RESEARCH ARTICLE

A Neural Network-Based Optimal Spatial Filter Design Method for Motor Imagery Classification

Ayhan Yuksel*, Tamer Olmez

Department of Electronics and Communication Engineering, Istanbul Technical University, Istanbul, Turkey

* yukselay@itu.edu.tr

Abstract

In this study, a novel spatial filter design method is introduced. Spatial filtering is an important processing step for feature extraction in motor imagery-based brain-computer interfaces. This paper introduces a new motor imagery signal classification method combined with spatial filter optimization. We simultaneously train the spatial filter and the classifier using a neural network approach. The proposed spatial filter network (SFN) is composed of two layers: a spatial filtering layer and a classifier layer. These two layers are linked to each other with non-linear mapping functions. The proposed method addresses two shortcomings of the common spatial patterns (CSP) algorithm. First, CSP aims to maximize the between-classes variance while ignoring the minimization of within-classes variances. Consequently, the features obtained using the CSP method may have large within-classes variances. Second, the maximizing optimization function of CSP increases the classification accuracy indirectly because an independent classifier is used after the CSP method. With SFN, we aimed to maximize the between-classes variance while minimizing within-classes variances and simultaneously optimizing the spatial filter and the classifier. To classify motor imagery EEG signals, we modified the well-known feed-forward structure and derived forward and backward equations that correspond to the proposed structure. We tested our algorithm on simple toy data. Then, we compared the SFN with conventional CSP and its multi-class version, called one-versus-rest CSP, on two data sets from BCI competition III. The evaluation results demonstrate that SFN is a good alternative for classifying motor imagery EEG signals with increased classification accuracy.

OPEN ACCESS

Citation: Yuksel A, Olmez T (2015) A Neural Network-Based Optimal Spatial Filter Design Method for Motor Imagery Classification. PLoS ONE 10(5): e0125039. doi:10.1371/journal.pone.0125039

Academic Editor: Luigi Bianchi, Univerity of Rome Tor Vergata, ITALY

Received: October 2, 2014

Accepted: March 6, 2015

Published: May 1, 2015

Data Availability Statement: The algorithm is available within the Supporting Information files. EEG data are owned by a third party, and are available upon request at the web site of the Brain Computer Interface Competititon III: http://www.bbci.de/competition/iii/.

Funding: The authors have no support or funding to report.

Competing Interests: The authors have declared that no competing interests exist.

Introduction

A Brain-Computer Interface (BCI) is an alternative method of communication between a user and system in which the user does not need to use his/her brain-muscular pathways to control an external device [1]. Because it is a direct communication method with the brain and outer world, the BCI system emerges as a useful communication and control method for severely paralyzed people. In such a system, the user should generate different signal patterns with his brain for different commands. Moreover, discriminating these brain patterns (typically EEG) and translating them to control commands for an electronic device is the most important part of the BCI system. In Motor Imagery (MI)-based BCI systems, the kinaesthetic imagination of body movement results in oscillations called event-related synchronization/desynchronization in the sensorimotor cortex in the μ and β frequency bands [2, 3].

Due to the topographical organization in the motor cortex, different motor imagery tasks can be identified based on their specific spatial location of related ERD rhythms [4]. However, due to the volume conduction effect, scalp EEG signals recorded from a specific area involve a mixture of several cortical sources located in different areas. Thus, raw scalp EEG potentials have poor spatial resolution [4]. To eliminate the volume conduction effect and reach the actual underlying signal sources, a spatial filtering step is an indispensable technique [5].

Common spatial patterns (CSP) is a very popular and powerful spatial filtering method used in motor imagery EEG classification [6]. When using band power features, CSP computes spatial filters, aiming to obtain optimal discrimination between two classes [7]. CSP finds optimal spatial filters that maximize the ratio of average variances that belong to two different classes. Computationally, CSP is solved by simultaneously diagonalizing the two covariance matrices of the two classes [8]. A computed CSP spatial filter projects the multi-dimensional EEG time domain signal to a one-dimensional time domain signal in which the power (variance) of one class is maximized while the power of the other class is minimized. Unlike PCA, CSP handles two classes at the same time and simultaneously diagonalizes the covariance matrices of both classes [9]. Moreover, the CSP algorithm was proven to be efficient in BCI competitions [10, 11].

Although CSP is a powerful and simple technique, it has some drawbacks. CSP optimizes the average power ratio of the two classes, and therefore, it requires only one average covariance matrix for each class. This may be a problem when addressing non-stationary signals such as EEG signals because the covariance matrix of an EEG signal may change over time due to artifacts such as changes in EEG electrode-skin impedances, muscular activities or user background EEG activity [7]. Representing all of the epochs of a class in a training set by only one average covariance matrix should result in inaccurate spatial filters.

Another disadvantage of CSP is its strict fitness function. CSP does not allow different types of fitness functions, which may be more useful in different situations [12]. CSP attempts to optimize the Rayleigh quotient, i.e., the ratio of average variances of the two classes, which is very sensitive to outliers that cause over fitting [13].

There are numerous methods for increasing the robustness of CSP. Recently, a method called regularized CSP (RCSP) has been proposed, which aims to compute more robust spatial patterns by adding a regularization term to the CSP formula [5, 7, 14]. The RCSP method uses some a priori knowledge and imposes various constraints in the CSP's formulation to obtain more robust spatial filters [12]. For example, Lotte [14] proposed spatially regularized CSP (SRCSP). He used the a priori knowledge that neighboring neurons tend to have similar functions, which supports the hypothesis that neighboring electrodes should measure similar brain signals. Another example of regularizing CSP is stationary CSP (sCSP), which was proposed by Samek et al. [7]. sCSP assumes that non-stationeries in EEG data come from processes that are not task related, such as eye movements or electrode artifacts. Another study on robust CSP is CSP-L1 by Wang et al. [8], who attempted to express the CSP formulation in L1 norm. He states that the original formulation of CSP was L2-norm, which implies that CSP was sensitive to outliers. Wang attempted to optimize the proposed alternative CSP formulation using an iterative algorithm.

In this study, we propose a general framework called spatial filter network (SFN), which calculates optimal spatial filters using a neural network approach. With SFN, each epoch in a training set is given to the network for learning the optimal spatial filters, unlike CSP, which only uses one average covariance matrix for each class. Additionally, SFN is trained to directly increase the classification accuracy, whereas the purpose of the CSP method is to maximize the given optimization function, which indirectly increases the classification accuracy. In this paper, the CSP algorithm is described first. Then, SFN is introduced along with its network structure and training methods. Finally, both methods are applied to motor imagery data and the results are compared.

The remainder of this paper is organized as follows. In the Materials and Methods section, the standard CSP method and its multi-class version are briefly reviewed. Then, the proposed SFN method and the proposed training methods are described with a simple toy data example. Finally, the motor imagery data set used in this study and the EEG preprocessing routine are described. The Results section summarizes the evaluations and the results of the study. The Discussion section investigates the advantages/disadvantages of the SFN. Finally, the Conclusion section summarizes the study and presents directions for future work.

Materials and Methods

Common spatial patterns

CSP is a widely used technique for obtaining good spatial resolution and discrimination between different classes of motor imagery signals. In general, a motor imagery experiment consists of epochs, in which the user imagines one type of motor imagery task requested on the screen. An epoch can be one of two classes: C_1 and C_2. (i.e., left hand—right hand). Let $X_{C,i} \in \mathbb{R}^{NxT}$ represent an epoch, where C is the class of the epoch, i is the epoch number belonging to class C, N is the number of EEG channels, and T is the number of samples in the epoch. Note that $X_{C,i}$ should be a zero average signal (i.e., band pass filtered). Let $\vec{w} \in \mathbb{R}^{Nx1}$ be a vector in N-dimensional space. A projection of an epoch onto this vector will be

$$\vec{y}_{C,i} = \vec{w}^T X_{C,i} \tag{1}$$

where $\vec{y}_{C,i} \in \mathbb{R}^{1xT}$ denotes the projection of epoch $X_{C,i}$ and T is the transpose operation. The projected signal power $P_{C,i}$ can be written as follows:

$$P_{C,i} = \vec{y}_{C,i}\vec{y}^T_{C,i} = \vec{w}^T X_{C,i} X^T_{C,i}\vec{w} \tag{2}$$

Let $R_{C,i} \in \mathbb{R}^{NxN}$ be the covariance matrix of the band pass-filtered signal $X_{C,i}$ and $\bar{R}_C \in \mathbb{R}^{NxN}$ be the average covariance matrix of class C:

$$R_{C,i} = \frac{X_{C,i}X^T_{C,i}}{tr(X_{C,i}X^T_{C,i})} \qquad \bar{R}_C = \frac{1}{n_C}\sum_{i\in C}^{n_C} R_{C,i} \tag{3}$$

where tr is the trace function and n_C is the number of epochs in C. Let the average power of class C be $\bar{P}_C$. Then, $\bar{P}_C$ is calculated as follows:

$$\bar{P}_C = \frac{1}{n_C}\sum_{i\in C}^{n_C}\vec{w}^T X_{C,i}X^T_{C,i}\vec{w} = \frac{1}{n_C}\sum_{i\in C}^{n_C}\vec{w}^T R_{C,i}\vec{w} = \vec{w}^T\bar{R}_C\vec{w} \tag{4}$$

For the two classes ($C = 1,2$) case, CSP searches for the maximum power ratio of the classes on the projected w axis. Thus, the average power of one class is maximized while that of the other class is minimized. In other words, the spatial filter should maximize the following

Rayleigh quotient problem [7]:

$$\arg\max_{\vec{w}} \frac{\vec{w}^T \bar{R}_1 \vec{w}}{\vec{w}^T \bar{R}_2 \vec{w}} \tag{5}$$

For any $\vec{w}$ that maximizes Eq (5), the denominator can be set to a constant value c by a scalar coefficient without changing the ratio. Thus, maximization of the Rayleigh quotient can be retranslated into a constrained optimization problem:

$$maximize \quad \vec{w}^T \bar{R}_1 \vec{w}, \qquad subject\ to \quad \vec{w}^T \bar{R}_2 \vec{w} = c \tag{6}$$

The above constrained optimization problem can be solved using the Lagrange multiplier method [15]:

$$L(\lambda, \vec{w}) = \vec{w}^T \bar{R}_1 \vec{w} - \lambda(\vec{w}^T \bar{R}_2 \vec{w} - c) \tag{7}$$

$$\frac{\partial L(\lambda, \vec{w})}{\partial \vec{w}} = 2\vec{w}^T \bar{R}_1 - \lambda(2\vec{w}^T \bar{R}_2) = 0 \tag{8}$$

where λ is the Lagrange multiplier. Because $\bar{R}_C$ is a symmetric matrix, the above equation can be written as a standard eigenvalue problem:

$$(\bar{R}_2{}^{-1} \bar{R}_1)\vec{w} = \lambda \vec{w} \tag{9}$$

According to Eq (9), w, which maximizes the Rayleigh quotient, is the eigenvector that corresponds to the largest eigenvalue of $(\bar{R}_2{}^{-1} \bar{R}_1)$.

