Pattern Recognition 145 (2024) 109924

Contents lists available at ScienceDirect

Pattern Recognition

journal homepage: www.elsevier.com/locate/pr

Supervised penalty-based aggregation applied to motor-imagery based brain-computer-interface

J. Fumanal-Idocin [a,*], C. Vidaurre [a], J. Fernandez [a], M. Gómez [a], J. Andreu-Perez [b,c], M. Prasad [d], H. Bustince [a]

[a] Public University of Navarra and Institute of Smart Cities, Campus Arrosadia s/n, 31006 Pamplona, Spain
[b] School of Computer Science and Electronic Engineering, University of Essex, Smart Health Technologies Group, United Kingdom
[c] Sinbad², University of Jaén, Campus Las Lagunillas s/n, 23071, Jaén, Spain
[d] School of Computer Science, FEIT, University of Technology Sydney, NSW, Australia

ARTICLE INFO

Keywords:
Brain–computer interface
Motor imagery
Penalty function
Aggregation functions
Classification
Signal processing

ABSTRACT

In this paper we propose a new version of penalty-based aggregation functions, the Multi Cost Aggregation choosing functions (MCAs), in which the function to minimize is constructed using a convex combination of two relaxed versions of restricted equivalence and dissimilarity functions instead of a penalty function. We additionally suggest two different alternatives to train a MCA in a supervised classification task in order to adapt the aggregation to each vector of inputs. We apply the proposed MCA in a Motor Imagery-based Brain–Computer Interface (MI-BCI) system to improve its decision making phase. We also evaluate the classical aggregation with our new aggregation procedure in two publicly available datasets. We obtain an accuracy of 82.31% for a left vs. right hand in the Clinical BCI challenge (CBCIC) dataset, and a performance of 62.43% for the four-class case in the BCI Competition IV 2a dataset compared to a 82.15% and 60.56% using the arithmetic mean. Finally, we have also tested the goodness of our proposal against other MI-BCI systems, obtaining better results than those using other decision making schemes and Deep Learning on the same datasets.

1. Introduction

Brain–Computer Interfaces (BCIs) provide new means of communication between the human brain and the devices or systems to be controlled by changes in brain dynamics [1]. There are several types of BCI systems, depending on the features extracted from the brain signals [2,3]. One popular type is based on the imagination of movements from specific body parts, and it usually referred to as Motor Imagery (MI) based BCI [4]. MI-based BCI systems construct features by exploiting the power changes in specific frequency bands that occur during the kinaesthetic imagery of body movements in the sensorimotor cortices. This power variability is known as Event-Related De/Synchronization (ERD/ERS) [5]. A MI-BCI based system is usually composed of several modules comprising signal processing, feature extraction, classification and control commands, for which EEG is the leading non-invasive technology to measure brain signals [4]. MI features are commonly computed by filtering the multivariate signals in subject-specific frequency bands to later compute spatial filters that are able to maximize power differences between different conditions [6]. Classification is usually performed employing linear classifiers such as Linear Discriminant Analysis (LDA). This is most common when the BCI system only discriminates between two different tasks (or classes), but also QDA or SVMs are popular classification procedures [7]. When more classes are involved, or different features are combined, the pattern recognition module might be composed by an ensemble of classifiers, where the common strategy to combine classification outputs is majority voting [8] or arithmetic classifier output mean [9].

Another way to combine information from different features is the inclusion of fuzzy techniques [10]. For example in [11], the authors presented a BCI framework employing fuzzy integrals [12] to model classifier interactions. Another example is [13], where the authors proposed the use of interval-valued aggregation functions. Furthermore, the promising results in [11] show that the classifier fusion in the control command phase is crucial to increase BCI performance. However, choosing the best aggregation function for such system depends on several factors, such as the type or number of classifiers used. Based on the theory of aggregation functions [12], one possible method to

* Corresponding author.
E-mail addresses: javier.fumanal@unavarra.es (J. Fumanal-Idocin), carmen.vidaurre@unavarra.es (C. Vidaurre), fcojavier.fernandez@unavarra.es (J. Fernandez), marisol@unavarra.es (M. Gómez), javier.andreu@essex.ac.uk (J. Andreu-Perez), Mukesh.Prasad@uts.edu.au (M. Prasad), bustince@unavarra.es (H. Bustince).

https://doi.org/10.1016/j.patcog.2023.109924
Received 14 September 2021; Received in revised form 24 July 2023; Accepted 29 August 2023
Available online 4 September 2023

combine classifier outputs is to use a dissimilarity measure between the data and the fused value. A way of measuring this dissimilarity is the so-called penalty functions.

Penalty functions are defined as a measure of deviation from a consensus value, or in other words, as a penalty for not reaching consensus. They have been widely studied in the fuzzy learning field. Penalty functions can be used to build fusion functions which take into account the lack of similarity between inputs. These functions are called penalty-based functions. Some examples are the weighted arithmetic and geometric means and median.

Penalty functions allow the choice of the "best" possible aggregation according to a dissimilarity measure, thereby solving the problem of choosing an aggregation function for a specific problem. However, care needs to be taken with their design. For example, when the quadratic error is set as a penalty function, the arithmetic mean will be selected as the best possible aggregation regardless of the data to be aggregated [14]. This is due to, by definition, the arithmetic mean of the input values is always the value that minimizes the penalty.

The main goal of this paper is to propose and apply a new method to fuse BCI classification outputs to generate a control command. This method is based on a special type of penalty-based aggregation functions: the Multi-Cost Aggregation-Choosing functions (MCAs). MCAs are similar to penalty-based aggregation functions because they establish a disagreement measure between the original data and the aggregated output in order to determine the "best" aggregation. The disagreement measure is constructed using a convex combination of two cost functions. Depending on the convex combination parameter, the proposed functions are able to obtain more meaningful results regarding which aggregation function is denoted as the "best", than the classical approaches. A second goal is to demonstrate the usefulness of MCA functions to classify MI-based BCI data in comparison to the arithmetic mean or the classical penalty-based aggregation functions. To show that this is the case, we perform several favourable comparisons between different aggregation functions and to other previously published work on the same dataset [11,15].

The paper is organized as follows. Section 2 revises the concepts of aggregation and penalty functions. Section 3 introduces the main contributions for this work: Section 3.1 illustrates the BCI framework, and Section 3.2 shows how to process the EEG data. Section 3.3 explains the concept of Quasi-Restricted Equivalence Functions and Quasi-Restricted Dissimilarity Functions and Section 3.4 explains how to use them to construct multi-cost functions. Section 3.5 explains how to mix the different cost functions in MCA in order to optimize the performance in a supervised learning task and, subsequently, Section 3.6 describes how to apply these functions to the BCI MI framework. Section 4 displays the experimental results for the popular BCI IV competition dataset [16] and the Clinical Brain–Computer Interface Challenge (CBCIC) at the IEEE World Congress of Computational Intelligence (WCCI) 2020 [17] using the MCA functions; and in Section 5 we compare those results with other BCI frameworks. Finally, in Section 6 we give our final conclusions and remarks for this work.

2. Preliminaries

This section discusses some of the basic concepts regarding aggregation functions and more precisely, penalty-based aggregation functions.

2.1. Aggregation functions

Aggregation functions are used to fuse information from n sources into one single output [12]. A function A: $[0,1]^n \to [0,1]$ is said to be a n-ary aggregation function if the following conditions hold for any vectors $(x_1, \dots, x_n) \in [0,1]^n$:

- A is increasing in each argument; that is, for every $x_i \in \{1, \dots, n\}$, if $x_i < y, A(x_1, \dots, x_i, \dots x_n) \leq A(x_1, \dots, y, \dots x_n)$
- $A(0, \dots, 0) = 0$
- $A(1, \dots, 1) = 1$

Some examples of n-ary aggregation functions are:

- Arithmetic mean: $A(X) = \frac{1}{n}\sum_{i=1}^{n} x_i$.
- Median: $A(X) = x_m$, where for any permutation $\sigma : \{1, \dots, n\}$ such that $x_{\sigma(1)} \leq \cdots \leq x_{\sigma(n)}, x_m = x_{\sigma(\frac{n+1}{2})}$, if n is odd, and $x_m = \frac{1}{2}(x_{\sigma(\frac{n}{2})} + x_{\sigma(\frac{n+1}{2})})$ if n is even.
- Max: $A(X) = max(x_1, \dots, x_n)$.
- Min: $A(X) = min(x_1, \dots, x_n)$.

