33rd Annual International Conference of the IEEE EMBS
Boston, Massachusetts USA, August 30 - September 3, 2011

Programming an Offline-Analyzer of Motor Imagery Signals via Python Language

Luz María Alonso-Valerdi[a], Francisco Sepulveda[a]
[a]School of Computer Science and Electronic Engineering, University of Essex, Colchester–Essex, UK

Abstract—**Brain Computer Interface (BCI) systems control the user's environment via his/her brain signals. Brain signals related to motor imagery (MI) have become a widespread method employed by the BCI community. Despite the large number of references describing the MI signal treatment, there is not enough information related to the available programming languages that could be suitable to develop a specific-purpose MI-based BCI. The present paper describes the development of an offline-analysis system based on MI-EEG signals via open-source programming languages, and the assessment of the system using electrical activity recorded from three subjects. The analyzer recognized at least 63% of the MI signals corresponding to three classes. The results of the offline analysis showed a promising performance considering that the subjects have never undergone MI trainings.**

I. Introduction

A Brain Computer Interface (BCI) is a system which attempts to connect a user with his/her environment and devices in it. The human-machine interaction of BCIs is accomplished by means of brain signals following one of two broad schemes: to evoke voluntary mental states (e.g., sensory-motor rhythms and slow cortical potentials), or to elicit a specific response in the user's brain via stimulation (e.g., P300 and visual evoked potentials) [1]. In particular, sensory-motor rhythms are electroencephalographic (EEG) oscillations occurring in α (8–12 Hz) and β (12–30 Hz) bands over the primary sensory-motor area of the cortex. These waves are enhanced by awake-restful states, attention-related demands (e.g., attentive expectation of relevant stimulus omission, working memory activation, and episodic short-term memory task), and cognitive-mnemonic processes [2], whereas, they are blocked or attenuated in episodes of: sensory processing, actual motor performance, motor imagery (MI) performance or visualization of movements [3]–[4]. The waves not only change according to the type of movement, but they are also differently modulated by each limb, especially with regard to the location of the most relevant information over the cortex. Furthermore, Green *et al.* (1999) showed that the sensory-motor rhythms are similarly modulated by amputees attempting to move the absent part of their body, and by healthy people performing imaginary movements [5].

Modulation of sensory-motor rhythms via MI has become a popular method in BCI research. Applying this method usually takes place in six phases. These are:

1. *Training session without performance feedback.* Consecutive and random imaginary movements without providing performance measurement.
2. *Offline analysis.* A Process that involves three main steps: (i) brain signal conditioning, (ii) MI-feature extraction, and (iii) parameter tuning of an adaptive model via samples of MI patterns (training dataset) in order to classify which movements is being executed (testing dataset).
3. *Optimization of the classifier.* Search of the most suitable frequency bands, based on event-related (de-) synchronization (ERD/ERS) studies, and search of the most convenient electrode positions.
4. *Training session with feedback*. Training session that makes use of the previously adapted system for providing performance measurement.
5. *Classifier update.* Retraining of the current classifier via the most recent recorded brain signals.
6. *Application of the BCI system*. Online analysis of the brain signals for translating the user's desires into control commands of a device of interest.

Despite the large number of references describing the aforementioned method (e.g., [6]-[12]), there is not enough information related to the available programming languages that could be suitable to prototype a MI-based BCI system. Guger *et al.* [13] proposed an EEG-based BCI by using Matlab, Simulink, and Real-Time Workshop (MathWorks, Inc.). Programming languages supplied by MathWorks are high performance languages conducted for mathematical, engineering, and science areas. The foremost drawback of this software is the cost. Although it is feasibly afforded by business environments, it may become a financial burden for the private sector. For instance, worldwide universities often purchase a limited number of licenses to deal with the cost. However, they seldom satisfy the demand. MathWorks usually suggests paying for a low-cost student edition in restricted budget cases, which is not often eligible.

On the other hand, there is an open source emulator of Matlab (Octave, [14]), unfortunately not all Matlab features and toolboxes work with Octave.