The CSP spatial filter $W_{CSP} \in \mathbb{R}^{MxN}$ matrix is constructed by taking $M = 2m$ ($M \leq N$) eigenvectors corresponding to the m largest and m smallest eigenvalues:

$$W_{CSP} = [\vec{w}_{\lambda_1}, \ldots \vec{w}_{\lambda_m} \ldots\ldots \vec{w}_{\lambda_{N-m+1}}, \ldots \vec{w}_{\lambda_N}]^T \tag{10}$$

where w_{λ_i} is the eigenvector that corresponds to the eigenvalue λ_i. Any epoch $X_{C,i}$ is spatially filtered by

$$Z_{C,i} = W_{CSP} X_{C,i} \tag{11}$$

where $Z_{C,i} \in \mathbb{R}^{MxT}$ is the spatially filtered signal. Band power (variance) is used as a feature for the classifier. For an epoch i, the CSP feature vector is given by

$$f\vec{csp}_{C,i}^k = log\left(\frac{var(Z_{C,i}^k)}{\sum_{l=1}^{2m} var(Z_{C,i}^l)}\right) k = 1, 2, \ldots M \tag{12}$$

where $f\vec{csp}_{C,i}^k$ is the k^{th} feature of feature vector $f\vec{csp}_{C,i} \in \mathbb{R}^{Mx1}$ that belongs to epoch i and $Z_{C,i}^k$ is the k^{th} row of $Z_{C,i}$. Here, the logarithm of the variance ratio is calculated to approximate the distribution of the features to a normal distribution [9]. Next, the features are used to train a linear classifier.

Extending CSP to multiclass

The optimization function of CSP is defined for two classes. When there are more than two motor imagery classes (e.g., *left hand*, *right hand*, *foot*, *tongue*, etc..), the CSP method requires some modifications. Extending CSP to multiclass is achieved via a combination of classes, either converting a multiclass problem to several binary problems or computing CSP for one class versus all other classes, called *One Versus the Rest (OVR) CSP* [16]. The OVR-CSP method aims to maximize the power of one class versus the total power of the rest of the classes. The OVR-CSP for class c is calculated as follows:

$$\vec{w}_c = \arg\max_{\vec{w}} \frac{\vec{w}^T \bar{R}_c \vec{w}}{\sum_{j\neq c}^{C} \vec{w}^T \bar{R}_j \vec{w}} \qquad (13)$$

where C is the total number of classes and $\vec{w}_c$ is the vector that maximizes the Rayleigh ratio between class c and the other classes. The CSP matrix is constructed by calculating the above equation for all classes. Note that OVR-CSP is a generalization for the CSP method, which is equal to (Eq 5) for the two classes ($C = 2$) case.

Spatial filter network (SFN)

In this section, the proposed spatial filter network (SFN) model is introduced. SFN is depicted in Fig 1. SFN consists of a spatial filter layer (Layer-1) and a classification layer (Layer-2), which are connected to each other with non-linear mapping functions. Layer-1 is formed with a spatial filtering matrix W and feature extraction functions. The input-output relations of the spatial filter layer are given below:

$$y_m(t) = \frac{\sum_{n=1}^{N} W_{nm}\vec{x}(t)}{\sqrt{\sum_{n=1}^{N} W_{nm}^2}} = \frac{\vec{w}_m^T}{\| \vec{w}_m \|}\vec{x}(t) \qquad (14)$$

where $\vec{x}(t) \in \mathbb{R}^{Nx1}$ is the input data at time t of the EEG epoch with N channels and T samples, $0 < t \leq T$. $\vec{w}_m \in \mathbb{R}^{Nx1}$ is the m^{th} column of the spatial filter matrix $W \in \mathbb{R}^{NxM}$, in which each

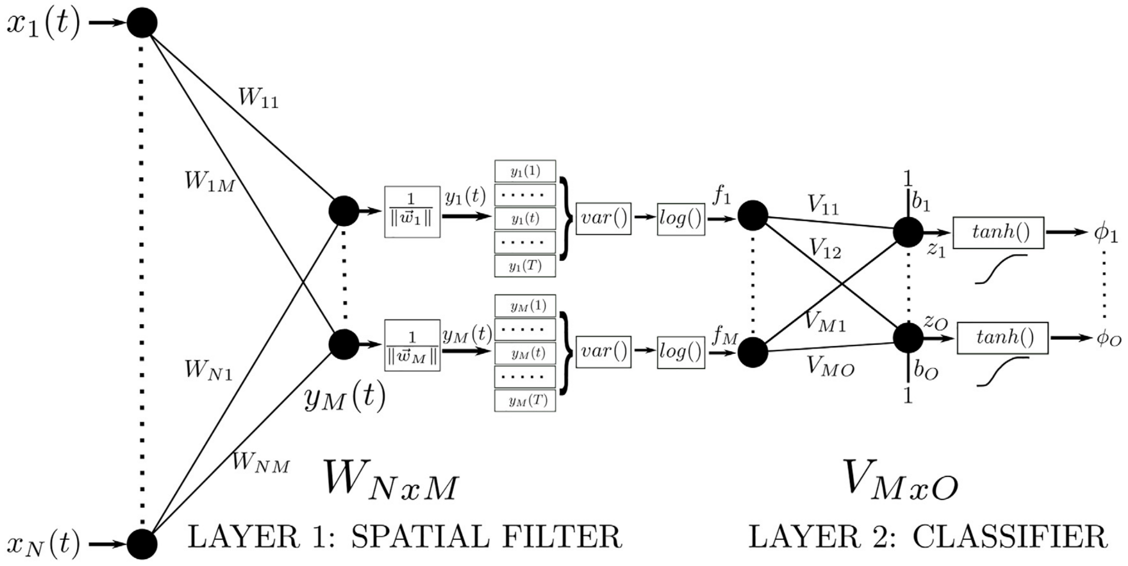

Fig 1. Structure of the proposed spatial filter network (SFN).

doi:10.1371/journal.pone.0125039.g001

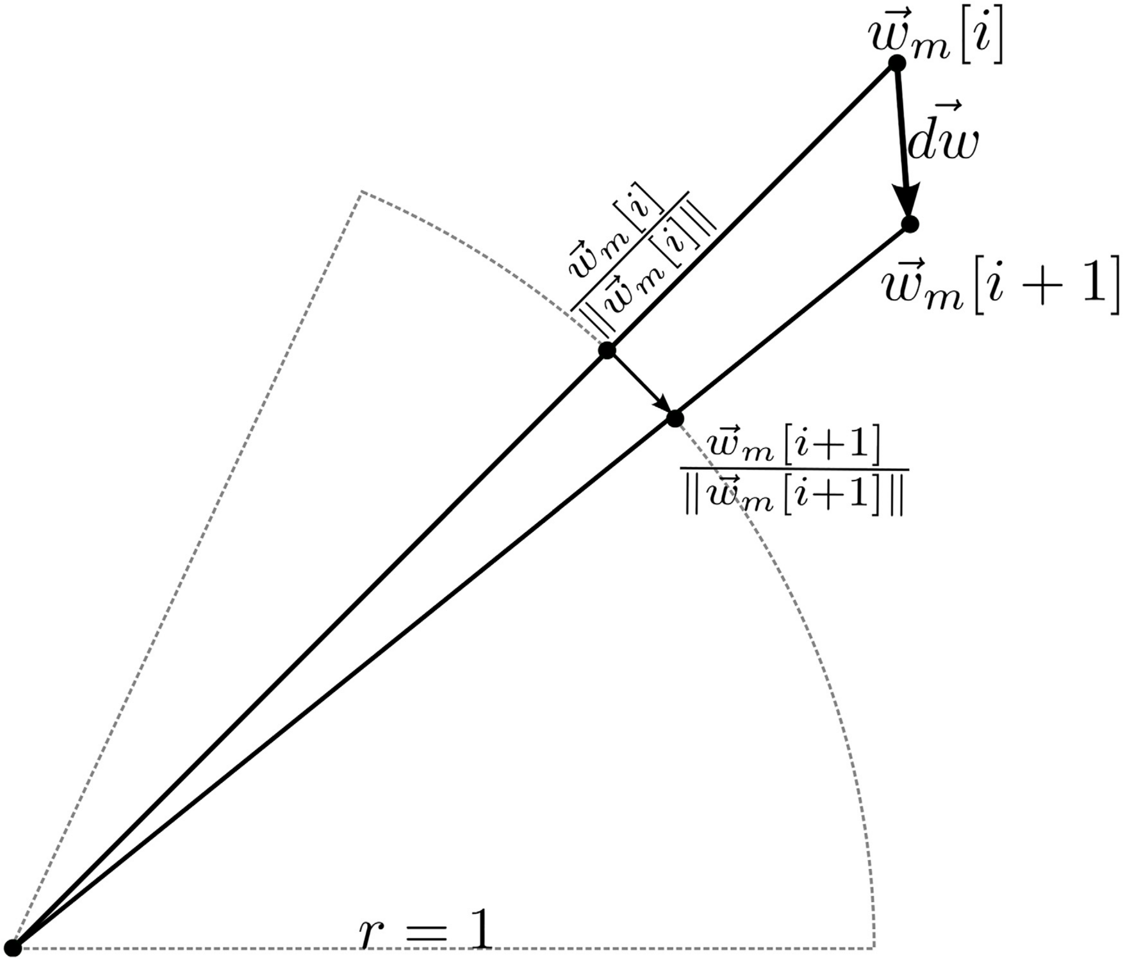
$\vec{w}_m[i]$
$\vec{dw}$
$\frac{\vec{w}_m[i]}{\|\vec{w}_m[i]\|}$
$\vec{w}_m[i+1]$
$\frac{\vec{w}_m[i+1]}{\|\vec{w}_m[i+1]\|}$
$r = 1$

Fig 2. A representative figure of searching the optimal spatial filter over the hypersphere of unit radius.

doi:10.1371/journal.pone.0125039.g002

column is a spatial filter and $y_m(t)$ is the output data at time t for the m^{th} spatial filtering output. $\vec{w}_m$ is divided by its norm to ensure that the input signal is spatially filtered with unit norm filters. Thanks to the $1/\|\vec{w}_m\|$ block, SFN searches for optimal spatial filters over the surface a hypersphere in N dimensions. As shown in Fig 2, the training algorithm, which moves $\vec{w}_m$ by d_w, actually moves the spatial filter over the hypersphere.

