across both conditions, on both datasets. Interestingly, the gap between LDA and other classifiers widened when applied to the data we collected; perhaps this is due to the noisier and messier nature of data acquired via a water-based system, such that more sophisticated classifiers are liable to overfit onto some aspect unique to that recording session. Furthermore, in an effort to simulate the rushed and sloppy application of the cap as might be expected in every-day usage, we shifted the cap's position on the scalp slightly between recording sessions; this is in great contrast to the PhysioNet data, where it can be expected that great care was taken to keep electrode position and signal quality as precise as possible.

The results of the between-subject transfer evaluation are extremely encouraging; one classifier (QDA) actually performed better during transfer, and whereas the mean performance of LDA did improve slightly during within-subject classification, so did the variance, suggesting that not much is lost by between-subject transfer; the potential benefit of skipping a training/calibration phase entirely, we feel, outweighs this meager loss of accuracy. The poorer performance in between-session transfer may once again be attributable to the greater variance in cap position, and lower quality of signal recorded using the water-based system.

Our results suggest that zero-training classifiers are better suited to the precise signals only obtainable using a gel-based system. However, variance in between-subject transfer performance is nonetheless quite large. Future work should focus on applying techniques for finding features that do not vary as greatly between subjects, one promising candidate for which might be the recently developed Vanishing Component Analysis [Livni et al. 2013]. Once suitable features have been extracted, these may be combined with more powerful classifiers, such as the currently popular 'deep' neural networks; sequence-sensitive recurrent neural networks could even allow for asynchronous classification of EEG signals. An intermediate approach would be to train a classifier as detailed above, skipping the calibration phase only to covertly update the classifier based on incoming data recorded from the new subject, although a supervised update of the classifier would only be possible in extremely controlled circumstances. A recent unsupervised technique using principles of thermodynamic diffusion to learn a transformation from the input data to a simple, target distribution [Sohl-Dickstein et al. 2015] may overcome this limitation, even removing the need for a feature-extraction step.

5. References

[Sleight et al.] Sleight J., Pillai P., Mohan S., Classification of executed and imagined motor movement EEG signals, 2009.

[Morimoto et al.] Morimoto T., Sketch S., Classifying the Brain's Motor Activity via Deep Learning, 2014.

[Lotte et al.] Lotte, Fabien, et al. "A review of classification algorithms for EEG-based brain–computer interfaces." Journal of neural engineering 4.2 (2007): R1.

[Alomari et al. 2013] Alomari, Mohammad H., Aya Samaha, and Khaled AlKamha. "Automated classification of l/r hand movement eeg signals using advanced feature extraction and machine learning." arXiv preprint arXiv:1312.2877 (2013).

[Schalk & Mellinger, 2010] G. Schalk, J. Mellinger: "A Practical Guide to Brain-Computer Interfacing with BCI2000", Springer, 2010

[Goldberger et al. 2000] Goldberger AL, Amaral LAN, Glass L, Hausdorff JM, Ivanov PCh, Mark RG, Mietus JE, Moody GB, Peng C-K, Stanley HE. PhysioBank, PhysioToolkit, and PhysioNet: Components of a New Research Resource for Complex Physiologic Signals. Circulation 101(23):e215-e220, 2000.

[Baudat et al. 2000] G. Baudat, F. Anouar, "Generalized Discriminant Analysis Using a Kernel Approach", Neural Computation, 12:2385-2404, 2000.

[Narsky et al. 2014] Narsky, Ilya, and Frank C. Porter. "Linear and Quadratic Discriminant Analysis, Logistic Regression, and Partial Least Squares Regression."Statistical Analysis Techniques in Particle Physics: Fits, Density Estimation and Supervised Learning: 221-249, 2014.

[Livni et al. 2013] Roi Livni, David Lehavi, Sagi Schein, Hila Nachlieli, Shai Shalev-Schwartz, Amir Globerson, "Vanishing Component Analysis", Proceedings of the 30th International Conference on Machine Learning, JMLR:W&CP volume 28, 597-605, 2013.