2.2. Penalty functions

Penalty-based aggregation functions aim to reduce the disagreement between the input data and the aggregated value in an information fusion process. This process is measured using a disagreement measure called the penalty function.

Let $X = (x_1, \dots, x_n)$ be the inputs and y be the output. If all the inputs coincide $x_1 = \cdots = x_n$, and the output y is the same as all the inputs, then there is no disagreement. If some input $x_i \neq y$, then we impose a "penalty" for this disagreement. The greater the disagreement, and the more inputs disagree with the output, the greater is the penalty. Then, the aggregation function is obtained by finding the aggregated value that minimizes the penalty.

The formal definition of a penalty function reads as follows.

Definition 1.

A function $P : [0,1]^{n+1} \to \mathfrak{R}$ is a penalty function if:

- $P(x, y) \geq 0$ for all x, y;
- $P(x, y) = 0$ if $x_i = y$ for every $i \in \{1, \dots, n\}$;
- $P(x, y)$ is quasi-convex in y for any x.

The penalty based function is $f(x) = \operatorname{argmin} P(x, y)$, if there is a unique minimizer, and $f(x) = \frac{p+q}{2}$ if the set of minimizers is in the interval $[p, q]$.

Any averaging aggregation function, i.e. an increasing function whose output is between the minimum and the maximum of the inputs, can be represented as a penalty based function.

1. Example 1: The arithmetic mean is represented via the penalty function $P(X, y) = \sum_{i=1}^{n}(x_i - y)^2$
2. Example 2: The median is represented via the penalty function $P(X, y) = \sum_{i=1}^{n} |x_i - y|$

Given a penalty function P, a list of n aggregation functions $(Ag_1, \dots, Ag_n)$, and a vector of values to aggregate, X, we compute a finite set of aggregation values over the vector X, $(Ag_1(X), \dots, Ag_n(X))$. Then, we compute $P(X, Ag_i(X))$ for all components in the $(Ag_1(X), \dots, Ag_n(X))$ vector and look for the component that minimizes the value of P, that is

$$\arg\min_i P(X, Ag_i(X))$$

3. Methods

This section illustrates the BCI framework used and how the EEG data were processed. We also introduce the new concepts of Quasi-Restricted Equivalence and Quasi-Restricted Dissimilarity Functions(Q-REF and Q-RDF), and how to construct the newly developed MCAs.

3.1. Motor imagery brain–computer interface framework

The usual modules of a BCI system can be summarized as follows:

1. EEG acquisition with an EEG device, notch filtering to remove power line noise and possibly subsampling and/or bad impedance channel removal.

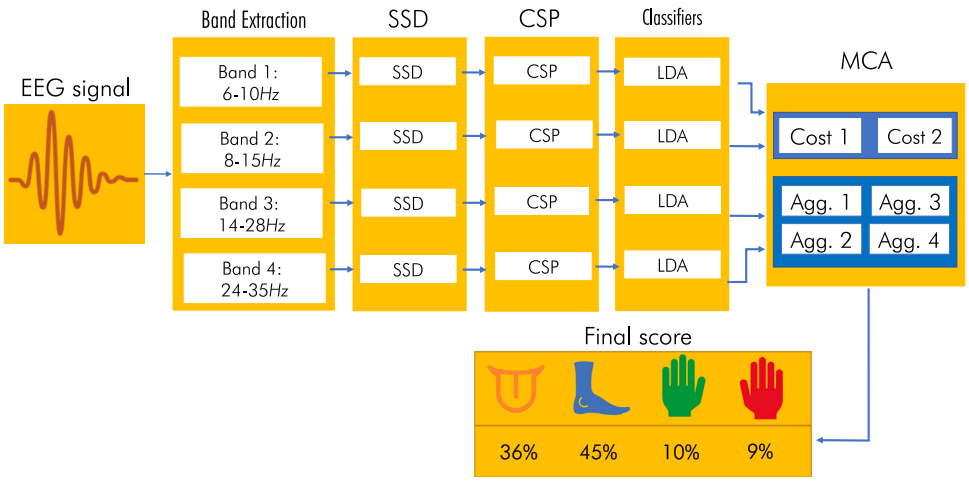

Fig. 1. Visual representation of the framework used in this study. First, we measure the EEG band, and extract the information from four different frequency bands. Then, we apply SSD and subsequently CSP to reduce dimensionality and extract features from each band. From each frequency band we train a different LDA classifier. We make a final decision by aggregating the output from all the LDA classifiers using a MCA (detailed in Section 3.4), which results in the estimated probabilities for each one of the possible classes.

2. Feature extraction from the EEG data measured. Often, band pass filtering in subject-specific or fixed bands is applied to extract specific EEG oscillations [18]. Then, some dimensionality reduction procedure such as Spatio-Spectral Decomposition (SSD) might be applied [19]. Then, Common Spatial Patterns (CSP) are usually employed to compute optimized spatial filters [20] to separate MI tasks. Other possibilities include using Riemannian geometry [21] or time-domain features modelling the signal as Laplacian and Gaussian random process [22]. The extracted features are log-transformed to normalize them.
3. Pattern classification is performed on the extracted features. In this paper we use an ensemble of classifiers to decode the imagery commands. Each base classifier is trained using for example a different band and the final decision is taken combining all of them. The most common way to obtain the final decision is to compute the arithmetic mean of the outputs of all the base classifiers (each one provides a probability for each class), and take the class with a higher aggregated value. The most common base classifier used in combination with CSP filters and log-transformed power values is the Linear Discriminant Analysis (LDA) [23].

A schematic view of the framework used in our experimentation can be found in Fig. 1.

3.2. Feature extraction and classification

In order to extract features, the EEG data were first filtered in four fixed and overlapped frequency bands, covering the range from low μ to high β bands: 6–10, 8–15, 14–28 and 24–35 Hz.

In more detail, the time interval to extract features was optimized for each of the bands using heuristics based on the Event-Related Desynchronization/Synchronization (ERD/ERS) effects typically observed in motor imagery data [24]. The time-resolved ERD/ERS curves were computed as follows: first, the EEG data were spatially filtered using small Laplacian derivations and those channels covering the sensorimotor cortex were selected. Then, these data were band-pass filtered at the band of interest. For each selected Laplacian derivation, the Hilbert transform [25] was applied to obtain the amplitude envelope of the oscillations. EEG activity processed in this way was averaged across epochs separately for each class (left hand/right hand/feet/tongue MI). The time-resolved ERD curve was calculated for each channel according to: $\text{ERD} = 100*(\text{POST}-\text{PRE})/\text{PRE}$, where POST is the EEG processed activity at the post-stimulus interval and PRE is the average activity in the pre-stimulus interval (−500 to 0 ms). Then, the subject-specific time interval (a range of time samples within the active trial time) was selected using heuristics on the ERD/ERS values (see [26]). These heuristics were based on the pair-wise class discriminability of each time sample that was assessed by the signed r^2-value (point biserial correlation coefficient). The signed r^2-value is a correlation coefficient between a real variable (in this case the ERD/ERS value) and a dichotomous one containing class information. Signed r^2-values were computed for each channel and time sample separately and smoothed with a sliding window of 200 ms. The most discriminative time samples were selected using signed r^2-coefficient with 0.8 as threshold value and more samples were iteratively added depending on the averaged discriminative value of the new interval. Fig. 2 shows time intervals averaged across subjects and partitions. They mostly cover the period between 1 and 4 s during feedback, although they are slightly different depending on the band.

After selecting time intervals for each class pair, the EEG data were epoched to form post-stimulus trials. The total dimensionality of the data was then reduced using SSD on the band of interest [27]. This method allows the extraction of oscillatory neuronal sources with optimized Signal-to-noise ratio. It linearly decomposes multivariate data maximizing the power of the signals at specific bands and at the same time minimizing it at the neighbouring frequency bins. After applying SSD, the selected sources were spatially filtered using common spatial pattern (CSP) analysis [26] . Then, log-variance features were computed for each trial of the training set.