Spatial filtering is an important step in motor imagery classification. Redundant data that belong to irrelevant channels are weakened with spatial filtering. Spatial filtering is applied to the input data, and outputs $\vec{y}_m(t)$ are stored until all samples in one epoch are counted. After the spatial filtering phase, Layer-1 calculates the feature vector to be an input for Layer-2.

$$\vec{f_m} = log(var(\vec{y}_m)) = log\left(\frac{1}{T}\sum_{t=1}^{T}(y_m(t) - \mu_{\vec{y}_m})^2\right) \qquad m = 1, 2, ...M \qquad (15)$$

where *var*() is the variance operation and *log*() is the natural logarithm function. Because input data X are zero mean, $\mu_{\vec{y}_m}$ will be zero. $\vec{f} \in \mathbb{R}^{Mx1}$ is the feature vector for a given epoch. Note that the logarithm is used as in CSP to approximate the distribution of the features to a normal distribution [9].

Layer-2 of SFN is the classification layer. An M-dimensional input feature vector (f) is mapped to an O-dimensional output vector z. The input-output relation of Layer-2 is given below:

$$\vec{z} = V^T\vec{f} + \vec{b} \tag{16}$$

$$\vec{\phi} = tanh(\vec{z}) \tag{17}$$

where $V \in \mathbb{R}^{MxO}$ is the weight matrix, $\vec{b} \in \mathbb{R}^{Ox1}$ is the bias vector, $tanh()$ is the tangent hyperbolic function used for clamping $\vec{z}$ to the range $(-1, +1)$, and $\vec{\phi}$ is the network output. Labeling of an input epoch X^k differs for binary and multi-category cases. For the two classes (binary) case, a single output neuron ($O = 1$) with a threshold value is used. Because $tanh$ is used as an activation function, the threshold value will be 0. For a multi-classes case, the number of output neurons should be equal to the number of classes ($O = C$). In this case, the class label of an input epoch X^k is assigned by selecting the output neuron with the highest $\vec{\phi}$ value among all output neurons.

$$Class(X^k) = \arg\max_c \, (\vec{\phi}_c^k) \qquad c = 1, 2, \ldots C \tag{18}$$

Because the class label is generated by selecting the maximum output value, the resulting classifier is called a *linear machine*, in which no ambiguous region exists. A linear machine divides the feature space into C decision regions, with $\vec{\phi}_c$ being the largest discriminant if X^k in region R_c [17].

Training of SFN

We used two methods to train the SFN: Backpropagation [18] and Levenberg-Marquardt. Both methods use partial derivatives to optimize the spatial filter coefficients W_{ij}. For the Backpropagation method, the coefficients are updated after each epoch is presented to SFN, whereas for the Levenberg-Marquardt method, updating is performed after all epochs in the training set are presented to the network. For both methods, the error function is calculated according to the network output and the class label of the epoch presented to the network. Additionally, the initial weights for W, V and $\vec{b}$ are set randomly with a normal distribution $\sigma = 0.1$ and $\mu = 0$.

Error function. SFN requires an error function E to optimize the spatial filter W_{nm} and classifier coefficients V_{mo}. E should be minimum when SFN successfully discriminates different classes in the training set. We used the Euclidean distance between the target class vector $\vec{T}_c$ and SFN output vector $\vec{z}$ as the error measure. The error value of the k^{th} epoch is given by:

$$E^k = \frac{1}{2}\sum_{o=1}^{O}(e_o^k)^2 = \frac{1}{2}\sum_{o=1}^{O}(\vec{\phi}_o^k - \vec{D}_o^k)^2 \quad ,1 \le k \le K \tag{19}$$

where k is the current epoch number presented to the SFN, $\vec{\phi}_o^k$ is the o^{th} output of the SFN, $\vec{D}_o^k$ is the o^{th} element of the target vector when the k^{th} epoch is given to the network, and K is the total number of epochs in the training set. SFN is trained iteratively such that the total error of the network for the training class should be minimized.

Backpropagation method. The Backpropagation (BP) method adapts all weights of a neural network to minimize the error on a set of vectors belonging to a pattern recognition problem [19]. The BP learning rule is based on gradient descent. The weights are initialized with random values and changed in a direction to reduce the error [17]. In this study, the BP method is used to optimize both the spatial filter layer and the classifier layer. In a classical pattern recognition problem, a feature vector is given to the feed-forward neural network and the weights of the network are updated according to the feature's class and the network's output. However, the proposed training method accepts all of the samples in an epoch and updates the weights when an epoch is completely given to the network. For each layer, the learning rule of SFN is given by

$$W_{nm} = W_{nm} - \mu \frac{\partial E}{\partial W_{nm}}$$

$$V_{mo} = V_{mo} - \mu \frac{\partial E}{\partial V_{mo}} \tag{20}$$

$$\vec{b}_o = \vec{b}_o - \mu \frac{\partial E}{\partial \vec{b}_o}$$

where V_{mo} and W_{nm} are the weights of the classifier and spatial filter layers, respectively, b_o is the bias value for the second layer of the network, and μ is the learning rate parameter. The partial derivatives for each layer are calculated using the chain rule:

$$\frac{\partial E}{\partial V_{m,o}} = \frac{\partial E}{\partial \vec{\phi}_o} \frac{\partial \vec{\phi}_o}{\partial \vec{z}_o} \frac{\partial \vec{z}_o}{\partial V_{m,o}} \tag{21}$$

$$\frac{\partial E}{\partial \vec{b}_o} = \frac{\partial E}{\partial \vec{\phi}_o} \frac{\partial \vec{\phi}_o}{\partial \vec{z}_o} \frac{\partial \vec{z}_o}{\partial \vec{b}_o} \tag{22}$$

$$\frac{\partial E}{\partial W_{n,m}} = \sum_{o=1}^{O} \frac{\partial E}{\partial \vec{\phi}_o} \frac{\partial \vec{\phi}_o}{\partial \vec{z}_o} \frac{\partial \vec{z}_o}{\partial \vec{f}_m} \sum_{t=1}^{T} \frac{\partial \vec{f}_m}{\partial \vec{y}_m(t)} \frac{\partial \vec{y}_m(t)}{\partial W_{n,m}} \tag{23}$$

Note that the error of any output propagates to all weights in the spatial filter layer. Because the classifier layer output and error value are calculated after all of the samples in one epoch are counted, backpropagation should be calculated after the epoch is fully presented to the SFN. The required backward equations (derivatives) are given in Eqs (24)–(28):

$$\frac{\partial E}{\partial \vec{\phi}_o} = \vec{\phi}_o - \vec{D} \tag{24}$$

$$\frac{\partial \vec{\phi}_o}{\partial \vec{z}_o} = 1 - (\vec{\phi}_o)^2 \tag{25}$$

$$\frac{\partial \vec{z}_o}{\partial V_{m,o}} = \vec{f}_m \quad \frac{\partial \vec{z}_o}{\partial \vec{f}_m} = V_{m,o} \quad \frac{\partial \vec{z}_o}{\partial \vec{b}_o} = 1 \tag{26}$$

$$\frac{\partial \vec{f}_m}{\partial \vec{y}_m(t)} = \frac{1}{\frac{1}{T}\sum_{t=1}^{T}(\vec{y}_m(t))^2}\frac{2}{T}\vec{y}_m(t) \tag{27}$$

$$\frac{\partial \vec{y}_m(t)}{\partial W_{nm}} = \frac{\|\vec{w}_m\|\vec{x}_n(t) - W_{nm}\vec{y}_m(t)}{\|\vec{w}_m\|^2} \tag{28}$$

With the BP method, the weights (W, V and $\vec{b}$) are updated after each epoch. Training SFN with backpropagation is listed in Algorithm 1. Here, *itr* refers to the iteration number, *MAXITR* is the maximum number of iterations, *EMIN* is the minimum error value to continue iterations, and W^0, V^0 and $\vec{b}^0$ are the initial values for W, V and $\vec{b}$, respectively.

```
Algorithm 1 Pseudo-code for BP algorithm
  $itr \leftarrow 0$, $W \leftarrow W^0$, $V \leftarrow V^0$, $\vec{b} \leftarrow \vec{b}^0$
  while $itr++ < MAXITR$ and $E > EMIN$ do
    Select an epoch from training set
    Calculate network output (Eqs 14–17)
    Calculate error function $E$ (Eq 19)
    Calculate $\partial E/\partial V_{mo}$, $\partial E/\partial \vec{b}_o$ and $\partial E/\partial W_{nm}$ (Eqs 21–23)
    Update network weights (Eq 20)
end while
```

