International Journal of Engineering

Journal Homepage: www.ije.ir

Common Spatial Patterns Feature Extraction and Support Vector Machine Classification for Motor Imagery with the SecondBrain

A. Rayatnia[a], R. Khanbabaie*[b]

[a] Department of Computer Engineering, Babol Noshirvani University of Technology, Babol, Iran
[b] Department of Physics, Babol Noshirvani University of Technology, Babol, Iran

PAPER INFO

Paper history:
Received 07 May 2019
Received in revised form 01 June 2019
Accepted 05 July 2019

Keywords:
Second Brain
Common Spatial Patterns
Electroencephalography
Brain–computer Interface
Python
EDF

ABSTRACT

Recently, a large set of electroencephalography (EEG) data is being generated by several high-quality labs worldwide and is free to be used by all researchers in the world. On the other hand, many neuroscience researchers need these data to study different neural disorders for better diagnosis and evaluating the treatment. However, some format adaptation and pre-processing are necessary before using these available data. In this paper, we introduce the SecondBrain as a new lightweight and simplified module that can easily apply various major analysis on EEG data with common data formats. The characteristics of the SecondBrain shows that it is suitable for everyday usage with medium analyzing power. It is easy to learn and accept many data formats. The SecondBrain module has been developed with Python and has the power to windowing data, whitening transform, independent component analysis (ICA), downloading the public datasets, computing common spatial patterns (CSP) and other useful analysis. The SecondBrain, also, employs a common spatial pattern (CSP) to extract features and classifying the EEG MI-based data through support vector machine (SVM). We achieved a satisfactory result in terms of speed and performance.

doi: *10.5829/ije.2019.32.09c.08*

1. INTRODUCTION

A brain–computer interface (BCI) is a tool that helps a computer to communicate directly with the brain. BCIs allow two-way transferring data from the brain to the external devices and vice versa. One direction involves a BCI, which sends brain signals to a machine or computer, and the machine translates brain activity into motor commands for an external device. As a reverse direction, the computer sends information directly to the brain of the BCI user. In that, the brain receives the commands from the machine and translate it into motor commands for organs of the body. BCIs are usually used for different researches, mapping, helping, augmenting, or repairing human sensory-motor or cognitive functions [1].

One of the last steps of the preprocessing before classifying the EEG MI data is a well-known method called Common Spatial Patterns (CSP). The algorithm of the CSP helps to maximize the differences in variance between any two classes of the EEG data. By projecting the EEG signals onto two different classes, the CSP minimizes the variance of one class while at the same time it tries to maximize the variance of the other class. The weights of each EEG channel give one row of the weight matrix. This weight matrix gives the directions that the EEG signals should be projected onto. By using the variances of the projected data, one can extract useful features and classify the EEG signals [2].

Historically, for the first time Koles introduced the CSP as a method to find the abnormal sections of the EEG data [3]. Ramoser et al. [4] then used the CSP to generate features to be used for the classification of event-related desynchronization (ERD) in the EEG data. These ERD occurred due to the imagined movements. Since then the CSP became one of the widely used method for classifying the motor imagery data [5].

Support Vector machines are a branch of supervised learning that contains a group of supervised algorithms

*Corresponding Author Email: *rkhanbabaie@nit.ac.ir* (R. Khanbabaie)

Please cite this article as: A. Rayatnia, R. Khanbabaie, Common Spatial Patterns Feature Extraction and Support Vector Machine Classification for Motor Imagery with the SecondBrain, International Journal of Engineering (IJE), IJE TRANSACTIONS C: Aspects Vol. 32, No. 9, (September 2019) 1284-1289

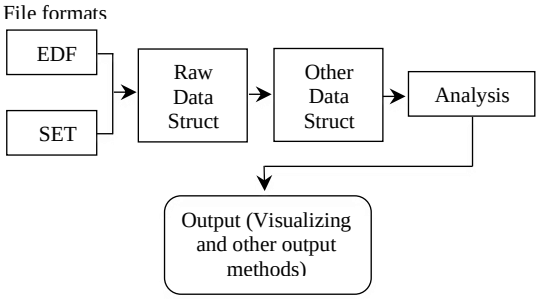


Figure 1. Simple view of sub-modules of the SecondBrain

for classification, detecting outliers and regression. The SVMs are not only very effective in high dimensional space, but also, they are still effective in cases where the number of samples are higher than the number of dimensions. A subset of training points is called support vectors. SVM is memory efficient since it uses support vectors. As various Kernel functions can be defined for the decision function, so SVMs are also flexible. Although usually common kernels are given, the custom kernels can be defined [6].