[Sohl-Dickstein et al. 2015] Sohl-Dickstein, J., Weiss, E.A., Maheswaranathan, N., & Ganguli, S., "Deep unsupervised learning using nonequilibrium thermodynamics", Proceedings of the 32nd International Conference on Machine Learning, JMLR:W&CP volume 37, 2015.

[Wolpaw et al. 2002] Jonathan R. Wolpaw, Niels Birbaumer, Dennis J. McFarland, Gert Pfurtscheller, Theresa M. Vaughan, Clinical Neurophysiology 113 767-791, 2002

[Sugiyama et al. 2007] Sugiyama, Masashi, Matthias Krauledat, and Klaus-Robert Müller. "Covariate Shift Adaptation by Importance Weighted Cross Validation." The Journal of Machine Learning Research 8 (2007): 985–1005.

[Llera et al. 2012] Llera, A., V. Gómez, and H. J. Kappen. "Adaptive Classification on Brain-Computer Interfaces Using Reinforcement Signals." Neural Computation 24, no. 11 (July 30, 2012): 2900–2923.

6. Supplementary Material

User's and Programmer's Guide

The entire experiment consists of 4 stages: the preparatory phase, the train phase, the calibration phase and the test phase.

- The preparatory phase sets up the required buffer-BCI architecture and includes capfitting.
- The train-phase is an online data-acquisition phase where the user is prompted to do several tasks.
- The calibration-phase is an off-line data-analysis phase where the data and events gathered from the train phase is pre-processed and is used to train the LDA classifier to
- The test-phase is the online phase that uses a synchronous data-acquisition session and classification session to classify on-line data gathered from the user, and sends feedback back to the user.

Below, each of these phases and their corresponding scripts will be explained in detail. The variables discussed can be changed if wanted, but can easily be left on their defaults.

Preparing the experiment

To prepare for the experiment, make sure the Mobita is charged, properly inserted into the cap electrodes array, the Mobita's wifi dongle is inserted into the analysis pc, and a connection is made between the dongle and the Mobita.

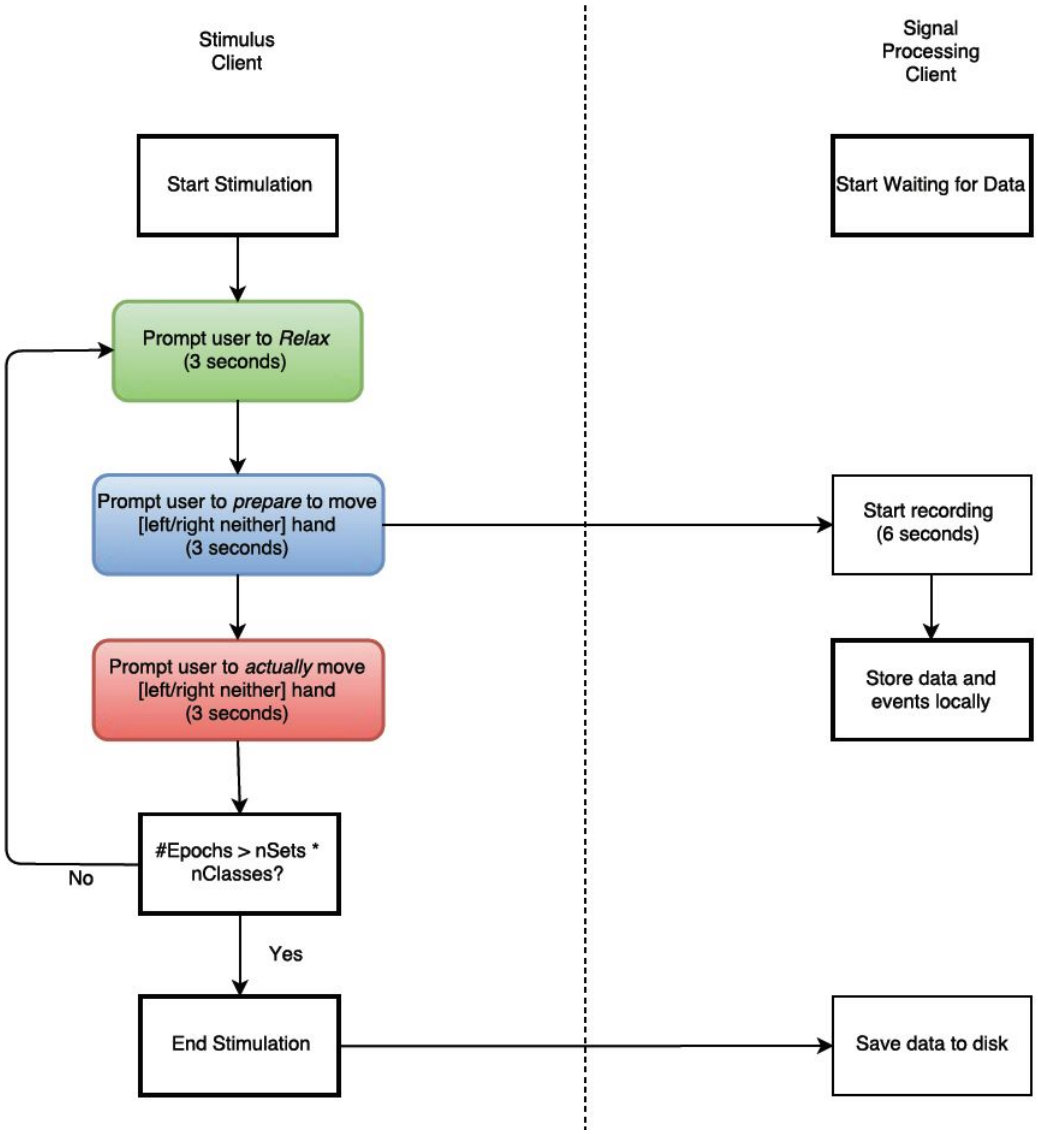
Now, edit the **startEverythingMobita.bat** file and change the SET Path="..\buffer_bci\dataAcq" line to point to the location on-disk where the buffer_bci's dataAcq folder is located. Save it, and run the batch file.

