Potential pitfalls of widely used implementations of common spatial patterns

Milan Rybář[1], Ian Daly[1] and Riccardo Poli[1]

***Abstract*— We have uncovered serious flaws in handling EEG signals with a decreased rank in implementations of the common spatial patterns (CSP). The CSP algorithm assumes covariance matrices of the signal to have full rank. However, preprocessing techniques, such as artifact removal using independent component analysis, may decrease the rank of the signal, leading to potential errors in the CSP decomposition. We inspect what could go wrong when CSP implementations do not take this into consideration on a binary motor imagery classification task. We review CSP implementations in open-source toolboxes for EEG signal analysis (FieldTrip, BBCI Toolbox, BioSig, EEGLAB, BCILAB, and MNE). We show that unprotected implementations decreased mean classification accuracy by up to 32%, with spatial filters resulting in complex numbers, for which corresponding spatial patterns do not have a clear interpretation. We encourage researchers to check their implementations and analysis pipelines.**

I. INTRODUCTION

The common spatial patterns (CSP) algorithm [1], [2] is a popular supervised decomposition method for the EEG signal analysis which is used to distinguish between two classes (conditions). It finds spatial filters that maximize the signal variance for one class, while simultaneously minimizing the signal variance for the opposite class. Derivations of CSP assume that two covariance matrices of two classes have full rank. However, preprocessing techniques may decrease the rank of the signal. We highlight this issue using independent component analysis (ICA) for artifact removal [3], [4]. We focus on ICA in this paper but our message applies for any technique with this feature.

A general solution has been shown in the literature [5]. Nevertheless, several studies using ICA for artifact removal followed by CSP in the original channel space struggle with this issue and often walk around the problem. For instance, work reported in [6] computed CSP on ICs to avoid the issue, while work in [7] used a preprocessing transformation to obtain a full rank matrix. On the other hand, many studies use the same pipeline without mentioning the issue. We highlight this issue here because in many cases it is unclear how other authors have solved this issue.

In this paper, we show what could happen when CSP implementations do not take this issue into consideration on a binary motor imagery classification task of EEG trials from the BCI competition III dataset IVa [8]. Additionally, we review implementations in open-source toolboxes for EEG signal analysis.

[1]Brain-Computer Interfacing and Neural Engineering Laboratory, School of Computer Science and Electronic Engineering, University of Essex, UK. Email: {milan.rybar, i.daly, rpoli}@essex.ac.uk

II. COMMON SPATIAL PATTERNS

Here, we review the CSP algorithm and its two main implementation approaches. We assume that the EEG is already band-pass filtered and centered. Let $X_i \in \mathbb{R}^{C\times T}$ be the EEG signal of trial i where C is the number of channels and T is the number of samples per trial. We compute the spatial covariance $R_1 \in \mathbb{R}^{C\times C}$ by averaging over trials of class 1:

$$R_1 = \frac{1}{|\mathcal{I}_1|}\sum_{i\in\mathcal{I}_1}\frac{X_i X_i^T}{\mathrm{tr}(X_i X_i^T)} \tag{1}$$

where $\mathcal{I}_1$ is the set of indices corresponding to trials belonging to class 1, $|\mathcal{I}_1|$ denotes the size of the set $\mathcal{I}_1$, and tr is the trace of a matrix, and spatial covariance R_2 equivalently for class 2. In the following derivations of CSP, we assume that R_1 and R_2 have full rank (i.e, $\mathrm{rank}(R_1) = \mathrm{rank}(R_2) = C$).

The goal of CSP is to find a decomposition matrix $W \in \mathbb{R}^{C\times C}$ that projects the signal $x(t) \in \mathbb{R}^C$ in the original channel space to $x_{CSP}(t) \in \mathbb{R}^C$ as follows:

$$x_{CSP}(t) = W^T x(t) \tag{2}$$

with the following properties:

$$W^T R_1 W = D_1 \tag{3}$$

$$W^T R_2 W = D_2 \tag{4}$$

and scaling such that

$$D_1 + D_2 = I_C \tag{5}$$

where $I_C \in \mathbb{R}^{C\times C}$ is the identity matrix. In other words, R_1 and R_2 share the same eigenvectors and the sum of the corresponding eigenvalues is always 1. The eigenvector with the largest eigenvalue for class 1 has the smallest eigenvalue for class 2 and vice-versa. Columns of W are spatial filters while columns of a matrix $A = (W^T)^{-1}$ represent spatial patterns.