Levenberg–Marquardt method. The Levenberg–Marquardt (LM) algorithm [20, 21] iteratively generates a solution for minimizing a non-linear problem. It is fast and has stable convergence [22]. Unlike the steepest descent method that the back-propagation algorithm uses, the LM method is an approximation to Newtons Method [23]. Let $E(\vec{q})$ be the total error that is iteratively minimized by the proposed method:

$$E(\vec{q}) = \frac{1}{2}\sum_{k=1}^{K}\sum_{o=1}^{O}(e_o^k)^2 \tag{29}$$

where e_o^k is the error at the output o for the epoch number k and $\vec{q} \in \mathbb{R}^{NM+O(M+1)}$ is the vector of all weights forming the SFN:

$$e_o^k = D_o^k - \phi_o^k \tag{30}$$

$$\vec{q} = [W_{11}, W_{12}...W_{NM}, V_{11}, V_{12}, ..., V_{MO}, b_1, b_2, ...b_O] \tag{31}$$

The Levenberg–Marquardt method aims to minimize $E(\vec{q})$ according to the following update rule:

$$\vec{q}_{i+1} = \vec{q}_i - (J_i^T J_i + \mu I)^{-1} J_i \vec{e}_i \tag{32}$$

where i is the iteration number, $\vec{e} \in \mathbb{R}^{KO}$ is a vector that holds the errors of all outputs, $J \in \mathbb{R}^{(NM+O(M+1))x(KO)}$ is the Jacobian matrix, and μ is called the combination coefficient. When μ is very small, the algorithm works as the Gauss-Newton method, but when μ is large, the algorithm turns to the steepest descend [23] method. $\vec{e}$ and J are introduced below:

$$e = [e_1^1, e_2^1, \ldots, e_O^1 \ldots, e_1^K, e_2^K, \ldots, e_O^K]^T \tag{33}$$

$$J = \begin{pmatrix} \frac{\partial e_1^1}{\partial \vec{q}_1} & \frac{\partial e_1^1}{\partial \vec{q}_2} & \cdots & \frac{\partial e_1^1}{\partial \vec{q}_{NM+O(M+1)}} \\ \frac{\partial e_2^1}{\partial \vec{q}_1} & \frac{\partial e_2^1}{\partial \vec{q}_2} & \cdots & \frac{\partial e_2^1}{\partial \vec{q}_{NM+O(M+1)}} \\ \cdots & \cdots & \cdots & \cdots \\ \frac{\partial e_O^1}{\partial \vec{q}_1} & \frac{\partial e_O^1}{\partial \vec{q}_2} & \cdots & \frac{\partial e_O^1}{\partial \vec{q}_{NM+O(M+1)}} \\ \cdots & \cdots & \cdots & \cdots \\ \frac{\partial e_1^K}{\partial \vec{q}_1} & \frac{\partial e_1^K}{\partial \vec{q}_2} & \cdots & \frac{\partial e_1^K}{\partial \vec{q}_{NM+O(M+1)}} \\ \frac{\partial e_2^K}{\partial \vec{q}_1} & \frac{\partial e_2^K}{\partial \vec{q}_2} & \cdots & \frac{\partial e_2^K}{\partial \vec{q}_{NM+O(M+1)}} \\ \cdots & \cdots & \cdots & \cdots \\ \frac{\partial e_O^K}{\partial \vec{q}_1} & \frac{\partial e_O^K}{\partial \vec{q}_2} & \cdots & \frac{\partial e_O^K}{\partial \vec{q}_{NM+O(M+1)}} \end{pmatrix} \tag{34}$$

When calculating the Jacobian matrix, the chain rule is applied to $\partial e/\partial \vec{q}$, and the following equations are obtained:

$$\frac{\partial e_o}{\partial V_{mo}} = \frac{\partial e_o}{\partial \vec{\phi}_o} \frac{\partial \vec{\phi}_o}{\partial \vec{z}_o} \frac{\partial \vec{z}_o}{\partial V_{mo}} \tag{35}$$

$$\frac{\partial e_o}{\partial \vec{b}_o} = \frac{\partial e_o}{\partial \vec{\phi}_o} \frac{\partial \vec{\phi}_o}{\partial \vec{z}_o} \frac{\partial \vec{z}_o}{\partial \vec{b}_o} \tag{36}$$

$$\frac{\partial e_o}{\partial W_{nm}} = \frac{\partial e_o}{\partial \vec{\phi}_o} \frac{\partial \vec{\phi}_o}{\partial \vec{z}_o} \frac{\partial \vec{z}_o}{\partial \vec{f}_m} \sum_{t=1}^{T} \frac{\partial \vec{f}_m}{\partial \vec{y}_m(t)} \frac{\partial \vec{y}_m(t)}{\partial W_{nm}} \tag{37}$$

Note that from (Eq 30), $\partial e_o/\partial \phi_o$ equals -1. Other terms may be calculated using Eqs (24)–(28). At the end of each epoch, the network outputs and error values are calculated, and then the Jacobian matrix is constructed. The LM method updates the network weights after all of the epochs in the training set are presented to the SFN. μ is increased or decreased with each iteration such that the convergence rate is adjusted. Additionally, if the total error of the network increases with new weights after updating, the LM algorithm sets the weights to their previous values and slows the convergence rate. Training of SFN with the LM method is demonstrated using the Algorithm 2. Here, μ_0 is the initial value for the combination coefficient μ and β is the multiplier (divider) of μ for increasing (decreasing) it. For other terms, please refer to the description of the BP Algorithm 1. Note that, depending on the convergence of the network, the algorithm slightly changes to the steepest-descend or Gauss-Newton method by adjusting the value of μ.

```
Algorithm 2 Pseudo-code for LM algorithm
  $itr \leftarrow 0$, $\mu \leftarrow \mu_0$, $W \leftarrow W^0$, $V \leftarrow V^0$, $\vec{b} \leftarrow \vec{b}^0$
  while $itr++ < MAXITR$ and $E > EMIN$ do
    for all epoch in Training Set do
      Calculate network output (Eqs 14–17)
      Calculate error for all outputs $e_o^k$ (Eq 30)
      Calculate Jacobian matrix rows for the current epoch (Eq 34)
    end for
    Calculate total error (Eq 29)
    if $E_{itr} < E_{itr-1}$ then
      $\mu \leftarrow \mu / \beta$
    else
      $\mu \leftarrow \mu * \beta$
      Revert to previous weights $q_{itr} \leftarrow q_{itr-1}$
    end if
    Calculate new weights vector $q_{itr}$ (Eq 32)
    Update network weights $W$, $V$ and $\vec{b}$ (Eq 31)
  end while
```

Running SFN with toy data

In this section, we test the SFN using generated toy data for 2 and 4 classes. Each class in the generated data has a different covariance matrix. For visualization purposes, the dimensions of the toy data and the size of the output neuron of the spatial filter layer were set to 2 ($N = 2$, $M = 2$). Note that the class label of an epoch in the toy data was randomly selected. Therefore, the number of epochs for each class will be approximately identical. Additionally, epoch data were generated with zero mean, fixed covariance matrices specific to each class using the Matlab command *mvnrnd* [24]. For both data sets, the epoch dimension (NxT) was set to $2x100$ and the total number of epochs (K) was set to 100. SFN was trained with the BP and LM algorithms. The network parameters are as follows: $\mu = 10^{-3}$ for the BP method and $\mu_0 = 100$, $\beta = 2$ for the LM method. Both methods successfully converged to the desired error value ($EMIN = 10^{-3}$). As expected, the LM algorithm converged faster than the BP algorithm. The convergence of the two methods for the 2-classes case is shown in Fig 3.

Fig 4 presents the input data and the SFN output data for the 2-classes case. In this Figure, (a) illustrates the log-variance feature of the input data:

$$\begin{aligned} f_1(X^k) &= log(var([X_{11}^k, X_{12}^k, \dots X_{1T}^k])) \\ f_2(X^k) &= log(var([X_{21}^k, X_{22}^k, \dots X_{2T}^k])) \end{aligned} \tag{38}$$

where X^k is the k^{th} epoch. Each red (circle) and blue (plus) point represent the epoch with class 1 or 2, respectively. In (b), the input data calculated by principle components are enclosed with ellipses. Note that the input data have the same variance in each dimension. (c) Shows the effect of the SFN spatial filter layer, i.e., scattering of spatially filtered data (f_m^k). The spatial filter layer successfully separates the two classes. Here, the black dashed line shows the between-class border created by SFN classifier layer. In (d), the effect of SFN spatial filtering is shown with enclosing ellipses. Note that spatial filtering manipulates the data, in which the features that belong to any class have maximum variance in one dimension, and they have minimum variance in the other dimension.

The SFN output for the 4-classes toy data is shown in Fig 5. In (a), the log-variance features of the 4 classes are shown. (b) Illustrates the enclosing ellipses for each class. The spatial filter output is given in (c). As for the 2-classes case, the spatial filter layer of SFN has two outputs

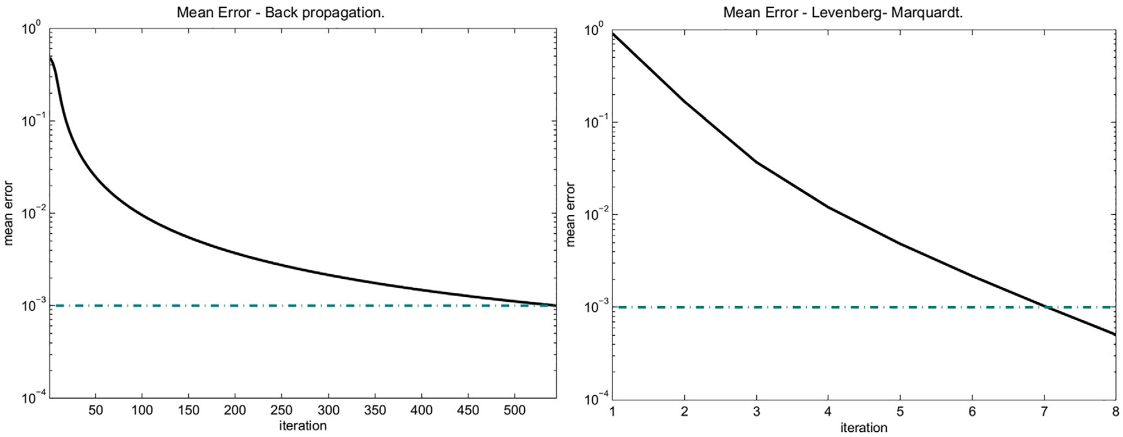

Fig 3. Convergence of Backpropagation and Levenberg–Marquardt algorithms to the desired error (*EMIN*, dashed line) for 2 classes toy data example.