This batch file will now open the JavaBuffer through which all the events and samples will be put (and saved on disk); it will start the Mobita signal acquisition, which will catch the signals sent by the Mobita and run it through the buffer; it will start an event viewer, which will show on-screen the events sent through the buffer by the scripts you will be running; and finally it will start the signal viewer which will open a Matlab session and prompt you to choose a cap file (when using the default Mobita 32 channel cap, choose cap_tmsi_mobita_32ch.txt in the buffer_bci's utilities/caps folder).

Now, proceed with capfitting: if not done yet, fit the Mobita's electrodes into the cap as specified by the capfile you see in the signal viewer. Wet and insert the sponges, and put the cap on the participant. In the signal viewer, set the low-band and high-band filter to 40 and 1 (to filter out environmental 50Hz noise from power lines and some misc. noise). Now navigate to the 'Power' section of the signal viewer's drop down menu and identify which electrodes are not in the green area of the color spectrum as seen on the right. Wiggle the electrode, push it through the hairs

against the skin as much as possible, even take out the sponge and re-wet them if necessary until all the electrodes are at least out of the red. This, because we will take a common-average re-referencing approach and bad electrodes can muddy this demeaning procedure.
Now, open up two more Matlab sessions. One will be used for the signal processing, and the other for stimulus presentation.
Before doing anything else, edit the **EasyInitForBCIBuffer** script and point the path variable in the line path = "..\buffer_bci" to wherever the main folder of the buffer_bci is located on-disk. This script will be called in almost every other script we will use further, so it won't be necessary to edit this in any other script individually now we have done this.

The Train-phase of the experiment



Now everything is set up, in one of the Matlab sessions, open up the **ClassifierDataAcqBlock_Signals** script. This script will catch the EEG signal samples from the buffer every time a move/no-move prompt is given to the user, and save it to a file when the stimulation-end event is put in the buffer. In the section surrounded by 'FILL IN CONSTANTS ABOVE', if you wish, you can change the following variables:

- dname: whatever name you wish to give to the file that will hold all the caught data and events. For ease of loading data into the preprocessing-and-calibration scripts later, keep the name in the following format: training_dataX, where X is the suffic that is to be changed (for example: 'training_data9').
- trialDuration: the amount of seconds of data you wish to save after every prompt (or leave it to the default of 6 seconds). Note that for easy-of-use, you could just catch the total trial time (i.e. the full seconds it takes the user to go through the prepare-to-move and move-phases during stimulation, which is by default 3 and 3 seconds, which is 6 seconds in total).