The features of the test set were computed by temporally filtering the EEG data in the four bands of interest. For each band and class pair, the corresponding SSD and CSP spatial filters were then applied. Then, the data were epoched using the previously found time intervals. Finally, the variance and logarithm were applied to each of the features in each trial. The features were then log-transformed and LDA classifiers were trained for subsequent classification. We also considered the use of SVM classifiers for this framework, but they showed worse results than those obtained using LDAs in our experiments.

3.3. Quasi-restricted equivalent functions and quasi-restricted dissimilarity functions

In this section we present the concept of Q-REF and Q-RDF. We recall here the notions of Restricted Equivalent Functions (REFs) and Restrict Dissimilarity Functions (RDFs) [28,29] that will be the basis for Q-REF and Q-RDF.

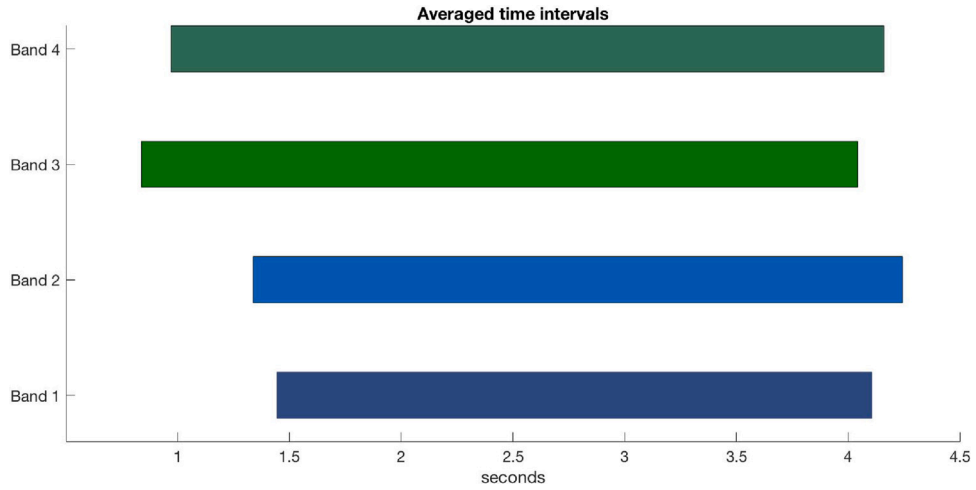

Fig. 2. Average time interval chosen for each wave band.

Definition 2. A function $c : [0,1] \to [0,1]$ is called a strong negation if and only if there exists an automorphism ϕ such that $c(x) = \phi^{-1}(1 - \phi(x))$.

Definition 3. A function $s : [0,1]^2 \to [0,1]$ is called a REF if:

1. $s(x,y) = s(y,x)$;
2. $s(x,y) = 1$ if and only if $x = y$;
3. $s(x,y) = 0$ if and only if $\{x,y\} = \{0,1\}$;
4. $s(x,y) = s(c(x),c(y))$ for all $x, y \in [0,1]$, c being a strong negation.
5. If $x \leq y \leq z$ then $s(x,z) \leq s(x,y)$ and $s(x,z) \leq s(y,z)$.

Definition 4. A function $d : [0,1]^2 \to [0,1]$ is called a RDF if:

1. $d(x,y) = d(y,x)$;
2. $d(x,y) = 0$ if and only if $x = y$;
3. $d(x,y) = 1$ if and only if $\{x,y\} = \{0,1\}$;
4. If $x \leq y \leq z$ then $d(x,y) \leq d(x,z)$ and $d(y,z) \leq d(x,z)$.

In order to deal with more than two inputs, properties are relaxed to introduce the notions of Q-REF function and Q-RDF.

Definition 5. Let $n \geq 1$. A Q-REF function is a function $H_s : [0,1]^{n+1} \to [0,1]$ such that:

$$H_s(X,y) = H_s(x_1,\dots,x_n,y) = 1 \text{ if } x_1 = \cdots = x_n = y. \tag{1}$$

Note that REFs are specific instances of Q-REF functions. And analogously:

Definition 6. Let $n \geq 1$. A Q-RDF function is a function $H_d : [0,1]^{n+1} \to [0,1]$ such that:

$$H_d(X,y) = H_d(x_1,\dots,x_n,y) = 0 \text{ if } x_1 = \cdots = x_n = y. \tag{2}$$

Again, RDFs are specific instances of Q-RDF functions. First of all, observe that these two types of functions are closely related. In fact, we have the following straightforward result.

Proposition 7. *Let $n : [0,1] \to [0,1]$ be a decreasing function such that $n(0) = 1$ and $n(1) = 0$ (a negation). Then, a function $H_s : [0,1]^n \to [0,1]$ is a Q-REF function if and only if $n(H_s)$ is a Q-RDF function.*

We can build general Q-REF and Q-REF functions as follows.

Proposition 8. *Let $h_{s1},\dots,h_{sn} : [0,1]^2 \to [0,1]$ be a family of Q-REF functions and let $A : [0,1]^n \to [0,1]$ be an aggregation function. Then, $H_s^A(x_1,\dots,x_n,y) = A(h_{s1}(x_1,y),\dots,h_{sn}(x_n,y))$ is also a Q-REF function.*

Proposition 9. *Let $h_{d1},\dots,h_{dn} : [0,1]^2 \to [0,1]$ be a family of Q-RDF functions and let $A : [0,1]^n \to [0,1]$ be an aggregation function. Then, $H_d^A(X,y) = A(h_{d1}(x_1,y),\dots,h_{dn}(x_n,y))$ is also a Q-RDF function.*

Proposition 10. *Let $H_{s1}, H_{s2} : [0,1]^n \to [0,1]$ be two Q-REF functions. Then, for every $\alpha \in [0,1]$*

$$\alpha H_{s1} + (1-\alpha) H_{s2} \tag{3}$$

is also a Q-REF function.

Proposition 11. *Let $H_{d1}, H_{d2} : [0,1]^n \to [0,1]$ be two Q-RDF functions. Then, for every $\alpha \in [0,1]$*

$$\alpha H_{d1} + (1-\alpha) H_{d2} \tag{4}$$

is also a Q-RDF function.

Now we consider the convex combination of a Q-REF and a Q-RDF function. If $x_1 = \cdots = x_n = y$, we have that:

$$\alpha H_d(x_1,\dots,x_n,y) + (1-\alpha) H_s(x_1,\dots,x_n,y) = 1 - \alpha \tag{5}$$

So:

Proposition 12. *Let $H_d, H_s : [0,1]^n \to [0,1]$ be a Q-REF and a Q-RDF function, respectively. Then, for any $\alpha \in [0,1[$, the function:*

$$H(X,y) = \min\left(\frac{\alpha H_d(X,y) + (1-\alpha) H_s(X,y)}{1-\alpha}, 1\right) \tag{6}$$

is a Q-REF function.

3.4. Multi-cost aggregation-choosing functions

A penalty function is characterized using a disagreement measure that quantifies how different the inputs X are with respect to the resulting aggregated value y. The use of penalty functions mitigates the problem of choosing an appropriate aggregation function: given a disagreement measure, the one whose output minimizes the disagreement measure will be chosen. The most common disagreement measure is the quadratic error, however, the arithmetic mean will always deliver the best result according to this measure [14].