A. Geometric approach

We factorize the composite spatial covariance $R_1 + R_2$ as

$$R_1 + R_2 = EFE^T \tag{6}$$

where E is the orthogonal matrix of eigenvectors (in columns) and F is the diagonal matrix of their corresponding eigenvalues. We define the whitening transformation matrix U as

$$U = F^{-1/2}E^T \tag{7}$$

and whiten matrix R_1

$$S_1 = UR_1U^T. \tag{8}$$

We factorize matrix S_1 as

$$S_1 = PD_1P^T \tag{9}$$

where P is the orthogonal matrix of eigenvectors and D_1 is the diagonal matrix of their corresponding eigenvalues. We define the decomposition matrix W^T as

$$W^T = P^TU. \tag{10}$$

Then this W satisfies (3) and also (4) using (5).

B. Generalized eigenvalue problem

We can directly solve W by getting W^T from (5) [5] and by inserting this into (3) we get

$$R_1W = D_1(R_1 + R_2)W, \tag{11}$$

which is an equation of the generalized eigenvalue problem.

C. Covariance matrices without full rank

If the covariance matrices R_1 and R_2 do not have full rank, the above CSP derivations do not hold. Putting aside mathematical incorrectness, what could go wrong in their direct implementations?

In the geometric approach, the first eigendecomposition in (6) may have some zero eigenvalues. In the case of using ICA for artifact removal, the number of zero eigenvalues equals the number of removed ICs. The whitening transformation U in (7) is undefined due to division by zero. We can remove dimensions with zero eigenvalues at this point and the rest would work. This is similar to dimensionality reduction by principal component analysis (PCA) before CSP.

In the generalized eigenvalue problem approach, the generalized eigendecomposition in (11) may have a complex solution. The complex spatial filters and their corresponding complex spatial patterns do not have a clear interpretation.

In both cases, the EEG should be first projected into a space with the number of dimensions equal to the rank of the EEG before CSP decomposition. We use PCA in this paper, see [5] for a general solution and [2] for the difference of CSP solution on spatially filtered data. The covariance matrices will have full rank in this space. Note that to compute spatial patterns on the original EEG channels, we must first multiply W^T with the PCA transformation matrix before the inversion.

III. METHODS

A. Evaluation

We evaluated CSP implementations on a binary motor imagery classification task using ICA for artifact removal in a preprocessing step. We chose this pipeline because it is commonly used and the complex number problem is hidden by the classifier. We used public dataset IVa from the BCI competition III [8]. The single-trial EEG signals were recorded from five healthy participants during imagination of right hand and right foot movement without feedback (140 trials per class), see [8] for more details.

First, EEG signals were preprocessed by removing artifacts. EEG signals were FIR band-pass filtered between 1-40 Hz to remove slow drifts in the signal and high-frequency noise. For each participant, ICA (FastICA) was trained on all windows 0-4.5 s after the task onset. Artifactual components were identified by thresholding peak amplitudes of the EEG time series [9], [10]. A scalp projection of each IC was thresholded to $\pm 100\,\mu V$ and peak-to-peak differences between maximum and minimum amplitudes in each window and channel were thresholded to $60\,\mu V$, with ICs exceeding any criterion marked for removal (35, 32, 37, 63, and 23 ICs were removed from a total of 118 for the 5 participants).

We adopted the winning solution from the BCI competition using CSP [11]. EEG signals were FIR band-pass filtered between 12-14 Hz and trials were extracted from 0.5-4.5 s after the task onset. Additionally, dimensionality reduction by PCA on all trials was used if it was required by a particular CSP implementation.