Then start the script.

In the other Matlab session, open up **ClassifierDataAcqBlock_Stimulus**. This will present stimuli to the user (in this case: prompt the user to move the left hand, the right hand, or neither hand). Experimental flow consists of a rest phase (in which the user is prompted to relax), then a prepare-to-move phase (in which the user is prompted to *prepare* to move their left/right/neither hand(s)), and a move-phase (during which the user is prompted to *actually* move their left/right/neither hand(s)). This repeats in a random, but balanced stimulation sequence for as long as is specified in the nSets and nSymbs variables (where the amount of experimental trials is nSets * nSymbs).

- If you wish, you can change the following variables:
- nSets: how many trials you'd like to see per class. i.e. when set to the default of 20, every class will be included in the random stimulation sequence 20 times.
- nSymbs: how many classes there are. Note that this is set to 4 as a default (rather than 3). See the variable 't' below for the reason for this.
- t: the texts associated with the classes. Note how for the default texts, the last two are both 'NEITHER hand'. This is because we want an equal amount of move/no-move trials. Because there are are 2 classes that are movement (left hand/right hand), we also need 2 classes that are non-movement (hence the double inclusion of 'neither hand'). Note that if you wish the experiment to only distinguish movement-vs-nonmovement classes, and you wish to change this script to feature purely movement-nonmovement (rather than left hand / right hand / no-movement) type of stimulation prompts, it is advised to, instead of reducing the classes, edit this variable to ['BOTH hands','BOTH hands','NEITHER hand','NEITHER hand'], as later preprocessing during the calibration phase relies on

there being 4 classes (the latter two of which being no movement, and the first two being movement).

- moveDuration: how long the 'move' phase of stimulation should be in seconds. (default: 3 seconds)
- prepDuration: how long the 'prepare-to-move' phase of stimulation should be in seconds. (default: 3 seconds)
- interTrialDuration: how long the 'relax' period between trials should be in seconds. (default: 3 seconds)

Now start the script. It will start a random sequence of 'prepare-to-move' and 'move' prompts, and as a default, will prompt the participant to move the left, right or neither hand, with a resting phase in between. At the start of each prepare-to-move phase, it will send an event which will prompt the ClassifierDataAcqBlock_Signals script above to catch data from the buffer and save it. Because of this, if you wish to catch the signal samples from the full task, it is important to set the trialDuration in said ClassifierDataAcqBlock_Signals script to the sum of moveDuration and prepDuration in this script. When the stimulation is finished, it will thank the user, and send the stimulus.end event, which will prompt the ClassifierDataAcqBlock_Signals to save the data to a file, after which we can use it for training the classifier in the next phase of the experiment.

The Calibration-phase of the experiment

In one of the Matlab sessions, open up the **ClassifierTrainBlock** script. In here, the LDA classifier is trained with the preprocessed training data that we have gathered. In this script, you can change the following variables:

- subjects: the names of the suffixes you assigned after 'training_data' (e.g. ['1','2','3','Pete1','Pete2']
- elects: the *indices* of the electrodes you wish to take into account during training the LDA.
- start: the start index of the sample you wish the LDA to take into account during training. As a default, we look at the 601st sample for a start (i.e. roughly one second before movement onset)
- dur: the duration (in amount of samples) you wish to examine after the start sample for training the LDA. Default is 300 (roughly 1 second).
- cname: the name it will save the classifier as after it has been trained.
- discrType: is set to its default value of 'linear', which seeks to linearly discriminate between the classes, but can also be set to 'quadratic', 'diagLinear', 'diagQuadratic', 'pseudoLinear' and 'pseudoQuadratic'.
- gamma: is set to default of 0.7. It is a parameter for regularizing the correlation matrix of predictors, and needs to be in the range of 0 to 1 if a linear discriminant type is chosen, or either 0 or 1 if quadratic discriminant type has been chosen.

- prior: should be set to ‘uniform’ if the class labels are uniformly distributed (as in our experiment, where movement / no-movement trials are balanced), else should be set

Then start the script. It will run and create a classifier, which is will save to the specified filename.