To solve this problem we propose the MCAs, that present two novelties compared to the already existing penalty-based aggregation functions:

- In order to measure the disagreement, we consider a cost function. We do so as a cost function can be applied in situations where the term consensus would not be adequate. For example,

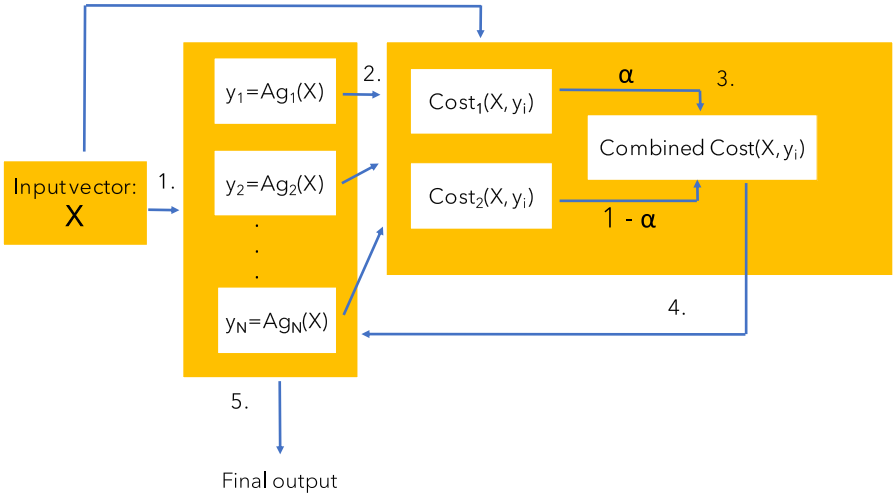

Fig. 3. Visual scheme for the MCA aggregation process. (In the case of the BCI framework in Fig. 1, X is the output of the LDA classifiers). **1**. We compute all the possible aggregations. **2**. We compute both cost functions for each aggregation output (y_i) with respect to the input vector. **3**. We combine both costs for each aggregation with the mixing parameter α. **4**. We select the aggregation with the least cost value. **5**. That aggregation is the final output of the MCA.

in a N-class classification problem, a result of 1/N probability for a specific class, indicates that the output does not contain almost any information. In this case, the cost function can be used to penalize this kind of outcome. In this manuscript, we chose Q-REFs and Q-RDFs as cost functions, as studied in Section 3.3.

- The use of a convex combination of two functions instead of a single function avoids trivial results such as the one regarding the quadratic cost, which will always be minimized by the arithmetic mean independently of the input data.

A schematic view of the aggregation process using a MCA can be found in Fig. 3.

3.4.1. Costs used

We have considered a set of Q-RDFs and Q-REF measures as cost functions. As studied in Section 3.3, and depending on the mixed functions, their convex combination is also a Q-REF or a Q-RDF. Given a vector X of size n, where each element of X is contained in the unit interval the Q-RDFs measures studied are the following:

- Huber loss:

$$h(x_i, y) = \begin{cases} (x_i - y)^2 & (x_i - y)^2 \leq M \\ 2 * M * (x_i - y)^2 - M * M & (x_i - y)^2 \geq M \end{cases} \quad (7)$$

where $H(X, y) = \frac{1}{n}\sum_{i=1}^{n} h(x_i, y)$. (We use $M = 0.3$ for our experimentation)

- Quadratic cost:

$$H(X, y) = \frac{1}{n}\sum_{i=1}^{n}(x_i - y)^2 \quad (8)$$

- Optimistic cost:

$$H(X, y) = (max(X) - y)^2 \quad (9)$$

- Pessimistic cost:

$$H(X, y) = (min(X) - y)^2 \quad (10)$$

The Q-REF measure studied is:

- Anti-consensus cost:

$$H(X, y) = \frac{1}{n}\sum_{i=1}^{n}(1 - (x_i - y)^2) \quad (11)$$

Fig. 4 shows the effects of a penalty aggregation using the classical aggregations applied to the BCI data, with a sample of 100 five dimensional (5-D) random vectors with numbers in [0,1]. The histograms are computed over the results of aggregating 100 5-D random vectors. It is visible that the optimistic and pessimistic costs have a "skewing effect", so that the histogram is sharply moved to greater and lower values, respectively. It can also be observed that there are two very similar cost functions: the quadratic and the Huber costs. This is expected as they only differ in "extreme" values. Finally, the anti-consensus cost exhibits the most disperse histogram.

3.4.2. Combining costs

The combination of two costs using a convex combination requires an $\alpha \in]0, 1[$ parameter. Depending which are the functions to be combined, Q-REFs or Q-RDFs, different formulas should be used:

- Both are the same type:

$$\text{Combined Cost} = \alpha * cost_1 + (1 - \alpha) * cost_2 \quad (12)$$

- One is a Q-REF and the other is a quasi-dissimilarity:

$$\text{Combined cost} = \min\left(\frac{\alpha cost_1 + (1 - \alpha) cost_2}{1 - \alpha}, 1\right) \quad (13)$$

Thus, the combined cost will be another Q-RDF when both cost_1 and cost_2 are both Q-RDF, and a Q-REF otherwise.

Fig. 5 shows how the cost functions behave for a five-dimensional vector of random numbers in the interval $[0, 1]$: $(0.60, 0.85, 0.61, 0.52, 0.52)$.

In order to show how each cost combination works, we computed each of them varying the parameter α within the $]0, 1[$ interval. We also marked the preferred value for each one. Fig. 5a and Fig. 5b correspond to Q-RDFs and Fig. 5c and Fig. 5d are Q-REFs.

Fig. 6 studies the effect of different α values in the quadratic and optimistic cost based on the same random vectors as before and shows that indeed the α parameter has a notorious influence in the chosen aggregation.

3.5. Selecting the α parameter in a multi-cost aggregation-choosing function for a supervised classification task

As studied in Section 3.4, α plays a crucial role in the output of a MCA. Choosing the optimal value for this parameter is not an easy task because it heavily depends on the application.

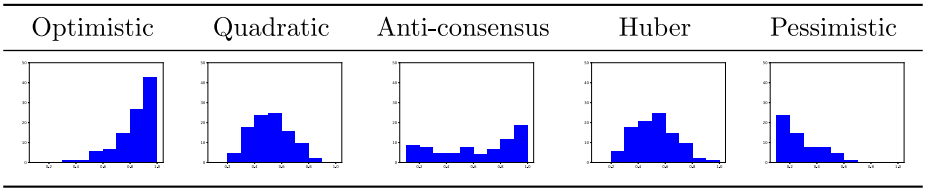

Fig. 4. Histogram of aggregated values for the optimistic, the quadratic, anti-consensus, Huber and pessimistic costs using the maximum, minimum, arithmetic mean and median as possible aggregations in the MCA, for a random sample of 100 vectors of size 5 in the $[0, 1]$ range. We represent in the x axis the α value and, in the y axis, the frequency of the aggregated output values in each range for each sampled random vector.