With the above considerations in mind, the goals of the present study are: (1) to develop an offline-analysis system of MI signals via a proficient open-source programming language; and (2) to assess the system using MI signals recorded from three subjects. Python was considered the ideal tool for the study-proposes because it is multi-platforms, it has clean syntax, it has a large number of standard libraries and modules, it has bindings to all standards graphical user interface (GUI) toolkits, and it engages additional modules for specific tasks such as digital signal processing (DSP), and machine learning (ML) [15]–[16].

II. Design of an Offline Analyzer of MI-EEG Signals via Python Programming Language

The offline analysis software is entirely written in Python[1], and it is essentially supported by Numpy[2] and Scipy[3] (scientific-computing libraries) in addition to matplolib[3] (plotting library). The GUI of the offline analyzer is programmed on PyGTK[4], and the ML process is carried out by means of the module LIBSVM[5]. The main functions of the software are hereunder summarized.

A. Data Upload

Data must be organized in three dimensions (channels, trials, and samples), and are uploaded as mat-file (binary form to store numerical arrays). No Matlab is needed.

B. EEG Signal Characterization

In order to process the electrical brain activity, and to identify a variety of brain states, the EEG signal must be analyzed according to its stochastic nature, and its response to visual/auditory stimuli or to mental tasks.

Firstly, stochastic processes require probabilistic structures that could accurately characterize their behaviour. The well-known central limit theorem is commonly used, assuming that an EEG signal is the addition of signals generated by independent neural oscillators. This assumption is reliable only if the EEG measurement interval has an adequate length. As a general rule, this EEG interval must not exceed one second length retaining its normal distribution, and thus satisfying the theorem conditions [17]. In order to adequately characterize the EEG data, the option '*Segmentation*' on the menu '*Data Acquisition*' (Fig. 1) fragments each signal according to the inserted time value.

Secondly, depending on the type of stimuli or mental task and the cortical area of interest, a specific response can be distinguished from the EEG recordings. The sensory-motor cortex neurons begin to code the forthcoming imaginary movement about 100ms after the cue onset. The electrical response can be sensed from the scalp between 250ms and 500ms after the target visualization [7]. Considering these facts, the offline analyzer has the alternative to select the time window of interest in each dataset. The option '*Samples*' on the menu '*Data Acquisition'* allows to set the initial and the final sample of the time series of interest.

Last but not least, classical methods of signal processing that accentuate significant features assume a time-invariant behaviour. However, this assumption is not suitable for long EEG recordings. The intrinsic properties of the EEG signal in study may be helpful to retrieve the normal spontaneous activity from the EEG [17]. In our case, the desired signal has slow time variant properties. Therefore, a method for stationary analysis can be repeatedly applied on consecutive and overlapping intervals of the signal. This overlapping requirement can be enabled via the option '*Overlapping*' on the menu '*Feature Extractor and Classifier*' (Fig. 2).

[1] http://www.python.org/
[2] http://numpy.scipy.org/, http://www.scipy.org/
[3] http://matplotlib.sourceforge.net/
[4] http://www.pygtk.org/
[5] http://www.csie.ntu.edu.tw/~cjlin/libsvm/

C. Signal Conditioning

EEG signal conditioning methods are used for eliminating the noise artefact effects, and to emphasize relevant signal features. For these reasons, specific tools that obtain more localized patterns (i.e., spatial filtering), and that determine the frequency content of the waveform (i.e., spectral filtering) are available in the menu '*Signal Conditioning*' (Fig. 2). The offline analyzer additionally plots the spectral behaviour of the conditioned EEG signals by using the option '*Spectrogram*' on the menu '*2D Plots*' (Fig. 4).

Spatial filtering methods are especially important for the treatment of MI signals. The electrical activity picked up from the scalp is detected by very distant (up to a few cm) sensors. As a result, the potentials caused by the neurons of interest spread to surrounding electrodes due to the volume conduction effects (e.g., through brain tissue, cerebrospinal fluid, surrounding tissues, skull and scalp). Weak signals such as the sensory-motor rhythms are also affected by α rhythms related to the visual system. It is thus required that we use a high-pass spatial filter that emphasizes focal activity and that reduces more diffuse components, hence improving spatial resolution. Common Average and Laplacian Methods (available options on the menu '*Signal Conditioning*') are frequently used to emphasize focal activity, and to reduce more diffuse components. The former diminishes the contribution of components present in the entire electrode montage, whereas the latter decreases the effect of common components adjacent to the channel of interest [18]-[19].