- fs: the sampling frequency the data was gathered at. Is set to a default of 250, but your own setup may be different.

Note that the pre-processing script is called in the line:

- [dat, lab, ~, remep, ~,fid] = Preprocess2FingersTimeFrequency(dat,lab,[1:64],fs)

Where the parameters are as follows:

- [data, labels, channel_labels, removed_epochs, frequencies, frequency_ids] = Preprocess2FingersTimeFrequency(data,labels,channel_labels,sampling_frequency);

The most important argument for this that you should change, when necessary is:

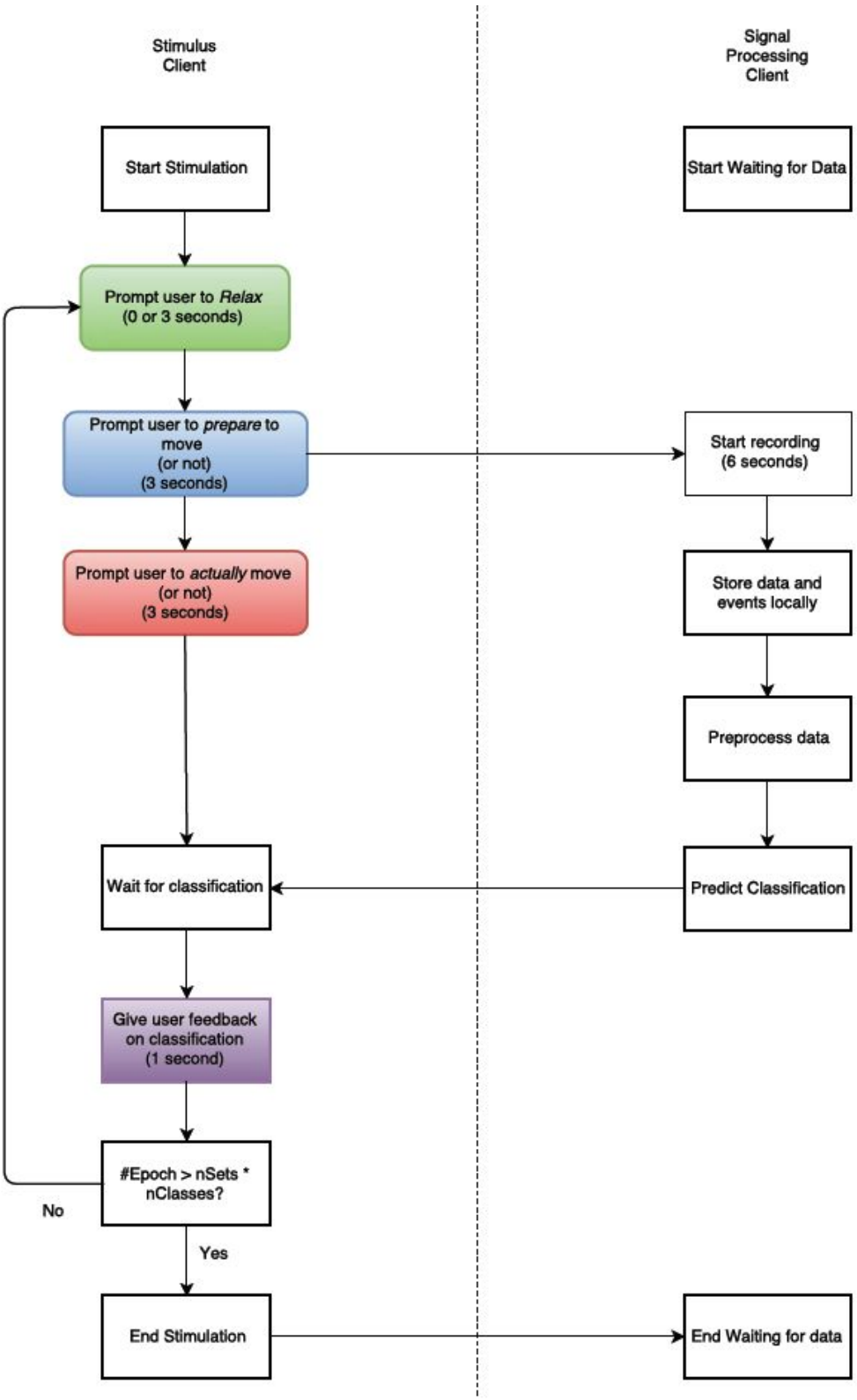
- the sampling frequency, which in the example above is 250, but may be different for your own set-up.