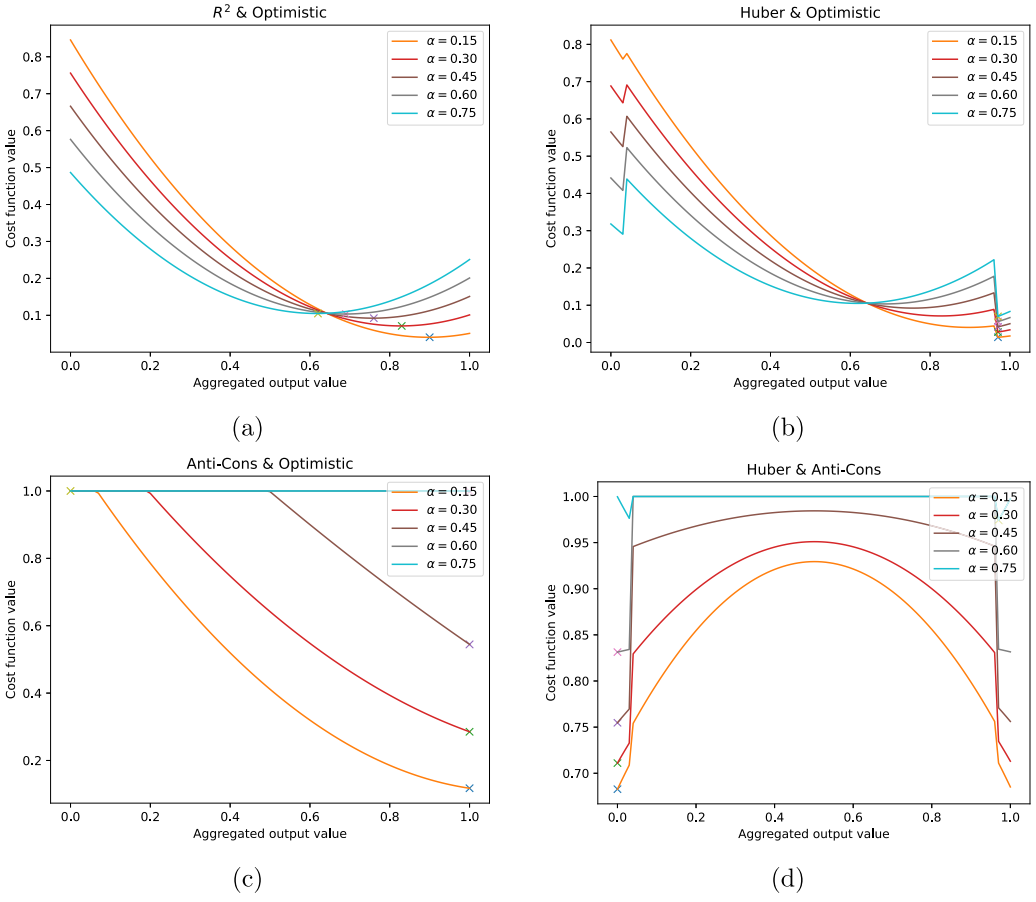

Fig. 5. Effect of different α parameters for a vector of five, randomly chosen numbers $\in [0, 1]$: $(0.60, 0.85, 0.61, 0.52, 0.52)$. The $\times$ marks the minimum for each α parameter in each error configuration.

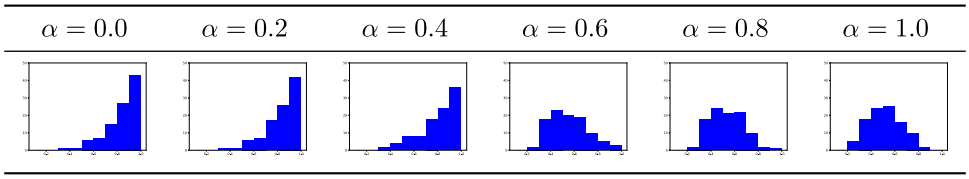

Fig. 6. Histogram of aggregated values for the quadratic & optimistic cost using the maximum, minimum, arithmetic mean and median as possible aggregations in the MCA, for a random sample of 100 vectors of size 5 in the $[0, 1]$ range, using different α values. We represent in the x axis the α value and, in the y axis, the frequency of the aggregated output values in each range for each sampled random vector.

Although some fixed α value might work sufficiently well for some applications, the fine-tuning of this parameter can also increase the performance of supervised classification scenarios. As α is restricted to the $[0, 1]$ interval, a dense sampling Montecarlo optimization with accuracy as target metric is a good option to select α. Nevertheless, when a system is composed of more than one aggregation process and more than one MCA, the optimization needs to be performed over a vector of numbers instead of just one value. Depending on the size of the vector, it is still possible to optimize it performing the same procedure as for a single value. However, one of the key ideas of the original penalty-based aggregations is to find a suitable aggregation for each vector of inputs. By choosing the same α parameter for each individual vector of inputs, this philosophy is somewhat disregarded. In that case, computing an adaptive α, chosen according to the vector

of inputs appears more appropriate. We named this procedure the adaptive MCA.

The adaptive MCA computation carries an additional difficulty, as we need to somehow relate our input vector with the final outcome, which is the label for each sample. We propose the use of a regression:

$$\alpha = f(WX + b) \tag{14}$$

where f is an activation function, W is the weight matrix, X is the input vector and b is the bias.

In this formula, both W and b matrices should be optimized. In this case a Montecarlo optimization using the accuracy as the target metric is not appropriate because the size of X might be too large, turning the optimization unstable due to the "curse of dimensionality". Thus, we propose to learn W and b using a gradient descent optimization, where a set of initial "real" α values is necessary.

Although "real" α values do not exist, there is a ground truth label for each example. Suppose an aggregation function exists whose result leads to a correct classification. Then, there is a possible value of α that selects this optimal aggregation, and thus, correctly classifies the sample. This value is considered a "real" α (α_{real}) because it correctly classifies the sample. Usually, there will be several different values of this parameter leading to the correct classification of that example. We call the set of α_{real} the $\boldsymbol{\alpha}_{est}$.

The next problem is determining which value in the set of $\boldsymbol{\alpha}_{est}$ should be selected as training label to obtain W and b. Since we are interested in maximizing the variability in the selection process, we should prefer an α whose output is as undetermined as possible. For example: in the case of the quadratic & optimistic costs if the α value is 0, the chosen aggregation will always be the arithmetic mean, and if it is 1, it will always be the maximum. Thus, the preferred α value should be the furthest from 0 and 1, or in other words, that α should be as close as possible to 0.5.

In the following Section 3.5.1 we illustrate this process for the quadratic & optimistic costs. The same procedure can be applied to the rest of the Q-REF and the Q-RDF combinations.

3.5.1. Training adaptive α values for the quadratic & optimistic costs

This section illustrates the process of generating a numerical value, out of the set of $\boldsymbol{\alpha}_{est}$, that can be used as label to train the Eq. (14). This process consists of two steps:

1. Compute the $\boldsymbol{\alpha}_{est}$ set.
2. Determine the best value in $\boldsymbol{\alpha}_{est}$, that will be the closest to 0.5.

We define the predicted probability of the sample x to be of class c as $c(x)$ within C possible classes, and the ground truth of x as y_x. We define the classification threshold as $t = \max c_i(x), c_i \in C$. It is evident that for any value $c(x) \geq t$, if $y_x = c$, then the classification is correct. As aforementioned the quadratic error favours the arithmetic mean over the rest of the aggregations, and the optimistic error, favours the maximum. We consider the convex combination of both errors and the mixing parameter α:

$$Cost(X)_\alpha = \alpha * mean(X) + (1 - \alpha) * max(X) \tag{15}$$

The MCA that uses this cost increases with respect to the α value, because as the value of α grows, the preferred value in the error formula gets closer to the maximum.

Supposing that an α' exists such that for the class c, and $c = y_x$, the $\text{MCA}_{\alpha'}(x) = t$, all $\alpha'' > \alpha'$ will result in a $\text{MCA}_{\alpha''}(x) >= MCA_{\alpha'}(x) >= t$, which will result in correct classification. This means that $\boldsymbol{\alpha}_{est}$ are all the α values bigger than α'. Then, the optimal $\boldsymbol{\alpha}_{est}$ is just the closest to 0.5.

The process is very similar for any other combination of Q-REF and Q-RDF functions, but if the combined cost is not monotone with respect to α, then $\boldsymbol{\alpha}_{est}$ can be disjoint.

Example 3.1. Taking a vector of five random numbers: $\mathbf{x} = [0.4, 0.9, 0.1, 0.5, 0.3]$, we consider these five random numbers the output of five classifiers, i.e. the probability of a sample to be of class y, being y the real label of that sample.

We select a MCA that uses the maximum and mean cost, and chooses among the average (0.44), median (0.40), minimum (0.1) and maximum (0.90) aggregations. Then, for any $\alpha < 0.5$, the MCA will choose the average, and for any $\alpha > 0.5$ the MCA will select the maximum. For $\alpha = 0.5$ both values are eligible. Since the average is 0.44, if we aggregate using this value, the final result will not correctly classify $\mathbf{x}$. If we aggregate using the maximum, then the aggregation will correctly classify the sample. So, α_{est} in this case will be all α values greater than 0.5.

As final training label we take the immediate value following 0.5 and adjust it to the desired precision. For example, if we consider decimals until the third digit, the target α_{real} to learn for $\mathbf{x}$ would be 0.501.

3.6. Multi-cost aggregation-choosing functions in the brain–computer interface framework

We use the MCA functions in the aggregation function in the decision making phase of the BCI framework. Each MCA is composed of a set of possible aggregations to choose from and a cost function. In the case of the adaptive-MCA, it is also composed of a weight matrix and a bias vector. We used a set composed of four classical aggregations: minimum, maximum, median, and the arithmetic mean. We tested all possible combinations of Q-REFs and Q-RDFs and presented them in Section 3.4.1.