D. Feature Extraction via BP Estimates

Some of the most common methods for extracting frequency specific EEG features are based on auto-regressive models and signal power analysis. Particularly, band power (BP) estimates have played an important role into control MI-based BCIs. BP parameters improve the classification accuracy by selecting the appropriate electrode positions (options '+ *Channels*' and '- *Channels*' on the menu '*Signal Conditioning*'), and by monitoring shift in α and β bands according to the user's intentions. Besides the θ and γ bands could also carry MI-related information [4], [7]. To take this into account, the software has the option to select the frequency bands and the corresponding bandwidths by the option '*Band Power Estimates*' on the menu '*Feature Extractor and Classifier*' (Fig. 3).

E. Classification via SVMs

A support vector machine (SVM) maps the input data to a higher dimensional space, where is more likely to identify different classes, via a kernel function. Considering x the training dataset and y the desired output, the kernel for this study is described by $K(x, y) = exp(-\gamma \|x - y\|^2)$, which is known as radial basis function. The procedure to adapt a SVM is focused on four steps: (i) scaling of datasets within the [-1 1] range, (ii) searching for the C and γ parameters by means of cross-validation, (iii) SVM model creation via the training dataset, and the most suitable C and γ parameters, and (iv) assessment of the current model with the testing dataset.

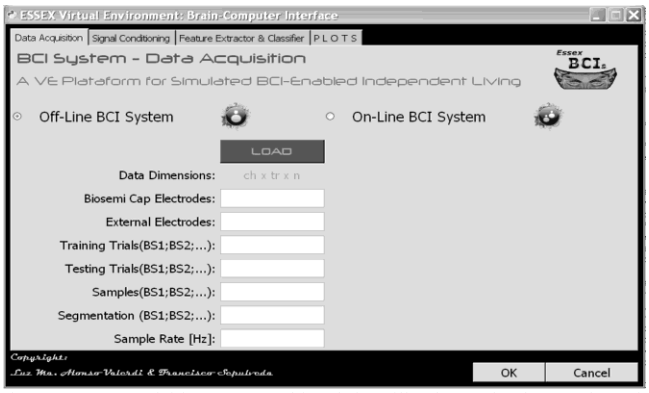

Fig. 1. Data Acquisition Menu. This tab is utilized to upload EEG-data via the button LOAD, and to characterize the EEG signals via the text-entries.

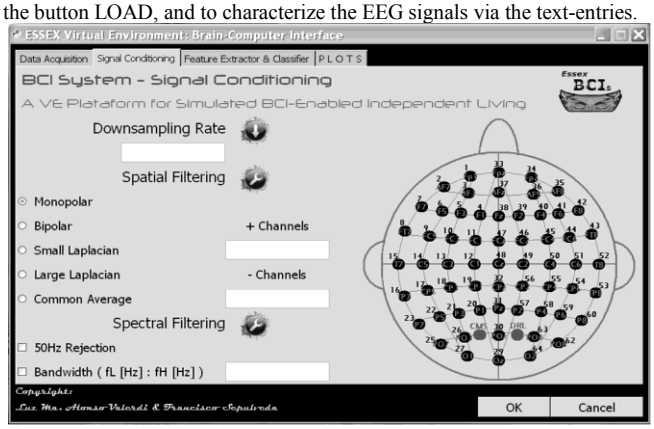

Fig. 2. Signal Conditioning Menu. This interface is useful to down-sample and to spatially/spectrally filter the EEG signals based on the layout above.