For each CSP implementation, we trained a classifier using stratified 10-fold cross-validation and measured classification accuracy from all test folds. CSP was trained and even numbers of CSP components, columns of W, from 2 to 20 were selected. Components were ordered by sorting their corresponding eigenvalues in ascending order and the first $k/2$ and last $k/2$ components were selected for a desired k components. We choose 20 as a maximum because we have not seen any usage of more CSP components reported. The winning solution from the BCI competition, we adopted here, used 2 CSP components. The classifier features were the log-variance of the selected components [1], [2]. Linear discriminant analysis (LDA) and support-vector machine (SVM), with the radial basis function kernel and the regularization parameter $C = 1$, were used as classifiers.

B. Implementations

In Python, there are two eigendecomposition methods, for the geometric approach, *eig* and *eigh* in the numpy and scipy packages. Implementations in both packages have the same behavior. The *eigh* is a specialized method for a real symmetric matrix, which always produces a real solution, while *eig* is for a general matrix. Nevertheless, *eigh* does not check this assumption and only uses the upper or lower triangular part. Similarly, two generalized eigendecomposition methods *eig* and *eigh* are implemented in the scipy package with the same logic as above. The *eigh* method raises an exception when the second matrix, $R_1 + R_2$ in (11) in our case, is not definite positive. We will refer to the used method in parentheses. In Matlab, there is only one method *eig* for everything.

We tested CSP implementations in Python 3.6 and Matlab R2018b Update 4, both 64-bit, with all the above permutations. The classification pipeline was implemented in Python and the only difference was a particular CSP implementation that ran in its required environment, directly Python or Matlab in a subprocess. [1] In the geometric approach, the same eigendecomposition method is used for both eigendecompositions in (6) and (9), and with and without removing dimensions during the whitening step with eigenvalues

[1] Source code for the classification pipeline with all CSP implementations is available at `https://github.com/milan-rybar/csp_evaluation`.