1. 1. The SecondBrain Here, we introduce a new module, the SecondBrain, which is a simplified module that can easily apply various major analysis on EEG data with common data formats such as EDF, EDF Plus, GDF, SET, etc. We see that the SecondBrain helps to easily extract features and increase the developing and computing speed. Figure 1 shows the flowchart of the sequence of data path in the SecondBrain.

As it is clear from Figure 1, a range of input file readers will transform different data formats to a general raw data structure. Then the general data structure can be transformed to any other data structure. At this point the data is computable. After computation one can get any desired output from the computed data. For example, a data with the format of SET is given. The SecondBrain will read this SET format and will transform to a general raw data structure. Then the user can transform it to any other data structure or not. After that, any computation such as denoising, clustering, classification, stochastic analysis, etc. can be done by the user according to the user's need. At the last step, the output data would be ready for any use.

The SecondBrain has been written in python and cython. The data structure was designed based on object oriented (OO) paradigm but the most of functions have been written based on functional paradigm. The SecondBrain is a prototype, which has been designed to use in real-time system. The analysis sub-module has around 25 major different methods from noise reduction to fast independent component analysis (ICA) which can run in either parallel or sequential. This module has been developed based on test-driven development (TDD) and it has around 68% test coverage.

The novelty of this work focuses on the use in products and analysis of scalable and real time computations. The primary goal of this prototype module was to provide a multilingual model so that developers do not engage in interlingual changes and can easily use it. This module consists of 25 main methods, each of which consists of several functions, so that the user can reach his computational goal without conflict with other methods.

The first important thing about the SecondBrain is that it is a lightweight module. We wrote only 3000 core code which can do the main functionality of EEG processing (it helps you to choose the package you really need and easily integrate) but the MNE-python have around 200k lines of code with 13 heavy package to do full functionality (it is not bad until you want do it in lightweight way). Most of the methods in MNE-python is based on OO that would not let us to fairly compare the efficiency of algorithm neither with stats nor with its architecture, since its time-space complexity is hard to compute because of its complicated architecture[1]. The key difference between MNE-python and the SecondBrain is that the SecondBrain is designed to separate easily (the only dependency is data IO sub-module) and be used everywhere but the MNE-python is designed to do heavy-duty task and user cannot separate methods and use it easily.

The problem with the PyEEG[2] and EEGtools[3] is that they are not complete in main functionality (preprocessing, stats, …). For example, they do not have artifact correction method such as ICA and SSP.

The EEGLAB needs to run over Matlab. Based on the EEGLAB report the octave version is 50% slower than Matlab version[4] and is recommended to have 4GB memory or more. The EEGLAB can run over python with oct2py but it won't work fully functional[5] and it's not compatible with python 2.7 and 3.3. But the SecondBrain can run even on raspberry pi zero easily.

2. METHOD

2. 1. Data Description As our test data, we have used the IV*a* selected dataset of the BCI Competition III from the Berlin BCI Competition IV[6]. Validating signal

[1] https://mne-tools.github.io/0.11/index.html
[2] https://github.com/forrestbao/pyeeg
[3] https://github.com/breuderink/eegtools
[4] https://sccn.ucsd.edu/wiki/EEGLAB_hardware_and_software_recommendations
[5] https://sccn.ucsd.edu/wiki/EEGLAB_and_python
[6] http://www.bbci.de/competition/iii/desc_IVa.html

processing and classification methods for BCI was the goal of the Berlin BCI Competition. The trials of the selected dataset were recorded from five healthy subjects. The number of labeled trials per subject was from 28 to 224.