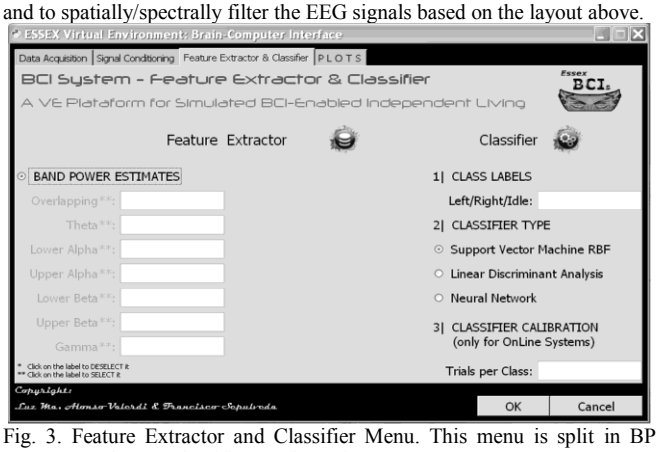

Fig. 3. Feature Extractor and Classifier Menu. This menu is split in BP extractor and SVM classifier configuration.

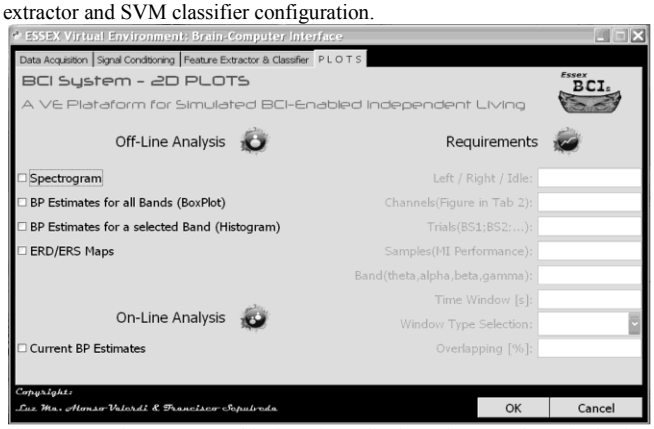

Fig. 4. 2D-Plots Menu. This interface displays four modes of plotting: spectro-temporal analysis (spectrogram), feature distribution according to frequency bands (boxplot and histogram), and ERD/ERS maps.

III. Experimental Procedure

Three untrained subjects (S1, S2, and S3), one female and two males, ages 26-31 participated in a 50-minute session. They were right-handed students and did not report neurological abnormalities. The session was arranged in a 30-minute phase for mounting 64 EEG channels, and a 20-minute phase for MI training purposes. All procedures were approved by the University of Essex's Ethics Committee.

A. Data Recording

The acquisition of the EEG activity was done with a Biosemi (TM) ActiveTwo system along with ActiView computer software. EEG activity was recorded via 64 active electrodes, plus a Driven-Leg-Right electrode and a Common-Mode-Sense electrode (Fig. 2). The ActiveTwo system was configured to acquire signals within a 400Hz bandwidth; and it sends the data to a sampling at 2048Hz. The ActiView software was programmed to decimate to 512Hz (configuration that limits the bandwidth to 104Hz), and to store the EEG recordings. The data are saved in BDF-format (i.e., 24-bit version of the popular 16-bit EDF format). The BDF-files are translated to MAT-files by using an open-source library BioSig[6], which provides a Python version of the BDF converter.

B. Training Protocol programmed on PyGTK

The training protocol is organized in four runs. Each run has 40 trials divided equally in two classes (left/right); that is, 80 trials per class per session. Each trial lasts between 10 and 11 seconds, and it has six epochs: (i) warning sign (upright person) from 0s to 2s; (ii) cue (person pointing towards left/right) onset plus a beep from 2s to 3s; (iii) left/right hand imaginary motion from 3s to 7s; (iv) idle state extraction from 8s to 10s, which is useful to identify non MI data; and (v) random intertrial interval from 10s to 11s.

C. Offline Analysis

For illustration purposes, MI and idle patterns were assembled using channels C_3, C_z, and C_4 of the international 10/10 system (Fig. 2) and processed offline as follows: Firstly, each trial for every mental state was fragmented into seven segments considering 1000ms-interval for left/right MI executions, 500ms-interval for idle state, and 50% overlapping. Secondly, each segmented waveform was down-sampled to 256 Hz and spatially filtered by small Laplacian or CAR methods. Thirdly, each EEG segment was filtered within α_{lower}, α_{upper}, β_{lower}, and β_{upper} bands, squared sample by sample, and averaged over time, giving a value later transferred to logarithmic scale. In summary, each feature set trial comprised three EEG channels; each channel was characterized by four frequency bands; and each band comprised seven intervals (i.e., 84 features per trial). Having assembled the patterns, 40 trials per class were taken for modelling the SVM classifier, and 40 trials per class were taken for assessing the classifier performance. The idle class contributed with 80 trials per phase (i.e., training/testing).