In the case of using a non adaptive MCA, the mixing parameter was learnt using a Montecarlo sampling of 200 possible α values in the $]0, 1[$ range. On the other hand, recall that in the case of the adaptive MCA we need to establish the X matrix and the activation function f of Eq. (14) to apply the procedure detailed in Section 3.5. Matrix X corresponds to the outputs of all the classifiers in the BCI framework for each sample, whereas f is a linear activation function ($f(x) = x$). Then, W and b in Eq. (14) are learnt using gradient descent optimization.

4. Results

In this section we discuss the outcomes of applying our new approaches to the BCI competition IV dataset 2a (IV-2a) [16] and the Clinical BCI Challenge WCCI 2020 dataset (CBCIC) [17].

- **Dataset 1**: The IV-2a dataset has been extensively used to test different BCI systems (see for example, [30–32]). It consists of four motor imagery tasks (tongue, foot, left-hand and right-hand) performed by 9 volunteers. For each task, 22 EEG channels were collected, with a total of 288 trials for each participant. Trials are evenly distributed among the 4 classes. For this dataset, we studied the classification task from two different perspectives: binary classification of the left and right hand classes, which is a common choice of tasks in the literature [32,33]; and four-class classification: left hand, right hand, foot and tongue.
- **Dataset 2**: the CBCIC dataset consist of brain imaging signals from 10 hemiparetic stroke patients with hand functional disability in a rehabilitation task. The data contains 80 diferent trials of left/right hand movements. Decoding motor cortical signals of brain-injured presents several challenges as the presence of irregular because of the altered neurodynamics [17].

For both datasets, the evaluation process is the same. Each participant's dataset was randomly sampled in ten different partitions (each with 50% train and 50% test trials). A total of 90, respectively 80 datasets were generated for the IV-a Competition and CBCIC datasets. The final performance of each configuration was obtained averaging

Table 1
Results using Penalty-based aggregation/arithmetic mean.

Dataset	Aggregation	Accuracy
IV-2a dataset	Average/Classic Penalty-based aggregation	0.7974
CBCIC dataset	Average/Classic Penalty-based aggregation	0.8215

Table 2
Accuracy results for binary classification using MCAs optimized with Montecarlo Sampling in the binary task.

Dataset		Quadratic	Optimistic	Huber	Pessimistic
IV-2a	Optimistic	0.7904			
	Huber	0.7960	0.7931		
	Pessimistic	0.7933	0.7938	0.7915	
	Anti-consensus	0.7955	0.8022	0.7939	**0.8030**
CBCIC	Optimistic	0.8123			
	Huber	0.8215	0.8142		
	Pessimistic	0.8113	0.8000	0.8224	
	Anti-consensus	0.8215	0.8221	**0.8231**	0.8215

Table 3
Accuracy results for the adaptive MCA optimized with the algorithm in Section 3.5 in the binary task.

Dataset		Quadratic	Optimistic	Huber	Pessimistic
IV-2a	Optimistic	0.8000			
	Huber	0.7974	0.7994		
	Pessimistic	0.8000	0.7798	0.7994	
	Anti-consensus	0.7974	**0.8038**	0.7970	**0.8038**
CBCIC	Optimistic	0.8123			
	Huber	0.8215	0.8215		
	Pessimistic	0.8113	0.8132	**0.8224**	
	Anti-consensus	0.8215	0.8212	0.8221	0.8136

each single dataset accuracy. The results were obtained using different aggregation functions in the decision making phase and compared the newly proposed MCAs. Both the adaptive and the non-adaptive mixing parameter were employed with a set of standard aggregations and also with the already existing penalty-based aggregation functions.

Furthermore, results for each individual subject are available in the following GitHub repository: https://github.com/Fuminides/MCA_BCI_results.

4.1. Results for left/right hand motor imagery classification with stroke patients (CBCIC) and BCI competition IV-2a datasets

Table 1 shows the results for the binary classification using the state-of-art BCI framework with the arithmetic mean as the fusion function for the classifiers output. Recall here that the choice of the penalty-based aggregation function is always the arithmetic mean, thus the results are the same for both.

Table 2 displays the results for all the possible MCA functions. The selected aggregation functions are a set of classical aggregation procedures: arithmetic mean, median, minimum and maximum. In this case, α was found with a simple Montecarlo sampling optimization, using a ten-fold validation on the train set to determine its performance. Table 3 presents analogous results to those in Table 2, but using the algorithm proposed in Section 3.5.1 to learn the α parameter.

These tables show that the best result is obtained for a MCA with α set by the procedure described in Section 3.5, resulting in 0.8038 of accuracy in the IV-2a dataset, and 0.8231 in the CBCIC dataset. The second best result is obtained for a MCA with a Montecarlo optimization. Both MCA optimization algorithms improve the result of the classical arithmetical mean: 0.7974 and 0.8224 for the IV-2a and CBCIC dataset, respectively. We performed a Friedman test, as both populations were not normal according to Shapiro–Wilk test. However, no statistical differences were found.

Table 4
Accuracy 4-class classification results using Penalty-based aggregation/arithmetic mean in the IV-2a dataset.

Dataset	Aggregation	Accuracy
IV-2a	Average/Classic Penalty-based aggregation	0.6056

4.2. Results for 4-class motor-imagery classification problem (BCI competition IV-2a)

The 4-class problem is analogous to the left/right hand problem including "foot" and "tongue" tasks, which are noticeably harder to discriminate [16]. To study this problem we have performed similar experiments to those of the left/right hand classification task.

Table 4 shows the results for the state-of-art BCI framework using the arithmetic mean, which is similar to computing the classical penalty-based aggregation. The obtained accuracy was 0.6056.

Table 5 displays the results for the state-of-art BCI aggregation framework using the MCA functions where the α parameter was optimized with the Montecarlo sampling algorithm. We found many combinations of costs that resulted in MCAs surpassing the result of the arithmetic mean. The best result found here was 0.6243 of accuracy.

Finally, Table 6 presents the results for the traditional BCI framework using the MCA functions where the α parameter was optimized with the algorithm in Section 3.5. Here, the best result was 0.6167, which was again better than the one obtained using the arithmetic mean, but worse than the one using the Montecarlo sampling optimization.

According to a Shapiro–Wilk test, the accuracy populations were not normal. So, we used a Friedman test followed by pairwise comparisons with Wilcoxon post-hoc tests to look for statistical differences. The resulting P-values are reported in Table 7. We found the Montecarlo optimization to significantly outperform the rest.

5. Comparison with other motor imagery-brain computer interface decoding methods

In this Section we compare our results with two other MI-BCI systems. We employed both the IV-2a and the CBCIC datasets. The selected BCI frameworks are described in the following:

1. Multimodal Fuzzy Fusion framework (MFF) [11]: in this work the authors use a Fast Fourier transform to extract features from the original EEG data, then they construct a classifier ensemble using different types of classifiers and a fuzzy integral.
2. One Versus One and Gradient Boosting [15]: the authors used Gradient Boosting classifiers [34] to select the optimal classification features. They structured the decision making phase with different One versus One (OVO) strategies: a classical OVO, and a tree structure for the OVO classifiers (tree-OVO).
3. Multiscale CSP [35]: the authors extended CSP using different time windows, to obtain features from different temporal scales, which then are used to train a SVM classifier.
4. EEG net [36]: in this work, the authors proposed a specific architecture of a Convolutional Neural Network for EEG signals, in order to incorporate in the network different well-known concepts of feature extraction in BCI.
5. Shallow and Deep nets [37]: are two convolutional neural networks, composed of 2 and 4 blocks of convolution and max pooling blocks.

In order to compare our feature extraction method with others, we also used the feature extraction method developed in [35] with the proposed MCA. In this framework, the features from different time windows are concatenated and fed to a classifier. In order to use the proposed MCA, instead of concatenating these features into a single

Table 5
Accuracy 4-class classification results using MCAs optimized with Montecarlo Sampling.

Dataset		Quadratic	Optimistic	Huber	Pessimistic
IV-2a	Optimistic	0.6066			
	Huber	0.6041	0.6040		
	Pessimistic	0.6027	0.6046	0.5993	
	Anti-consensus	0.6056	0.6087	0.6018	**0.6243**

Table 6
Accuracy 4-class classification results adaptive MCAs optimized with the algorithm in Section 3.5.