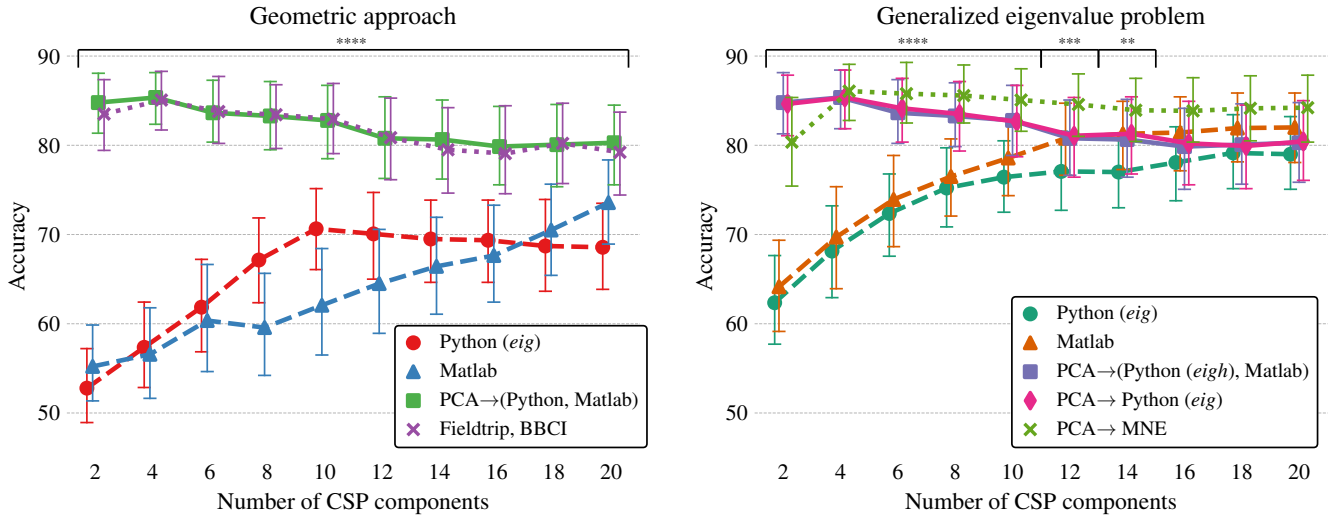

Fig. 1. Mean classification accuracy over all test folds from cross-validation (10) and all participants (5) for different CSP implementations. Unprotected CSP implementations are shown with dashed lines, protected with solid lines, and external implementations with dotted lines. Error bars are 95% confidence intervals. Points for a particular number of CSP components are offset on the X axis for better visibility. Intervals of numbers of CSP components with asterisks represent statistical difference between our CSP implementations for the particular number of CSP components by Friedman test where **** is $p < 0.0001$, *** is $p < 0.001$, and ** is $p < 0.01$. 'PCA→' denotes dimensionality reduction by PCA before the CSP algorithm.

smaller than 10^{-14} after (6). All these possibilities are used with and without dimensionality reduction by PCA before CSP. We refer to the CSP implementation with the correct mathematical background 'protected' CSP and otherwise refer to CSP as 'unprotected'.

C. EEG Toolboxes

We review CSP implementations in popular open-source toolboxes for the EEG analysis in Matlab and Python.

1) FieldTrip: FieldTrip (v. 20191025, October 2019) [12] for Matlab implements a geometric approach but uses singular value decomposition instead of eigenvalue decomposition. During the whitening step, it removes dimensions with eigenvalues of absolute value smaller than 10^{-14}.

2) BBCI Toolbox: BBCI Toolbox (commit a30ce0bc8d, March 2019) [13] for Matlab implements a geometric approach. During the whitening step, it keeps dimensions with eigenvalues larger than the fraction 10^{-10} of the largest eigenvalue. It shows a warning of this dimensionality reduction when it is applied.

3) BioSig: BioSig (v. 3.6.0, April 2019) [14] for Matlab implements both approaches. Both of them are unprotected. The implementation always returns only four spatial filters, two per class, thus we excluded it for our evaluation.

4) EEGLAB, BCILAB: EEGlab [15] for Matlab has 2 plugins with CSP implementations. CSP plugin (v. 1.1) implements a geometric approach and BCILAB (v. 1.1) [16] solves a generalized eigenvalue problem. Both of them are unprotected. Both implementations are difficult to adapt for our programmatic evaluation on given trials outside their desired processing pipelines without taking the code outside, thus we excluded them for our evaluation.

5) MNE: MNE (v. 0.17.2, April 2019) [17] for Python solves a generalized eigenvalue problem. It uses the *eigh* method for real symmetric matrices, which raises an exception when the covariance matrix $R_1 + R_2$ in (11) is not definite positive.

IV. RESULTS

We mainly compare our CSP implementations because we can guarantee that they differ only in the inspected part. Other implementations may differ in their definition of the covariance matrix in (1) and selection of CSP components. For instance, MNE implements different selection criterion based on [2]. Their results are provided mainly as a reference. Two methods are compared by their differences in classification accuracies on each test fold from cross-validation (10) and each participant (5). Using LDA and SVM gave similar results, thus we decided to report results only for LDA as a simpler classifier. Figure 1 shows mean classification accuracies for all tested CSP implementations.

A. Geometric approach

Unprotected Python (*eigh*) encountered division by zero in (7) due to zero eigenvalues. On the other hand, unprotected Python (*eig*) is a little bit tricky. The *eig* has a complex solution in (6) and it depends on the particular data how "close" complex eigenvalues are to zero, whether it raises division by zero or not in (7).

Reducing dimensionality during the whitening step (i.e, removing eigenvectors with zero eigenvalues after (6)) or dimensionality reduction by PCA before the CSP (or both together) had equivalent results on the classification pipeline for any classifier, Python (*eig*, *eigh*) and Matlab implementation. We will group their results together and call them protected regardless what dimensionality reduction is used. However, Python (*eig*) with dimensionality reduction during the whitening step (without PCA) returned W and

eigenvalues in complex numbers but with zero imaginary parts. Matlab did not have this issue. Additionally, these methods also gave equivalent results to protected Python (*eigh*) and protected Matlab in the generalized eigenvalue problem approach.