[6] http://biosig.sourceforge.net/index.html

IV. Results

The offline analyzer recognized at least 63% of the MI patterns corresponding to three classes of three subjects (Table 1). The results of the offline analyzer showed a promising performance considering that the subjects have never undergone MI trainings. Moreover, the results of the analysis selecting small Laplacian are slightly higher than the results of the analysis using CAR, which is congruent with the outcomes reported by H. Ramoser, J. Müller-Gerking, and G. Pfurtscheller [19].

TABLE 1
CONFIGURATION AND ASSESSMENT OF THE OFFLINE ANALYZER

System Configuration			
Frequency Bands [Hz]		α_{lower}:8-10,α_{upper}:10-12,β_{lower}:16-20,β_{upper}:20-24	
Spatial Filtering		Small Laplacian	Common Average
System Results			
S1	**Training**	81.3%	71.9%
	Testing	78.1%	63.1%
S2	**Training**	77.5%	78.8%
	Testing	65.6%	65.0%
S3	**Training**	87.5%	83.1%
	Testing	76.1%	73.6%

V. Discussion

The use of Python programming language along with ML plug-ins for developing MI training sessions, and offline analysis systems is relative straightforward. The proposed offline MI-based BCI system successfully discriminated up to 87.5% of MI-patterns of three untrained subjects, which is enough to control an online system. It is important to point out that the present results are limited to two system-configurations; however, the parameters can be tuned in order to find out a more satisfactory human-machine adaptation. Furthermore, the offline analyzer has been implemented with some tools (such as ERD/ERS maps) to figure out the most suitable BCI parameters for the current user. Moreover, the Python community provides extensive variety of modules to design more complex systems. For example, it is strongly recommended to make use of Python(x,y)[7] that is interactive scientific software embedding worthwhile sources; as well as, it could be helpful to harness ML libraries such as Elephant[8] (multi-purpose library for ML), Shogun[9] (comprehensive ML toolbox), Orange[10] (General-purpose data mining), PyML[11] (ML in Python), or MDP[12] (modular data processing.

Acknowledgment

This project would not have been possible unless the sponsor of the National Council of Science and Technology of Mexico, which grants a scholarship for Alonso-Valerdi.

[7] http://www.pythonxy.com/
[8] http://elefant.developer.nicta.com.au
[9] http://www.shogun-toolbox.org
[10] http://www.ailab.si/orange
[11] http://pyml.sourceforge.net
[12] http://mdp-toolkit.sourceforge.net

References

[1] J. R. Wolpaw, N. Birbaumer, D. J. McFarland, G. Pfurtscheller, and T.M. Vaughan, "Brain-computer interfaces for communication and control", *Clin. Neurophysiol.*, vol. 113, no. 6, Jun. 2002, pp. 767–791.

[2] J. A. Pineda, "The functional significance of mu-rhythms: translating seeing, and hearing into doing", *Brain Res. Rev.*, vol. 50, 2005, pp. 57–68.

[3] C. Neuper, R. Scherer, S. Wiressnegger, and G. Pfurtscheller, "Motor imagery and action observation: modulation of sensorimotor brain rhythms during mental control of a brain-computer interface", *Clin. Neurophysiol.*, vol. 120, no. 2, Feb. 2009, pp. 239–247.

[4] G. Pfurtscheller, and F. H. Lopes da Silva, "Event-related EEG/EMG synchronization and desynchronization: basic principles", *Clin. Neurophysiol.*, vol. 110, no. 11, Nov. 1999, pp. 1842–1857.

[5] R. Krepki, G. Curio, B. Blankertz, K. R. Müller, "Berlin brain computer interface – The HCI communication channel for discovery", *Int. J. Hum.-Comput. St.*, vol. 65, 2007, 460-477.