2. 2. Experimental Setup Based on the Berlin BCI Competition website, there have been five healthy patients labeled with *aa*, *al*, *av*, *aw* and *ay*. The subjects were in a comfortable enviorment, sitting on a chair while their arms were resting on the chair's armrests. The data used for the analysis were obtained from only the first 4 initial sessions with no feedback. By indicating visual cues for 3.5 s, one of the (R) right hand, (L) left hand or (F) right foot motor imageries should be performed by subjects. To let the subject relax during the procedure, the target presentation were paused for random lengths between 1.75 to 2.25 s. Visual stimulation were divided into 2 classes. The first class, which might induces little eye movement correlated to the target, were letters appearing behind a fixation cross. As the second class, which induces eye movements uncorrelated to the target, a randomly moving object was the target.

Two sessions of both the above mentioned classes were recorded from subjects *al* and *aw*. But from the other subjects, 1 session of class (1) and 3 sessions of class (2) were recorded[1]. The recording tools were BrainAmp amplifiers and a 128 channel Ag/AgCl elctrode cap from ECI. The positions of 118 EEG electrodes were based on the extended version of international 10/20-system. These 118 channels were measured and recorded. The data has been filtered by band-pass filter between 0.05 Hz and 200 Hz. Then it has been digitized at 100 Hz and with the accuracy of 16 bit and 0.1μV. In order to prepare the data for usual analysis, a downsampled version of the data was provided by sampling at 100 Hz[1].

2. 3. Preprocessing As the first step of preprocessing, the recorded data was re-referenced and the bandpass filter was applied. Then the data was separated into trials of 3.5 seconds. The re-referencing was done by Common Average Reference or in short term CAR. The Bandpass filtering of the data to alpha and beta frequency was performed using a zero-phase FIR digital filter by processing the data in both forward and reverse directions. The FIR-filter was a 400 taps Blackman-window with a bandwidth of 8-30 Hz. The approximate transition band was 1 Hz. Based on the markers provided with the dataset, the data was split into trials and labeled. The trials without labels were removed.

2. 4. CSP The number of channels of the EEG data was 118. As it is mentioned earlier, the data considered for processing are two classes. Class 1 contains the right-hand data and Class 2 contains the right foot data. For applying the machine learning algorithm, data are split into 2 classes of trials, training and testing. The validation and training data has been separated from the source by bbci team[1], but the training and test data are splitted by shuffle split class of scikit-learn package with test ratio of 0.2 and n_split of 10.

For projecting the data into space, we used the CSP algorithm that makes the variance of the 1st minimal and the variance of the 2nd class maximal. Solving an optimization problem can do this variance optimization, however there is a cost function. The resulting variance of the projected signals (any class) would help us to evaluate the cost function. When the sum of both signal classes variances is fixed, then the cost function value will be minimum [3]. The only dataset used in the CSP algorithm to find spatial filters is the training data. In other words, the task of the CSP algorithm is to calculate a high variance W matrix for one class and low variance W matrix for the other one with spatial filters. The property of W (which is a transformation matrix with the size of M*N, for the definition of M and N see below) is given by the following equations:

$$W(\mathrm{n}) = a_0 - a_1 \cos\left(\frac{2\pi n}{N-1}\right) + a_2 \cos\left(\frac{4\pi n}{N-1}\right) \quad (1)$$

with

$$.a_0 = \frac{1-\alpha}{2},\ a_1 = \frac{1}{2} \text{ and } a_2 = \frac{\alpha}{2} \quad (2)$$

$$Cov\,(WX_1) = D \quad (3)$$

and

$$Cov\,(WX_1) + Cov\,(WX_2) = I. \quad (4)$$

where I is the identity matrix, the diagonal matrix with monotonically descending elements is called D and the covariance of X Matrix is shown by $Cov(X)$ with a rank M ($rank\,(Cov(X)) = M$). The number of channels is given by N. The EEG data for one class (e.g. class I) is stored in the columns of matrix X_i and its observations are stored in the rows of X_i.