doi:10.1371/journal.pone.0125039.g003

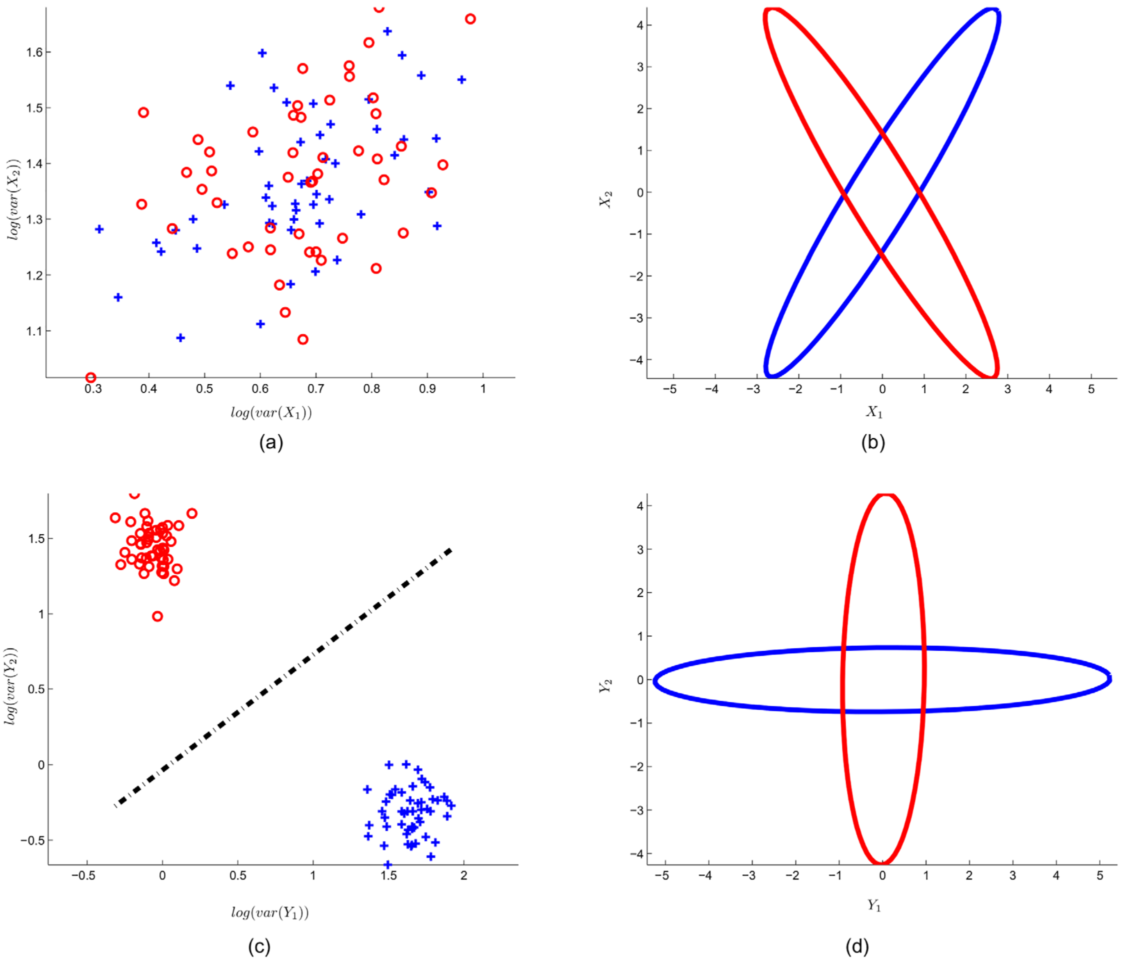

Fig 4. Input data and SFN output data for toy data with 2 classes. (a) log-variance feature for 2-dimensional input data. Note that each point represents an epoch that belongs to class 1 (red circle) or class 2 (blue plus). (b) Enclosing ellipses represent the input data. (c) SFN spatial filter layer output (*f*) with generated class border (black dashed line) of the classifier layer. (d) Enclosing ellipses represent the spatially filtered input data (*y*).

doi:10.1371/journal.pone.0125039.g004

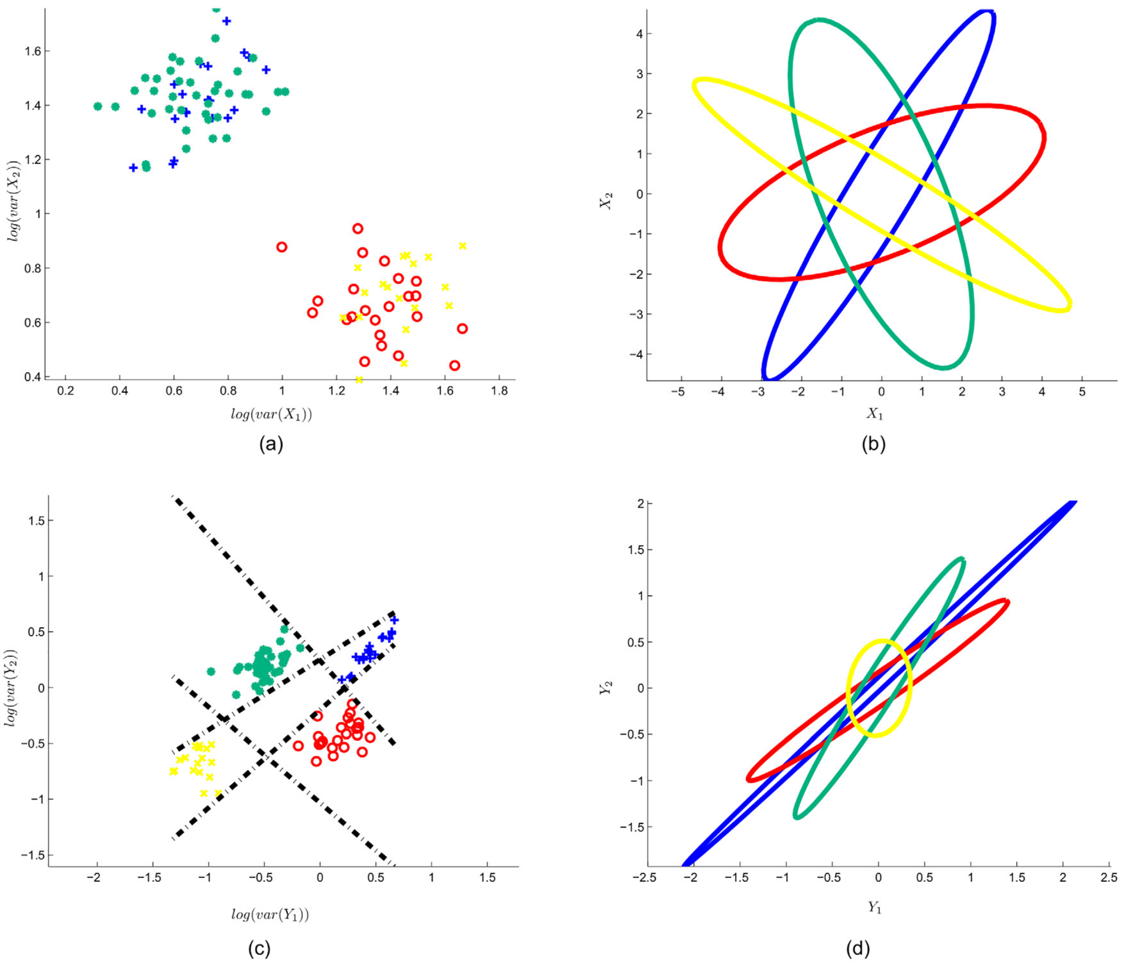

Fig 5. Input data and SFN output data for toy data with 4-classes. (a) log-variance feature for 2-dimensional input data. Note that each point represents an epoch that belongs to class 1 (red circle), class 2 (blue plus), class 3 (green asterisk) or class 4 (yellow cross). (b) Enclosing ellipses represent the input data. (c) SFN spatial filter layer output (*f*) with generated class borders (black dashed lines) of the classifier layer. (d) Enclosing ellipses represent the spatially filtered input data (*y*).

doi:10.1371/journal.pone.0125039.g005

($M = 2$) such that we are able to visualize the output data. However, a larger output dimension may be used for data with more complex scattering. Because the SFN classifier has 4 outputs ($O = 4$), there are 4 class borders, which are drawn in (c). Note that, as a result of the selected target vectors in the one-versus-rest style, each borderline separates one class from the other classes. As shown in (d), the spatial filter layer manipulated the input data such that each class will have a different variance vector.

EEG data sets

In this study, we used two publicly available EEG motor imagery data sets: BCI competition III Data Set IVa and BCI competition III Data Set IIIa [11]. Data Set IVa consists of EEG recordings of 5 subjects who performed motor imagery of the right hand and foot. A total of 118 electrodes were used for recording EEGs with a sample rate of 100 Hz. There are 280 trials for each subject. However, the numbers of training and test sets differ for each subject: 168, 224, 84, 56 and 28 trials are the sizes of the training sets for subjects labeled as *aa*, *al*, *av*, *aw* and *ay*, respectively, and the remaining trials form the test set. Note that no validation set was used for network training, and we applied the test set after SFN approached the target error value with the training set in the training phase. However, we limited the iteration number and minimum error value to avoid over-fitting.

Data Set IIIa is a four classes data set with a 60-channel EEG signal sampled at 250 Hz and recorded from 3 subjects. The class labels are *left hand*, *right hand*, *foot* and *tongue*. For each subject, there are minimum 60 trials per class; however, some of the trials are marked with *rejected trial*. Detailed information about the data sets may be found on the web site of the third BCI competition [25] and in the related paper of Blankertz et al. [11].

Preprocessing, network configuration

We applied the same preprocessing steps for both data sets: i) We used EEG electrodes that roughly cover the motor cortex. Selected electrodes for the two data sets are shown in Fig 6. The numbers of electrodes are 23 and 29 for the the two classes data set (BCIC III-IVA) and for the four classes data set (BCIC III-IIIA), respectively. ii) The EEG signal is band bass filtered with a 8–30 Hz 5th-order Butterworth filter. iii) For each trial, we used EEG signals in the time segment between 0.5s–2s after the instruction cue. Furthermore, trials marked with *rejected trial* were excluded. The preprocessing phase is illustrated in Fig 7.