Dataset		Quadratic	Optimistic	Huber	Pessimistic
IV-2a	Optimistic	0.6124			
	Huber	0.6056	0.6113		
	Pessimistic	0.6050	0.5966	0.5990	
	Anti-consensus	0.6056	0.6148	0.6033	**0.6167**

Table 7
Statistical significances in the four classes classification problem with Wilcoxon post-hoc for the different MPA approaches and the arithmetic mean.

Dataset		Arithmetic mean	MCA Montecarlo
IV-2a	MCA Montecarlo	$P < .001$	
	MCA adaptive	$P < .001$	$P < .001$

Table 8
Results of each BCI framework in the IV-2a dataset, full task.

BCI framework (IV-2a)	Accuracy	F1-Score
MCA Montecarlo	0.6243	0.6225
MCA adaptive	0.6167	0.6016
Multiscale MCA	**0.7433**	**0.7271**
MFF-Sugeno [11]	0.6424	0.6110
MFF-Sugeno Hamacher [11]	0.6898	0.6889
Gradient boosting OVO [15]	0.5245	0.2264
Gradient boosting tree-OVO [15]	0.4524	0.1163
Multiscale CSP [35]	0.7328	0.7066
EEG Net [36]	0.5747	0.3698
Shallow Net [37]	0.6362	0.5986
Deep net [37]	0.5196	0.4218

vector, we form k different vectors concatenating the features from adjacent frequencies. For each of these feature vectors we train a classifier, and then we fuse the logits from these classifiers using a MCA. We call this framework the multiscale MCA

We performed these comparisons using the same procedure as in Section 4 and that we summarize here: we randomly sampled 10 partitions composed of 50% train and 50% test data for each subject. This resulted in 90 different datasets for the IV-2a competition data, and 80 datasets for the CBCIC. As evaluation metric, we used the mean accuracy obtained in the test partitions.

Table 8 shows the results for each of the different configurations tested for the IV-2a dataset. Table 9 shows the same comparison for the CBCIC dataset. We found that the our method performed best for the CBCIC dataset, and that the Multiscale MCA over performed the rest for the IV-2a dataset. In this configuration of the Multiscale MCA we used two feature vectors and the Huber & Anti-consensus cost.

Table 10 shows the results for the Wilcoxon post-hoc after Friedman test, comparing the MCA Montecarlo with the rest of the frameworks tested for the IV-2a dataset. We found that MCA Montecarlo significantly outperforms OVO, tree-OVO frameworks but the MFF performed statistically better than our proposal. Table 11 shows the analogous results for the CBCIC dataset. In this case we found that our method performed significantly better than the MFF.

Table 9
Results of each BCI framework in the CBCIC dataset.

BCI framework (CBCIC)	Accuracy	F1-Score
MCA Montecarlo	**0.8231**	**0.8243**
MCA adaptive	0.8224	0.8224
Multiscale MCA	0.7777	0.7551
MFF-Sugeno [11]	0.7990	0.7919
MFF-Sugeno Hamacher [11]	0.8145	0.7922
Gradient boosting [15]	0.5956	0.5354
Multiscale CSP [35]	0.7956	0.7911
EEG Net [36]	0.6562	0.5933
Shallow Net [37]	0.7453	0.7342
Deep net [37]	0.5331	0.4218

Table 10
Results for the Wilcoxon post-hoc, comparing the two best MCA solutions with other BCI systems in the IV-2a dataset.

(IV-2a)	MFF-Sugeno	MFF-Sugeno Hamacher	OVO	Multiscale CSP
MCA Montecarlo	$P = .02$	$P < .001$	$P < .001$	$P < .001$
Multiscale MCA	$P < .001$	$P < .001$	$P < .001$	$P < .001$

Table 11
Results for the Wilcoxon post-hoc, comparing the MPA Montecarlo with other aggregation based BCI systems in the CBCIC dataset.

(CBCIC)	MFF-Sugeno	MFF-Sugeno Hamacher
MCA Montecarlo	$P < .001$	$P < .001$

6. Conclusions and future work

In this paper we introduced the combination of two generalized versions of REFs and RDFs cost functions to choose an optimal aggregation regarding a vector of inputs. We showed that this technique is able to enhance the classifier fusion phase in two BCI frameworks and can improve the results of the arithmetic mean (and subsequently, the classical penalty-based aggregations) for both binary and multiclass MI classification problems of the BCI Competition IV 2a and CBCIC datasets.

For the latter dataset, our BCI framework performed better than the Deep Learning, OVO, Multimodal Fusion and Multiscale CSP proposals regardless the aggregation function chosen. We also found that the best MCAs computed included the Anti-Consensus cost, which favours values that differ from the consensus. This result suggests that the most useful aggregated values to perform classification can be different to the original consensus of the classifiers. This idea differs from the original penalty functions intention, which was to measure disagreement in order to choose the value that minimizes it.

Future research shall aim at improving the accuracy of the system by studying different ways to learn which costs should be combined for

a given task. We also intend to study the trade-off between diversity and accuracy in the classifiers to aggregate, as the more diverse these outputs are, the more meaningful the aggregation process can be.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Data availability

No data was used for the research described in the article.

Acknowledgements

Javier Fumanal Idocin, Javier Fernandez, and Humberto Bustince's research has been supported by the project PID2019-108392GB I00 (AEI/10.13039/501100011033).

Carmen Vidaurre research has been funded by the project RyC-2014-15671.