The transformed data with a high variance for one class will have a low variance value for the other one. The value of this variance can be used for classifying the data. Equation (3) is equivalent to the CSP equations given below:

$$WCov(WX_1)W^T = D \quad (5)$$

$$WCov(WX_2)W^T = I - D \quad (6)$$

To transform the data to have an identity matrix we can basically use a principal component analysis (PCA) and normalize the variance to one. To do that we use singular value decomposition (SVD) to calculate a matrix P as follows:

[1] http://www.bbci.de/competition/iii/desc_IVa.html

$$Cov(PX) = I \rightarrow Cov(X) = (P^T P)^{-1} = U\lambda U^T \quad (7)$$

$$P = \sqrt{\lambda^{-1}}U \quad (8)$$

where λ is a diagonal matrix and U is an orthogonal matrix. When the rank of *Cov(X)* is less than *M*, then the significant eigenvectors and eigenvalues are enough to calculate *P*.

2. 5. SVM The SecondBrain uses support vector machine (SVM) package, as a tool in data mining projects and as a linear binary non-probabilistic classification tool. SVM can also be used in a setting of probabilistic classification, like Platt method.

SVMs are a set of supervised learning methods for analyzing a data and recognizing the patterns. In machine learning, SVMs are used for classification and category prediction. Based on the provided examples, an SVM model will predict which class a subject belongs to [7, 8]. SVMs can perform linear and nonlinear classification efficiently. The nonlinear categorization is done using the kernel trick, which implicitly maps the inputs into high-dimensional feature spaces.

There are various number of kernel functions that different kernel-based learning algorithms use them. Among these kernel functions, the radial basis function (RBF), also called Gaussian kernel, is well-known. SVMs usually use RBF. If we consider two samples, they can be represented as two feature vectors, like x and x', in an input space. Then the RBF kernel on these two samples is defined as follows [9, 10]:

$$K(x, x') = \exp(-\gamma||x - x'||^2) \quad (9)$$

where the spread of the kernel is set by γ.

Another commonly used group of kernels used with SVMs is polynomial kernel. The similarity of training samples in a feature space can be represented by polynomials of the variable. This will allow non-linear models-based learning [11].

3. RESULTS

With the implementation of our program as a part of the SecondBrain, we achieved significant results in terms of speed (computing time) and cost (resource allocation). The specification of the computer used is given in Table 1. As can be seen in Table 2, using a support vector machine with linear kernel, we got a precision of 75% and a recall rate of 73%. But what matters is the real-time execution for the app's end user. By executing the code by a computer with the specification given below (Table 1), the total time that the end user was waiting was around 1.62 seconds (Table 3).

Worth noting in the results the change occurred in learning precision by changing the SVM's kernel. As you can see, the learning curve with the linear kernel (Figure 2a) gives a better response than the polynomial (Figure 2b), RBF (Figure 2c) and sigmoid (Figure 2d). The reason for this is the separation of samples and the small number of classes that can be easily and more accurately classified linearly. Note that the fault tolerance of the iterations is given by color margins in the figures.

Another point, which is very important in the speed of computing, is the normalization of data. If data is not normalized, its use will slow down the calculation significantly. One more thing to add is that linear SVM is less prone to overfitting than the non-linear SVM. The Average CPU time for 700 times in 1000 loops was 0.19 ns.

TABLE 1. The specification of the computer

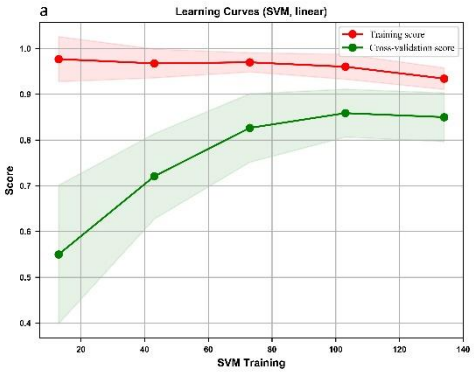
Processor	Intel Core-i5 processor (5257U) 2.7 GHz
Memory	8 GB 1867 MHz DDR3
OS	Mac OSX 10.12.6
Device	macbook pro MF-840