[6] R. Boostani, B. Graimann, M. H. Moradi, and G. Pfurtscheller, "A comparison approach toward finding the best feature and classifier in cue-based BCI", *Med. Biol. Eng. Comput.*, vol. 45, no. 4, Feb. 2007, pp. 403–412.

[7] G. Pfurtscheller, and C. Neuper, "Motor imagery and direct brain-computer communication", *P. IEEE*, vol. 89, no.7, Jul. 2001, pp.1123–1134.

[8] G. Pfurtscheller, C. Neuper, C. Guger, W. Harkam, H. Ramoser, A. Schlögl, B. Obermaier, and M. Pregenzer, "Current trends in Graz brain-computer interface (BCI) research", *IEEE T. Rehabil. Eng.*, vol. 8, no.2, Jun. 2000, pp. 216–219.

[9] G. Pfurtscheller, C. Neuper, G. R. Müller, B. Obermaier, G. Krausz, A. Schlögl, R. Scherer, B. Graimann, C. Keinrath, D. Skliris, M. Wörtz, G. Supp, and C. Schrank, "Graz-BCI: state of the art and clinical applications", *IEEE Trans. Neural Syst. Rehabil. Eng.*, vol. 11, no. 2, Jun. 2003, pp. 1–4.

[10] J. R. Wolpaw, D. J. McFarland, and T. M. Vaughan, "Brain-computer interface research at the Wadsworth center", *IEEE T. Rehabil. Eng.*, vol. 8, no. 2, Jun. 2000, pp. 222–226.

[11] J. R. Wolpaw, N. Birbaumer, W. J. Heetderks, D. J. McFarland, P. H. Peckham, G. Schalk, E. Donchin, L. A. Quatrano, C. J. Robinson, and T. M. Vaughan, "Brain–computer interface technology: a review of the first international meeting", *IEEE T. Rehabil. Eng.*, vol. 8, no. 2, Jun. 2000, pp 164–173.

[12] A. Kübler, B. Kotchoubey, J. Kaiser, J. R. Wolpaw, N. Birbaumer, "Brain–computer communication: unlocking the locked in", Psychol. Bull., vol. 127, no. 3, May 2001, pp. 358-375.

[13] C. Guger, A. Schlögl, C. Neuper, D. Walterspacher, T. Strein, and G. Pfurtscheller, "Rapid prototyping of an EEG-based brain computer interface (BCI)", *IEEE Trans. Neural Syst. Rehabil. Eng.*, vol. 9, no. 2, Jun. 2003, pp. 1–4.

[14] http://www.gnu.org/software/octave/

[15] G. Lindstrom, "Programming with Python", *IEEE Computer Society*, Oct. 2005, pp. 10–16.

[16] T. E. Oliphant, "Python for scientific computing", Computing in Science and Engineering, University Brigham Young, pp. 10–20.

[17] L. Sörnmo, and P. Laguna, "Bioelectrical signal processing in cardiac and neurological applications", Elsevier Academic Press, 2005, pp. 55–90, 97– 103.

[18] D. J. McFarland, L. M. McCane, S. V. David and J. R. Wolpaw, "Spatial filter selection for EEG-based communication", *Electroen. Clin. Neuro.*, vol. 103, no. 3, Sep. 1997, pp. 386–394.

[19] H. Ramoser, J. Müller-Gerking, and G. Pfurtscheller, "Optimal Spatial Filtering of Single Trial EEG During Imagined Hand Movement", *IEEE Trans. Neural Syst. Rehabil. Eng.*, vol. 8, no. 4, Dec. 2000, pp. 441–446.

[20] C. W. Hsu, C. C. Chang, and C. J. Lin, "A practical guide to support vector classification", May 2009. Available: http://www.csie.ntu.edu.tw/~cjlin/papers/guide/guide.pdf.

[21] G. Dornhege, J. R. Millán, T. Hinterberger, D. J. McFarland, and K. R. Müller (eds.), "Toward brain-computer interfacing", MIT Press, 2007, ch. 4 and 13–15.

[22] W. J. Chun, "Core Python programming", 2nd ed., Prentice Hall Ed., 2007, ch. 1–10.

[23] C. M. Bishop, "Pattern Recognition and Machine Learning", Springer Ed., 2006, ch. 1.