The difference in classification accuracy between unprotected Python (*eig*) and protected Python (*eig*, *eigh*) or Matlab, which were equal as described above, was statistically significant when using any number of CSP components ($p < 0.0001$, one-sided Wilcoxon signed-rank test with Pratt modification for zero-differences). The difference decreased from 32 ± 2.1 (mean $\pm$ standard error) for 2 components to 11.7 ± 1.3 for 20 components. The difference between unprotected Matlab and protected Matlab was statistically significant for 2 to 16 components ($p < 0.0001$), and 18 components ($p < 0.01$). The difference decreased from 29.5 ± 2.2 for 2 components to 9.5 ± 2.7 for 18 components. Both unprotected Python (*eig*) and Matlab had complex solutions, thus the resulting W and eigenvectors were complex.

B. Generalized eigenvalue problem

Unprotected Python (*eigh*) raised an exception because the covariance matrix $R_1 + R_2$ is not definite positive.

The difference in classification accuracy between unprotected Python (*eig*), protected Python (*eig*), and protected Python (*eigh*) was statistically significant for 2 to 12 components ($p < 0.0001$, Friedman test), and for 14 components ($p < 0.01$). The difference in classification accuracy between unprotected Python (*eig*) and protected Python (*eigh*) or Matlab, which were equal to protected CSP in geometric approach as described above, was statistically significant for 2 to 10 components ($p < 0.0001$, one-sided Wilcoxon test) and for 12 to 14 components ($p < 0.01$). The difference decreased from 22.4 ± 2.2 for 2 components to 3.6 ± 1.4 for 14 components. Similarly, the difference between unprotected Python (*eig*) and protected Python (*eig*) was statistically significant for 2 to 10 components ($p < 0.0001$), for 12 components ($p < 0.001$), and for 14 components ($p < 0.01$). The difference decreased from 22.2 ± 2.2 for 2 components to 4.2 ± 1.3 for 14 components. The difference between unprotected Matlab and protected Matlab was statistically significant for 2 to 6 components ($p < 0.0001$), 8 components ($p < 0.01$), and less for 10 components ($p < 0.05$). The difference decreased from 20.6 ± 2.4 for 2 components to 6.7 ± 2.1 for 8 components.

Unprotected Python (*eig*) and unprotected Matlab had a complex solution, thus the resulting W and eigenvectors were complex. Protected Python with *eig* and *eigh* did not have equivalent results, but they were not statistically different for any number of CSP components (two-sided Wilcoxon test). Surprisingly, protected Python (*eig*) had complex eigenvalues with zero imaginary parts but a W with real numbers (before components selection). Further inspection showed that the difference in their eigenvalues D_1 was always less than 10^{-12} but their W were different, not just flipped signs or reverse ordering.

V. CONCLUSION

We showed that unprotected CSP implementations in Python and Matlab can significantly decrease accuracy on a binary motor imagery classification task. Our results suggest that the less CSP components are used the higher the decrease in classification accuracy between protected and unprotected CSP implementations. In Python, we strongly recommend using the *eigh* method in both CSP implementation approaches. We encourage researchers to check their implementations.

REFERENCES

[1] J. Müller-Gerking, G. Pfurtscheller, and H. Flyvbjerg, "Designing optimal spatial filters for single-trial EEG classification in a movement task," *Clinical Neurophysiology*, vol. 110, no. 5, pp. 787–798, 1999.

[2] B. Blankertz, R. Tomioka, S. Lemm, M. Kawanabe, and K.-r. Muller, "Optimizing Spatial filters for Robust EEG Single-Trial Analysis," *IEEE Signal Processing Magazine*, vol. 25, no. 1, pp. 41–56, 2008.