Finally, note that the classifier is actually trained in the lines:

- class = fitcdiscr(data, labels, 'discrimtype',discrType, 'Gamma', gamma);
- class.Prior= prior;

Note that the classifier is built into Matlab’s Statistics and Machine Learning Toolbox, and more info can be found about this by typing: help fitcdiscr or doc fitcdisc into the Matlab’s command line interface if said toolbox has been installed on your Matlab instance, or by going to Mathworks’ website at: http://nl.mathworks.com/help/stats/fitcdiscr.html .

The Test-phase of the experiment



Now, in one of the Matlab sessions, open up the **ClassifierTest_Signals** script. This script will listen for events that specify a stimulation event has started, and will catch data starting from there for an amount of time. It will then preprocess and classify the signals using the specified classifier, and send out an event saying which class it predicted the signals to be. The constants that can be changed are:

- trialDuration: how many seconds one trial takes (again, the prep phase plus the move phase). It is important these are exactly the same as the ones from the DataAcquisition/Train blocks. Default is 6 seconds.

Before starting the script, load the classifier into Matlab by double clicking the file in the file navigator. Now we can start the script.
In the other Matlab session, open one of the following scripts:

- **ClassifierTestBlock_Stimulus**
- **ClassifierTestBlock_Stimulus_NoTask**
- **ClassifierTestBlock_Stimulus_Pong**

Each one has the same type of constants, which are:

- nSeq: how many trials per class we wish to prompt the user with.
- nSymbs: how many classes. Note how these have been reduced from left/right/neither hand, to the 2-classes of ‘no movement’ vs ‘movement’.
- t: the texts associated with the classes. (default: ‘NO movement’ and ‘MOVEMENT’)
- prepDuration: how long the ‘prepare to move’ phase should be in seconds. (default is 3, which is the same as we used in the training phase).
- moveDuration: how long the ‘movement’ phase should be in seconds. (default is 3, which is the same as we used in the training phase).
- interTrialDuration: how long the ‘relax’ period before stimulation should be in seconds (default: 3, except in the pong script where intertrial periods are removed so the game can be played without rest periods, i.e. interTrialDuration is set to 0)
- beforeTrialDuration: how long the amount of time before stimulation begins should be in seconds. (default: 3 seconds)
- predictionDuration: how long the text feedback on classifier’s prediction should be shown in seconds. (default: 1 second)

For the actual pong game, in the **ClassifierTestBlock_Stimulus_Pong** script, there are a number of additional options, namely:

- sleepBeats: whether or not to sleep (i.e. pause the pong game) during the ‘prepare to move’ and ‘move’ phases, or whether to continue the game. (default: false, i.e. it will NOT sleep/pause, and simply continue the game, even during the prepare to move and move phases).

- ballSpeed: the ratio of the screen the ball moves every frame. (default: 0.0005)
- paddleSpeed: the ratio of the screen the paddle moves with every frame. (default: 0.0005)

You can now start any of these scripts to test the classifier. The scripts themselves differ in that they test the classifier in different ways:

- **ClassfierTestBlock_Stimulus**: prompts the user to move or not move via a pre-generated stimulus sequence, displays the prediction result from the classifier, and computes the performance of the classifier as measured by amount of correct/incorrect predictions when comparing the classifier's predictions with the actually prompted classes.
- **ClassifierTestBlock_Stimulus_NoTask**: simply prompts the user to move or not move, and displays the result. There is no set stimulation sequence, so the user can freely test by deciding for him/herself to move or not move (rather than being prompted by the stimulation itself).
- **ClassifierTestBlock_Stimulus_Pong**: does the same as the ClassifierTestBlock_Stimulus_NoTask script, only wrapped within a game of single-player pong where the paddle will change direction if a movement prediction is received from the classifier.