We used one-versus-rest CSP for comparison with SFN. As a classifier to classify the output of OVR-CSP, we selected linear discriminant analysis (LDA), which is a popular classifier in MI classification. OVR-CSP may be configured via setting the number of spatial filters for each class (m). In this study, we tested OVR-CSP with various values for m along with various spatial filter layer matrix widths (M) for SFN. We trained SFN with the LM method because of its convergence speed. SFN was configured as follows: $\mu_0 = 100$, $\beta = 2$, $EMIN = 10^{-1}$ and $ITRMAX = 1000$.

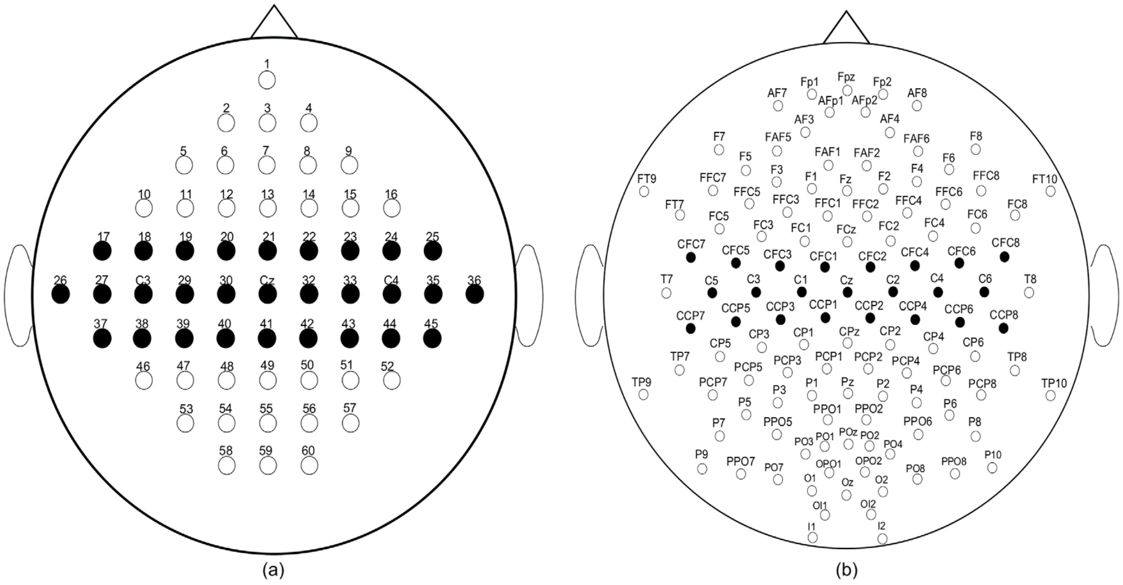
1
2 3 4
5 6 7 8 9
10 11 12 13 14 15 16
17 18 19 20 21 22 23 24 25
26 27 C3 29 30 Cz 32 33 C4 35 36
37 38 39 40 41 42 43 44 45
46 47 48 49 50 51 52
53 54 55 56 57
58 59 60
(a)
Fp1 Fpz Fp2
AF7 AFp1 AFp2 AF8
AF3 AF4
F7 FAF5 FAF6 F8
F5 FAF1 FAF2 F6
F3 F4
FT9 FFC7 F1 Fz F2 FFC8 FT10
FFC5 FFC6
FT7 FFC3 FFC1 FFC2 FFC4 FC8
FC5 FC6
FC3 FC1 FCz FC2 FC4
CFC7 CFC5 CFC3 CFC1 CFC2 CFC4 CFC6 CFC8
T7 C5 C3 C1 Cz C2 C4 C6 T8
CCP7 CCP5 CCP3 CCP1 CCP2 CCP4 CCP6 CCP8
CP5 CP3 CP1 CPz CP2 CP4 CP6
TP7 PCP5 PCP3 PCP1 PCP2 PCP4 PCP6 TP8
TP9 PCP7 P1 Pz P2 P4 PCP8 TP10
P3
P5 P6
PPO1 PPO2
P7 PPO5 PPO6 P8
PO3 PO1 POz PO2 PO4
P9 PPO7 PO7 OPO1 OPO2 PO8 PPO8 P10
O1 Oz O2
OI1 OI2
I1 I2
(b)

Fig 6. Selected electrodes (black circles) for BCI competition III Data Set IIIa (a) and BCI competition III Data Set IVa (b).

doi:10.1371/journal.pone.0125039.g006

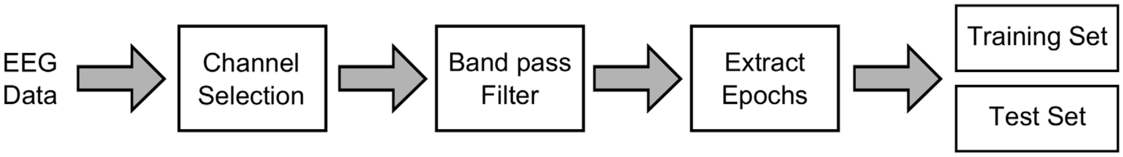

Fig 7. Preprocessing phase for EEG data sets.

doi:10.1371/journal.pone.0125039.g007

Results

Figs 8 and 9 show the SFN accuracies for subjects of BCIC-III-IVA and BCIC-III-IIIA, respectively. For any subject and *m* value, the accuracy of OVR-CSP is constant for every running of the algorithm because OVR-CSP does not accept any hyper parameter and because it does not need an initial weight setting. However, because the initial weights were set randomly, the obtained accuracies vary at each running of the SFN. Therefore, SFN accuracies were plotted with box plots, where the box boundaries represent the upper and lower 25% quantiles of the input data (outliers were excluded), which was obtained by running the proposed algorithm 30 times for each user and *m* value. The bold lines inside the boxes represent the median values. Note that the total number of spatial filters (*M*) for OVR-CSP is a multiple of the number of classes. Therefore, the OVR-CSP method was run with *M* = [2, 4, 6, 8, 10] for the 2 classes data set BCIC-III-IVA and with *M* = [4, 8, 12, 16, 20] for the 4 classes data set BCIC-III-IIIA. The OVR-CSP accuracies for those *M* values are indicated with black dots in figures.

Although the classification accuracies vary significantly across subjects, it is clear that SFN increases the classification accuracy for most of the subjects in both data sets. For example, for subject *k6b*, whose classification results are the poorest in the 4 classes data set, SFN increases

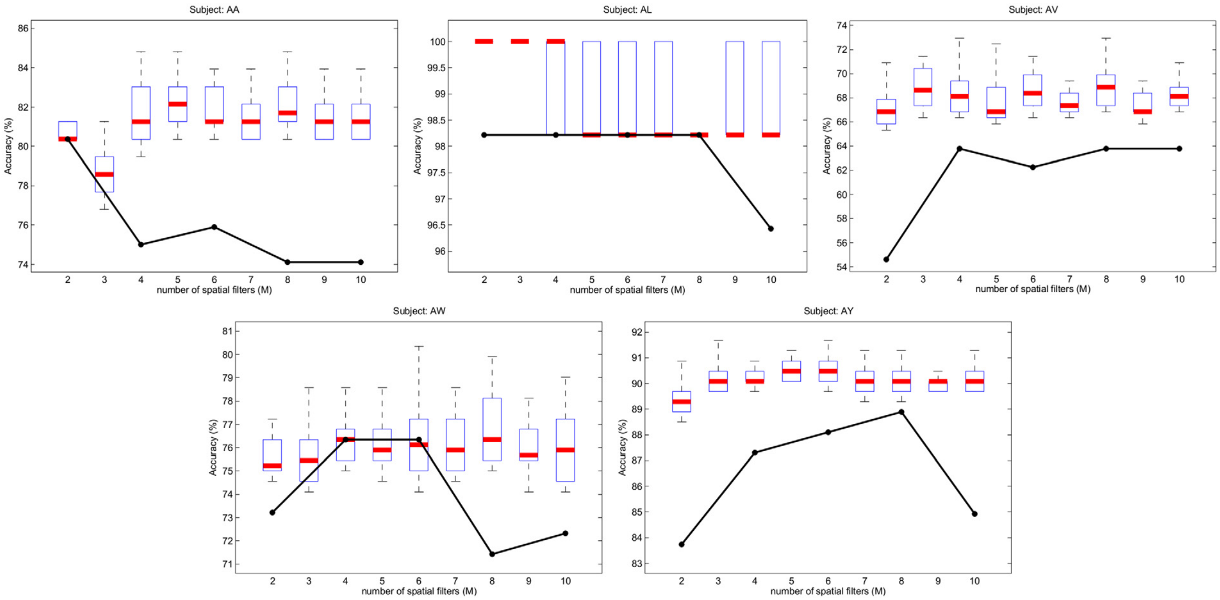

Fig 8. Accuracy of SFN and OVR-CSP over the subjects of data set BCIC-III-IVA.

doi:10.1371/journal.pone.0125039.g008

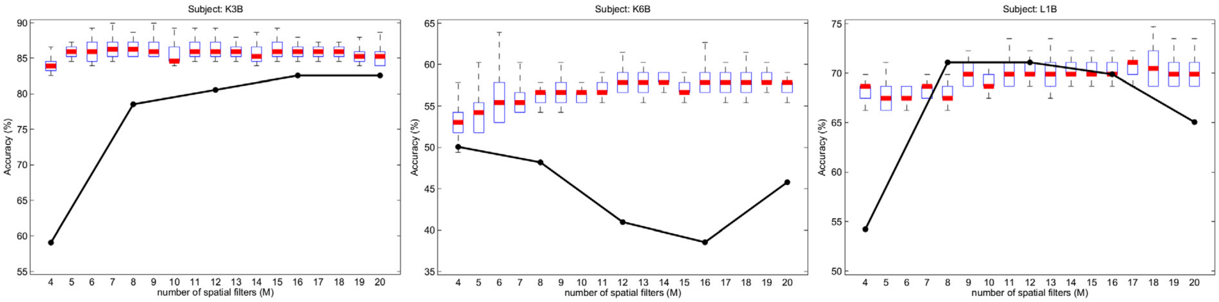

Fig 9. Accuracy of SFN and OVR-CSP over the subjects of data set BCIC-III-IIIA.

doi:10.1371/journal.pone.0125039.g009

the accuracy by approximately 12%. Even though the proposed method may fall behind the multi-class CSP, it may achieve better accuracies in some trials (see the vertical lines above each box). Because the initial weights of SFN are set randomly, the accuracy for any subject and *M* value vary across each running of the algorithm. This situation may be improved with better and more robust selection of the initial network weights, which provides constantly higher results. However, increasing the robustness of the initial weight selection is beyond the scope of this paper.