[3] T.-P. Jung, S. Makeig, C. Humphries, T.-W. Lee, M. J. McKeown, V. Iragui, and T. J. Sejnowski, "Removing electroencephalographic artifacts by blind source separation," *Psychophysiology*, vol. 37, no. 2, pp. 163–178, 2000.

[4] M. K. Islam, A. Rastegarnia, and Z. Yang, "Methods for artifact detection and removal from scalp EEG: A review," *Clinical Neurophysiology*, vol. 46, no. 4-5, pp. 287–305, 2016.

[5] L. C. Parra, C. D. Spence, A. D. Gerson, and P. Sajda, "Recipes for the linear analysis of EEG," *NeuroImage*, vol. 28, no. 2, pp. 326–341, 2005.

[6] I. Winkler, S. Haufe, and M. Tangermann, "Automatic Classification of Artifactual ICA-Components for Artifact Removal in EEG Signals," *Behavioral and Brain Functions*, vol. 7, no. 1, p. 30, 2011.

[7] A. Ferrante, C. Gavriel, and A. Faisal, "Data-efficient hand motor imagery decoding in EEG-BCI by using Morlet wavelets & Common Spatial Pattern algorithms," in *2015 7th International IEEE/EMBS Conference on Neural Engineering*. IEEE, 2015, pp. 948–951.

[8] B. Blankertz, K. Muller, D. Krusienski, G. Schalk, J. Wolpaw, A. Schlogl, G. Pfurtscheller, J. Millan, M. Schroder, and N. Birbaumer, "The BCI Competition III: Validating Alternative Approaches to Actual BCI Problems," *IEEE Transactions on Neural Systems and Rehabilitation Engineering*, vol. 14, no. 2, pp. 153–159, 2006.

[9] E. Niedermeyer, "The Normal EEG of the Waking Adult," *Electroencephalography: Basic Principles, Clinical Applications and Related Fields*, 1999.

[10] S. P. Fitzgibbon, D. M. W. Powers, K. J. Pope, and C. R. Clark, "Removal of EEG Noise and Artifact Using Blind Source Separation," *Journal of Clinical Neurophysiology*, vol. 24, no. 3, pp. 232–243, 2007.

[11] Y. Wang, "Classifying Single-Trial EEG during Motor Imagery with a Small Training Set," *ArXiv*, vol. abs/1306.3, 2013.

[12] R. Oostenveld, P. Fries, E. Maris, and J.-M. Schoffelen, "FieldTrip: Open Source Software for Advanced Analysis of MEG, EEG, and Invasive Electrophysiological Data," *Computational Intelligence and Neuroscience*, vol. 2011, pp. 1–9, 2011.

[13] B. Blankertz, L. Acqualagna, S. Dähne, S. Haufe, M. Schultze-Kraft, I. Sturm, M. Uščumlic, M. A. Wenzel, G. Curio, and K.-R. Müller, "The Berlin Brain-Computer Interface: Progress Beyond Communication and Control," *Frontiers in Neuroscience*, vol. 10, p. 530, 2016.

[14] C. Vidaurre, T. H. Sander, and A. Schlögl, "BioSig: The Free and Open Source Software Library for Biomedical Signal Processing," *Computational Intelligence and Neuroscience*, vol. 2011, pp. 1–12, 2011.

[15] A. Delorme and S. Makeig, "EEGLAB: an open source toolbox for analysis of single-trial EEG dynamics including independent component analysis," *Journal of Neuroscience Methods*, vol. 134, no. 1, pp. 9–21, 2004.

[16] C. A. Kothe and S. Makeig, "BCILAB: a platform for brain–computer interface development," *Journal of Neural Engineering*, vol. 10, no. 5, p. 056014, 2013.

[17] A. Gramfort, M. Luessi, E. Larson, D. Engemann, D. Strohmeier, C. Brodbeck, R. Goj, M. Jas, T. Brooks, L. Parkkonen, and M. Hämäläinen, "MEG and EEG data analysis with MNE-Python," *Frontiers in Neuroscience*, vol. 7, p. 267, 2013.