References

[1] Y.-K. Wang, T.-P. Jung, C.-T. Lin, Eeg-based attention tracking during distracted driving, IEEE Trans. Neural Syst. Rehabil. Eng. 23 (6) (2015) 1085–1094.
[2] M.-H. Lee, S. Fazli, J. Mehnert, S.-W. Lee, Subject-dependent classification for robust idle state detection using multi-modal neuroimaging and data-fusion techniques in bci, Pattern Recognit. 48 (8) (2015) 2725–2737.
[3] T. Nierhaus, C. Vidaurre, C. Sannelli, K.-R. Mueller, A. Villringer, Immediate brain plasticity after one hour of brain–computer interface (bci), J. Physiol. 599 (9) (2021) 2435–2451.
[4] R. Scherer, C. Vidaurre, Chapter 8 - motor imagery based brain–computer interfaces, in: P. Diez (Ed.), Smart Wheelchairs and Brain-Computer Interfaces, Academic Press, 2018, pp. 171–195.
[5] C. Sannelli, C. Vidaurre, K.-R. Müller, B. Blankertz, A large scale screening study with a smr-based bci: Categorization of bci users and differences in their smr activity, PLoS One 14 (1) (2019) e0207351.
[6] J. Jin, R. Xiao, I. Daly, Y. Miao, X. Wang, A. Cichocki, Internal feature selection method of csp based on l1-norm and dempster–shafer theory, IEEE Trans. Neural Netw. Learn. Syst. 32 (11) (2021) 4814–4825.
[7] P. Arpaia, A. Esposito, A. Natalizio, M. Parvis, How to successfully classify eeg in motor imagery bci: A metrological analysis of the state of the art, J. Neural Eng. 19 (3) (2022) 031002.
[8] A. Soria-Frisch, A Critical Review on the Usage of Ensembles for BCI, Springer, 2013, pp. 41–65, (Chapter 3).
[9] S. Aggarwal, N. Chugh, Review of machine learning techniques for eeg based brain computer interface, Arch. Comput. Methods Eng. (2022) 1–20.
[10] D. Achanccaray, K. Acuna, E. Carranza, J. Andreu-Perez, A virtual reality and brain computer interface system for upper limb rehabilitation of post stroke patients, in: 2017 IEEE International Conference on Fuzzy Systems, FUZZ-IEEE, IEEE, 2017, pp. 1–5.
[11] L.-W. Ko, Y.-C. Lu, H. Bustince, Y.-C. Chang, Y. Chang, J. Ferandez, Y.-K. Wang, J.A. Sanz, G.P. Dimuro, C.-T. Lin, Multimodal fuzzy fusion for enhancing the motor-imagery-based brain computer interface, IEEE Comput. Intell. Mag. 14 (1) (2019) 96–106.
[12] G. Beliakov, H. Bustince, T.C. Sánchez, A Practical Guide To Averaging Functions, 329, Springer, 2016.
[13] J. Fumanal-Idocin, Z. Takac, J. Fernandez, J.A. Sanz, H. Goyena, C.-T. Lin, Y. Wang, H. Bustince, Interval-valued aggregation functions based on moderate deviations applied to motor-imagery-based brain computer interface, IEEE Trans. Fuzzy Syst. (2021).
[14] H. Bustince, G. Beliakov, G.P. Dimuro, B. Bedregal, R. Mesiar, On the definition of penalty functions in data aggregation, Fuzzy Sets and Systems 323 (2017) 1–18.
[15] M. Vijay, A. Kashyap, A. Nagarkatti, S. Mohanty, R. Mohan, N. Krupa, Extreme gradient boosting classification of motor imagery using common spatial patterns, in: 2020 IEEE 17th India Council International Conference, INDICON, 2020, pp. 1–5.
[16] M. Tangermann, K.-R. Müller, A. Aertsen, N. Birbaumer, C. Braun, C. Brunner, R. Leeb, C. Mehring, K.J. Miller, G. Mueller-Putz, et al., Review of the bci competition iv, Front. Neurosci. 6 (2012) 55.
[17] A. Chowdhury, J. Andreu-Perez, Clinical brain-computer interface challenge 2020 (cbcic at wcci2020): Overview, methods and results, IEEE Trans. Med. Robot. Bionics (2021).
[18] Z. Khademi, F. Ebrahimi, H.M. Kordy, A review of critical challenges in mi-bci: From conventional to deep learning methods, J. Neurosci. Methods 383 (2023) 109736.
[19] C. Vidaurre, T. Jorajuría, A. Ramos-Murguialday, K.R. Müller, M. Gómez, V.V. Nikulin, Improving motor imagery classification during induced motor perturbations, J. Neural Eng. 18 (4) (2021).
[20] J. Jin, R. Xiao, I. Daly, Y. Miao, X. Wang, A. Cichocki, Internal feature selection method of csp based on l1-norm and dempster–shafer theory, IEEE Trans. Neural Netw. Learn. Syst. 32 (11) (2021) 4814–4825.
[21] A. Barachant, S. Bonnet, M. Congedo, C. Jutten, Classification of covariance matrices using a riemannian-based kernel for bci applications, Neurocomputing 112 (2013) 172–178.
[22] M. Hamedi, S.-H. Salleh, A.M. Noor, I. Mohammad-Rezazadeh, Neural network-based three-class motor imagery classification using time-domain features for bci applications, in: 2014 IEEE Region 10 Symposium, IEEE, 2014, pp. 204–207.
[23] A.J. Izenman, Linear discriminant analysis, in: Modern Multivariate Statistical Techniques, Springer, 2013, pp. 237–280.
[24] C. Vidaurre, S. Haufe, T. Jorajuría, K.-R. Müller, V.V. Nikulin, Sensorimotor functional connectivity: a neurophysiological factor related to bci performance, Front. Neurosci. (2020) 1278.
[25] M. Feldman, Hilbert transform in vibration analysis, Mech. Syst. Signal Process. 25 (3) (2011) 735–802.
[26] C. Sannelli, C. Vidaurre, K. Müller, B. Blankertz, A large scale screening study with a smr-based bci: Categorization of bci users and differences in their smr activity, PLoS One 14 (2019) e0207351.
[27] V.V. Nikulin, G. Nolte, G. Curio, A novel method for reliable and fast extraction of neuronal eeg/meg oscillations on the basis of spatio-spectral decomposition, Neuroimage 55 (4) (2011) 1528–1535.
[28] H. Bustince, E. Barrenechea, M. Pagola, Restricted equivalence functions, Fuzzy Sets and Systems 157 (17) (2006) 2333–2346.
[29] H. Bustince, E. Barrenechea, M. Pagola, Relationship between restricted dissimilarity functions, restricted equivalence functions and normal en-functions: Image thresholding invariant, Pattern Recognit. Lett. 29 (4) (2008) 525–536.
[30] M. Xu, J. Yao, Z. Zhang, R. Li, B. Yang, C. Li, J. Li, J. Zhang, Learning eeg topographical representation for classification via convolutional neural network, Pattern Recognit. 105 (2020) 107390.
[31] J. Fumanal-Idocin, Y.-K. Wang, C.-T. Lin, J. Fernández, J.A. Sanz, H. Bustince, Motor-imagery-based brain-computer interface using signal derivation and aggregation functions, IEEE Trans. Cybern. (2021).
[32] A. Jafarifarmand, M.A. Badamchizadeh, S. Khanmohammadi, M.A. Nazari, B.M. Tazehkand, A new self-regulated neuro-fuzzy framework for classification of eeg signals in motor imagery bci, IEEE Trans. Fuzzy Syst. 26 (3) (2017) 1485–1497.
[33] A.S. Aghaei, M.S. Mahanta, K.N. Plataniotis, Separable common spatio-spectral patterns for motor imagery bci systems, IEEE Trans. Biomed. Eng. 63 (1) (2015) 15–29.
[34] A. Natekin, A. Knoll, Gradient boosting machines, a tutorial, Front. Neurorobot. 7 (2013) 21.
[35] M. Hersche, T. Rellstab, P.D. Schiavone, L. Cavigelli, L. Benini, A. Rahimi, Fast and accurate multiclass inference for mi-bcis using large multiscale temporal and spectral features, in: 2018 26th European Signal Processing Conference, EUSIPCO, 2018, pp. 1690–1694.
[36] V.J. Lawhern, A.J. Solon, N.R. Waytowich, S.M. Gordon, C.P. Hung, B.J. Lance, Eegnet: a compact convolutional neural network for eeg-based brain–computer interfaces, J. Neural Eng. 15 (5) (2018) 056013.
[37] S.R. Tibor, S.J. Tobias, F.L.D. Josef, G. Martin, E. Katharina, T. Michael, H. Frank, B. Wolfram, B. Tonio, Deep learning with convolutional neural networks for eeg decoding and visualization, Hum. Brain Mapp. 38 (11) (2017) 5391–5420.

Javier Fumanal is a predoctoral researcher at the Public University of Navarre. His mains research interests consists of brain–computer interfaces and network analysis.

Carmen Vidaurre is Ph.D. in Telecommunication Engineering. She was Marie Curie Fellow at the Fraunhofer Institute (Berlin) and postdoc in the ML group at TU-Berlin. She was then Ramon y Cajal Fellow at UPNA and currently is Ikerbasque Research Associate at Tecnalia. She develops methods to study the nervous system.

Javier Fernandez is currently an Associate Lecturer with the Department of Statistics, Computer Science and Mathematics, Public University of Navarra., Pamplona, Spain. He is the author or coauthor of approximately 70 original articles His research interests include fuzzy aggregation functions, and handling of uncertainty.

Marisol Gómez has a degree in Mathematics (University of Salamanca)) and a Ph.D. in Mathematics (Public University of Navarra). She is co-author of more than 60 publications in books, magazines and international conferences. Her research interests include the applications of linear and abstract Algebra to the analysis of biomedical signals.

Javier Andreu-Perez is Senior Lecturer at the Centre for Computational Intelligence, University of Essex, UK. His main interests are machine/deep learning, and human-centred artificial intelligence, and cognitive neuroscience. He received his Ph.D. from Lancaster University (2012), UK, and held positions as a postdoctoral researcher at Imperial College London, UK.

Mukesh Prasad is a Senior Lecturer at the School of Computer Science in the Faculty of Engineering and IT, University of Technology Sydney (UTS), Australia. His main interest are machine learning, artificial intelligence, internet of things, big data, computer vision, brain–computer interface, and evolutionary computation. He received his Ph.D. (2015) from National Chiao Tung University in Taiwan.

Humberto Bustince is currently a Full Professor with the Department of Statistics, Computer Science and Mathematics, Public University of Navarra and honorary professor at the University of Nottingham. He is the author of more than 200 published original articles His research interests include Deep learning, fuzzy logic or aggregation functions.