Discussion

Spatial Filters

The spatial filter layer of SFN aims to maximize the separability of the obtained features that belong to different classes. Thus, we expect each column of *W*, which is the weight matrix of SFN layer 1, to lighten a special area over the coverage of electrodes. We illustrate the spatial filters of the CSP and SFN methods in Fig 10 for 2 and 4 classes. Here, the upper and lower rows display the spatial filters of the SFN and CSP methods, respectively. Gray tones describe the absolute value of the illustrated spatial filter because the sign of the spatial filter weights does not affect the obtained energy features. Because the target functions of CSP and SFN are not identical, there are some differences between the the spatial filters obtained with these methods. However, both methods provide physiologically satisfying results. For example, right-hand motor imaginary is localized in the left hemisphere of the sensory-motor-cortex area, which is suitable with the Homunculus figure given in [26].

While CSP works with the average covariance matrices to calculate the spatial filters, SFN optimizes its spatial filters and classifier by handling each epoch in the training set. Therefore, spatial filters obtained with SFN provide *better* spatial filters that increase the between-class variance while decreasing within-class variance, thereby increasing the ratio of between-class variance and within-class variance, which is called the Fischer discriminant criterion or separation [27].

Fig 11 shows an example about separation of classes by using CSP and SFN spatial filters. Separation values for CSP and SFN features are given on the figure. As expected, SFN provides much higher separation value with proper organization of features (i.e. features belong to different classes separated in feature space) Class border given for CSP method was found by using LDA classifier while it was directly calculated for SFN algorithm.

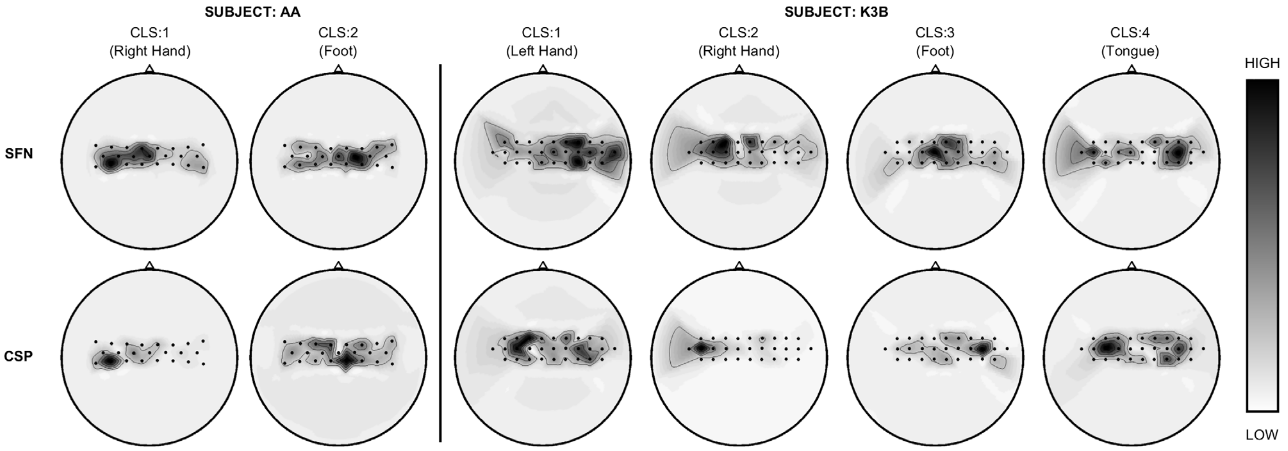

Fig 10. Illustration of spatial filters obtained with the SFN and CSP methods. Head figures are displayed for subject AA of data set BCIC-III-IVA (left) and subject K3B of dataset BCIC-III-IIIA. Each column corresponds to a class of the given data set.

doi:10.1371/journal.pone.0125039.g010

Convergence of network

We analyzed the convergence behavior of the network. As previously stated, the Levenberg-Marquardt (LM) method is used for training. Because the LM method has superior convergence properties, as proven in previous papers [28, 29], we expect SFN to converge to a desired error value. Fig 12 displays the mean error at each iteration for multiple training experiments with subjects AA and K3B. SFN reaches the desired error at each experiment for both of the subjects. However, the average number of iterations until convergence is higher for subject K3B of the data set BCIC-III-IIIA, which is a 4 classes data set.

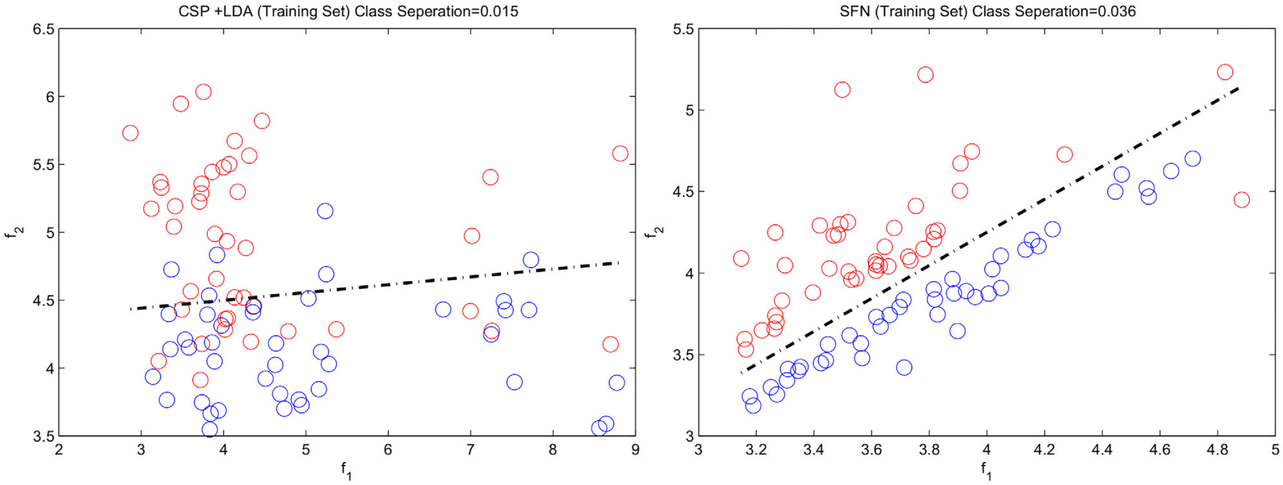

Fig 11. Features extracted with CSP (left) and SFN (right) from the training set of subject *av*. Blue: class 1, red: class 2. Dashed lines represents the class borders.

doi:10.1371/journal.pone.0125039.g011

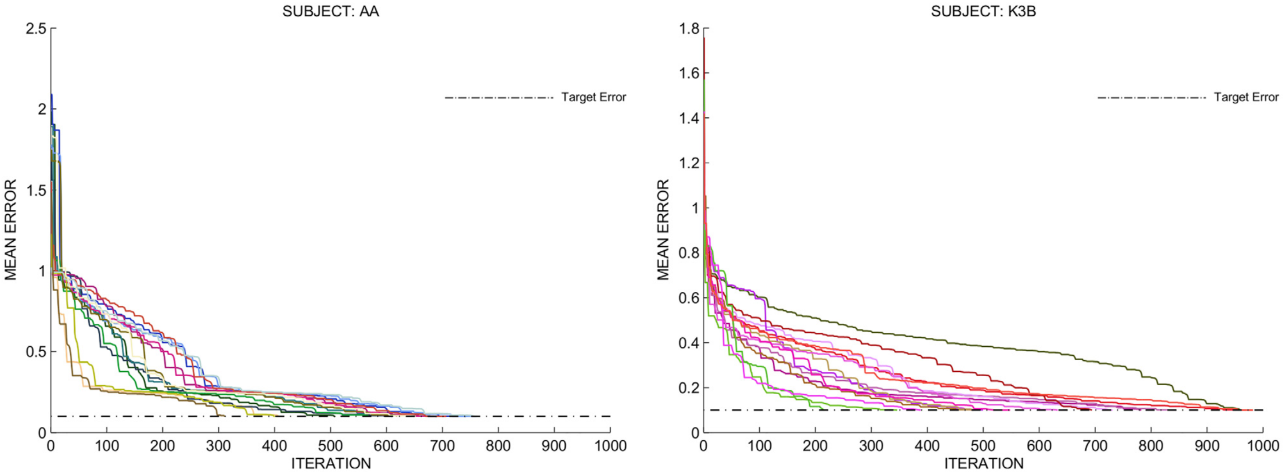

Fig 12. Converge behavior of the SFN. Network had been trained 15 times for subject AA (on the left) and for subject K3B (on the right). Resulting mean error value at each iteration until convergence for each training experiment is displayed in different colors.

doi:10.1371/journal.pone.0125039.g012

Training time

Because SFN learns the training set iteratively, the training time is longer than that for any direct method, such as CSP. Fig 13 plots the average training times for subject *aa* in the BCI-C-III-IVA data set versus the number of spatial filters (*M*). This figure also plots the training time across each subject, where each subject has a different number of training sets. As shown in this figure, the required time for training the network increases linearly with the size of the SFN weight matrices and the number of epochs in the training set. The algorithm was run on a single-core Intel® Xeon® CPU operating at 2 GHz.