TABLE 2. Accuracy measure

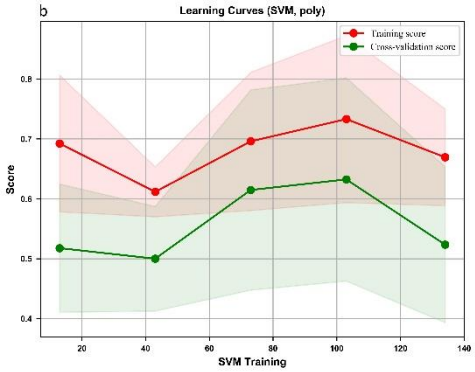
Class	Precision	Recall	F1-score	Support
1	0.69	0.90	0.78	60
2	0.82	0.54	0.65	52
Average/Total	0.75	0.73	0.72	112

TABLE 3. Running Time

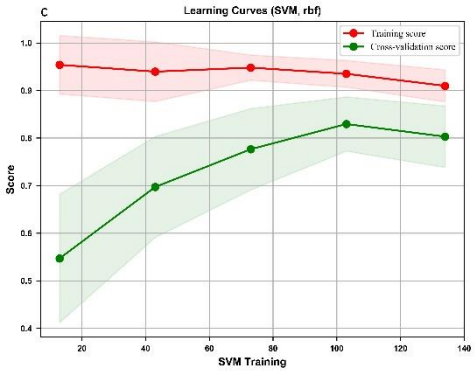
Real	1.62 s
System	0.62 s
User	1.0 s



(a)



(b)



(c)

(d)

Figure 2. Learning curve with (a) linear, (b) polynomial, (c) RBF, and (d) sigmoid kernel

4. CONCLUSIONS

The final goal of this work was to test the correctness and performance of our newly developed lightweight and simplified module, the SecondBrain.

The SecondBrain is compared with other available modules for EEG processing in terms of programming language, data input/output, monitoring method, analyzing power, visualizing power, easy to learn, and the variety of the data sources accessible to public.

TABLE 4. Comparing the SecondBrain with other available EEG processing toolboxes

Name	PyEEG	MNE-Python	Second-Brain	EEGLab
Language	Python 2	Python2-3, C++	Cython, Python2, Python3	MATLAB
Data I/O	EDF	EDF, GDF, BDF, FIFF, etc.	EDF, GDF, SET	EDF, GDF, MAT, ASC, etc.
Monitoring method	EEG	EEG/MEG, ECoG, etc.	EEG	EEG
Analysis*	weak	strong	medium	strong
Visualizing	No	Yes	Yes†	Yes
Easy to learn	It is not well documented	heavy module, hard to learn in true way	easy to learn, well documented	easy to use in GUI mode
Access to public data set	Yes♣	Yes	Yes	Yes

*Analysis: it means how much this module is used on a daily basis.
†The SecondBrain can visualize the output data but does not support topographical density map and heatmap.
♣The PyEEG is weak in accessibility to public data, since its variety of data sources is low.

The characteristics of the SecondBrain shows that it is suitable for everyday usage with medium analyzing power. It is easy to learn and accept many data formats. Its programming language is Python which is free and simple. We achieved a satisfactory result in terms of speed and performance. To improve the target functionality, we can use the generalization method and decreasing the space dimensions to achieve more accurate classification and results.

Also, for commercial use with classifying and learning data in real-time, we are able to use this module for disabled people in the real world. In the next steps of this project, we will work on multilingual feature and GPU processing.

5. ACKNOWLEDGEMENTS

This work was supported by the Babol Noshirvani University of Technology under research grant number BNUT/388054/97 to Reza Khanbabaie. Authors declare that the study sponsors had no involvement in the study design, in the collection, analysis and interpretation of data; in the writing of the manuscript; and in the decision to submit the manuscript for publication.

6. REFERENCES

1. Issa, T., Kommers, P., Issa, T., Isaias, P. and Issa, T.B., "Smart technology applications in business environments, IGI Global, (2017).
2. Yong, X., Ward, R.K. and Birch, G.E., "Robust common spatial patterns for eeg signal preprocessing", in 2008 30th Annual International Conference of the IEEE Engineering in Medicine and Biology Society, IEEE. Vol., No. Issue, (2008), 2087-2090.
3. Koles, Z.J., Lazar, M.S. and Zhou, S.Z., "Spatial patterns underlying population differences in the background eeg", ***Brain Topography***, Vol. 2, No. 4, (1990), 275-284.
4. Ramoser, H., Muller-Gerking, J. and Pfurtscheller, G., "Optimal spatial filtering of single trial eeg during imagined hand movement", ***IEEE Transactions on Rehabilitation Engineering***, Vol. 8, No. 4, (2000), 441-446.
5. Popescu, F., Fazli, S., Badower, Y., Blankertz, B. and Müller, K.-R., "Single trial classification of motor imagination using 6 dry eeg electrodes", ***PloS one***, Vol. 2, No. 7, (2007), e637.
6. Cortes, C. and Vapnik, V., "Support-vector networks", ***Machine Learning***, Vol. 20, No. 3, (1995), 273-297.
7. Lemm, S., Blankertz, B., Curio, G. and Muller, K.-R., "Spatio-spectral filters for improving the classification of single trial eeg", ***IEEE Transactions on Biomedical Engineering***, Vol. 52, No. 9, (2005), 1541-1548.
8. Miner, L., Bolding, P., Hilbe, J., Goldstein, M., Hill, T., Nisbet, R., Walton, N. and Miner, G., "Practical predictive analytics and decisioning systems for medicine: Informatics accuracy and cost-effectiveness for healthcare administration and delivery including medical research, Academic Press, (2014).
9. Vert, J.-P., Tsuda, K. and Schölkopf, B., "Kernel methods in computational biology", MIT Press, (2004).
10. Xu, G., Zong, Y. and Yang, Z., "Applied data mining, CRC Press, (2013).
11. Ben-Hur, A. and Weston, J., A user's guide to support vector machines, in Data mining techniques for the life sciences. 2010, Springer.223-239.

Common Spatial Patterns Feature Extraction and Support Vector Machine Classification for Motor Imagery with the SecondBrain

A. Rayatnia[a], R. Khanbabaie[b]

[a] *Department of Computer Engineering, Babol Noshirvani University of Technology, Babol, Iran*
[b] *Department of Physics, Babol Noshirvani University of Technology, Babol, Iran*

P A P E R I N F O

Paper history:
Received 07 May 2019
Received in revised form 01 June 2019
Accepted 05 July 2019

Keywords:
Second Brain
Common Spatial Patterns
Electroencephalography
Brain–computer Interface
Python
EDF

چکیده

اخیرا مجموعه ای از داده های EEG توسط چندین آزمایشگاه با کیفیت بالا در سراسر جهان تولید شده است که می تواند توسط همه محققین در جهان مورد استفاده قرار گیرد. از سوی دیگر، بسیاری از محققان علوم اعصاب نیازمند این اطلاعات برای مطالعه اختلالات عصبی مختلف برای تشخیص بهتر و ارزیابی درمان هستند. با این حال، قبل از استفاده از این اطلاعات موجود، برخی هم سازگارسازی و پیش پردازش اطلاعات مورد نیاز است. در این مقاله، SecondBrain را به عنوان یک ماژول سبک و ساده معرفی می کنیم که به راحتی می تواند تجزیه و تحلیل های متنوع و عمده ای را در داده های EEG با فرمت های رایج رایج انجام دهد. ویژگی های SecondBrain نشان می دهد که ماژول مناسبی برای استفاده روزمره با قدرت تجزیه و تحلیل متوسط است. بسیاری از فرمت های داده یادگیری و پذیرش آسان است. ماژول SecondBrain با Python توسعه داده شده و توانایی تبدیل اطلاعات سفید، تبدیل ICA، دانلود مجموعه داده های عمومی، محاسبه الگوهای فضایی مشترک (CSP) و سایر تحلیل های مفید است. SecondBrain نیز یک الگوی فضایی مشترک (CSP) را برای استخراج ویژگی ها و طبقه بندی داده های مبتنی بر MI EEG از طریق دستگاه بردار پشتیبانی (SVM) استفاده می کند. ما به نتیجه رضایت بخش در سرعت و عملکرد دست یافتیم.

***doi**: 10.5829/ije.2019.32.09c.08*