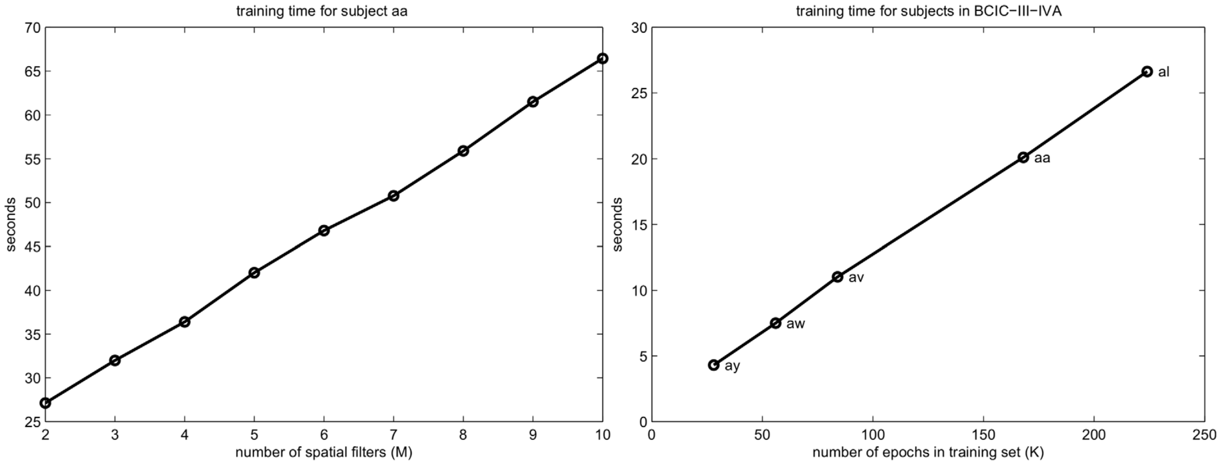

Fig 13. Training time of SFN. Left: number of spatial filters (*M*) versus training time for subject *aa*. Right: number of epochs versus training time for the subjects of BCIC-III-IVA when *M* = 2. SFN was configured with *EMIN* = 0, *ITRMAX* = 1000, *μ* = 100 and *B* = 2.

doi:10.1371/journal.pone.0125039.g013

Although the training of SFN takes a long time, once the network is trained, the classification time for an epoch is very small. Therefore, a longer training time is not an obstacle for such a BCI system because training is performed off-line most of the time.

Conclusions

In this study, we presented a combined method called spatial filter network (SFN) that combines spatial filter optimization and classification for motor imagery-based BCIs. The results indicated that the SFN method is an alternative approach to CSP with increased classification performance on two and four classes data sets and that it has two main advantages: ability to handle each single epoch in the training set and combining of spatial filter and classifier in a single structure. The originality of this paper is the combined optimization of spatial filters and classifier for single trial EEG motor imagery experiments. To the best of our knowledge, there are no methodologies in the literature that combine a spatial filter and classifier for such signals.

The following further work can be considered in future studies: searching for better initial weights to obtain consistently higher accuracies, studying more robust training methods that are insensitive to the outliers, and modifying the training algorithm for more efficient and faster convergence. Additionally, we could study the effect of increasing the number of spatial filter layers that are connected to each other with non-linear functions. To the best of our knowledge, multi-layer spatial filters have not previously been studied, and there may be some interesting results, particularly for dealing with non-stationary signals.

Supporting Information

S1 Video. An animation captured while training SFN for subject *av*. Number of spatial filters was set to $M = 2$ for display purposes. Left: Training set features ($\vec{f} \in \mathbb{R}^{2x1}$) scattered on two-dimensional feature space, where each circle represents an epoch that belongs to either class 1 (blue) or class 2 (red). Class separation value is displayed on the top of the window. Class border found by classifier layer is plotted with a black dashed line. Note that the two axes were normalized to the range [0–1] for display purposes. Right: Mean error of the current iteration was plotted with target error value (green horizontal line). Error value is displayed on the top of the window.
(MP4)

S1 Code. Matlab code for SFN training and classification routines. See README and specific help texts of the supplied files.
(M)

Author Contributions

Conceived and designed the experiments: AY TO. Performed the experiments: AY. Analyzed the data: AY. Contributed reagents/materials/analysis tools: AY TO. Wrote the paper: AY.

References

1. Wolpaw JR, Birbaumer N, McFarland DJ, Pfurtscheller G, Vaughan TM. Brain–computer interfaces for communication and control. Clinical neurophysiology. 2002; 113(6):767–791. doi: 10.1016/S1388-2457(02)00057-3 PMID: 12048038

2. Pfurtscheller G, Lopes da Silva F. Event-related EEG/MEG synchronization and desynchronization: basic principles. Clinical neurophysiology. 1999; 110(11):1842–1857. doi: 10.1016/S1388-2457(99)00141-8 PMID: 10576479

3. Barachant A, Bonnet S, Congedo M, Jutten C. Classification of covariance matrices using a Riemannian-based kernel for BCI applications. Neurocomputing. 2013; 112:172–178. doi: 10.1016/j.neucom.2012.12.039

4. Blankertz B, Tomioka R, Lemm S, Kawanabe M, Muller KR. Optimizing spatial filters for robust EEG single-trial analysis. Signal Processing Magazine, IEEE. 2008; 25(1):41–56. doi: 10.1109/MSP.2008.4408441

5. Blankertz B, Kawanabe M, Tomioka R, Hohlefeld F, Müller Kr, Nikulin VV. Invariant common spatial patterns: Alleviating nonstationarities in brain-computer interfacing. In: Advances in neural information processing systems; 2007. p. 113–120.

6. Ramoser H, Muller-Gerking J, Pfurtscheller G. Optimal spatial filtering of single trial EEG during imagined hand movement. Rehabilitation Engineering, IEEE Transactions on. 2000; 8(4):441–446. doi: 10.1109/86.895946

7. Samek W, Vidaurre C, Müller KR, Kawanabe M. Stationary common spatial patterns for brain–computer interfacing. Journal of Neural Engineering. 2012; 9(2):026013. doi: 10.1088/1741-2560/9/2/026013 PMID: 22350439

8. Wang H, Tang Q, Zheng W. L1-norm-based common spatial patterns. Biomedical Engineering, IEEE Transactions on. 2012; 59(3):653–662. doi: 10.1109/TBME.2011.2177523

9. Falzon O, Camilleri KP, Muscat J. The analytic common spatial patterns method for EEG-based BCI data. Journal of Neural Engineering. 2012; 9(4):045009. doi: 10.1088/1741-2560/9/4/045009 PMID: 22832090

10. Blankertz B, Muller KR, Curio G, Vaughan TM, Schalk G, Wolpaw JR, et al. The BCI competition 2003: progress and perspectives in detection and discrimination of EEG single trials. Biomedical Engineering, IEEE Transactions on. 2004; 51(6):1044–1051. doi: 10.1109/TBME.2004.826692

11. Blankertz B, Muller KR, Krusienski DJ, Schalk G, Wolpaw JR, Schlogl A, et al. The BCI competition III: Validating alternative approaches to actual BCI problems. Neural Systems and Rehabilitation Engineering, IEEE Transactions on. 2006; 14(2):153–159. doi: 10.1109/TNSRE.2006.875642

12. Fattahi D, Nasihatkon B, Boostani R. A general framework to estimate spatial and spatio-spectral filters for EEG signal classification. Neurocomputing. 2013; 119:165–174. doi: 10.1016/j.neucom.2013.03.044

13. Reuderink B, Poel M. Robustness of the common spatial patterns algorithm in the BCI-pipeline; 2008. TR-CTIT-08-52.

14. Lotte F, Guan C. Regularizing common spatial patterns to improve BCI designs: unified theory and new algorithms. Biomedical Engineering, IEEE Transactions on. 2011; 58(2):355–362. doi: 10.1109/TBME.2010.2082539

15. Bertsekas DP. Constrained optimization and Lagrange multiplier methods. Computer Science and Applied Mathematics, Boston: Academic Press, 1982 1982;1.

16. Dornhege G, Blankertz B, Curio G, Muller K. Boosting bit rates in noninvasive EEG single-trial classifications by feature combination and multiclass paradigms. Biomedical Engineering, IEEE Transactions on. 2004; 51(6):993–1002. doi: 10.1109/TBME.2004.827088

17. Duda RO, Hart PE, Stork DG. Pattern classification. John Wiley & Sons; 2012.

18. Rumelhart DE, Hintont GE, Williams RJ. Learning representations by back-propagating errors. Nature. 1986; 323(6088):533–536. doi: 10.1038/323533a0

19. Cortes C, Vapnik V. Support-vector networks. Machine learning. 1995; 20(3):273–297. doi: 10.1023/A:1022627411411

20. Levenberg K. A method for the solution of certain problems in least squares. SIAM J Appl Math. 1944; 11(2):431–441.

21. Marquardt DW. An algorithm for least-squares estimation of nonlinear parameters. Journal of the Society for Industrial & Applied Mathematics. 1963; 11(2):431–441. doi: 10.1137/0111030

22. Yu H, Wilamowski BM. Levenberg-marquardt training. The Industrial Electronics Handbook. 2011; 5:1–15.

23. Hagan MT, Menhaj MB. Training feedforward networks with the Marquardt algorithm. Neural Networks, IEEE Transactions on. 1994; 5(6):989–993. doi: 10.1109/72.329697

24. MATLAB. version 8.1.0 (R2013a). Natick, Massachusetts: The MathWorks Inc.; 2013.

25. Blankertz B. BCI Competition III @ONLINE; 2014. Available from: http://www.bbci.de/competition/iii/.

26. Penfield W, Boldrey E. Somatic motor and sensory representation in the cerebral cortex of man as studied by electrical stimulation. Brain: A journal of neurology. 1937;. doi: 10.1093/brain/60.4.389

27. Fisher RA. The use of multiple measurements in taxonomic problems. Annals of eugenics. 1936; 7 (2):179–188. doi: 10.1111/j.1469-1809.1936.tb02137.x

28. Yamashita N, Fukushima M. On the rate of convergence of the Levenberg-Marquardt method. In: Topics in numerical analysis. Springer; 2001. p. 239–249.

29. Moré JJ. The Levenberg-Marquardt algorithm: implementation and theory. In: Numerical analysis. Springer; 1978. p. 105–116.