Applied Soft Computing 67 (2018) 232–244

Contents lists available at ScienceDirect

Applied Soft Computing

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

Evaluating spiking neural models in the classification of motor imagery EEG signals using short calibration sessions

R. Salazar-Varas [a,1], Roberto A. Vazquez [b,*,1]

[a] Digital Signal Processing Group - Facultad de Ingeniería – Universidad La Salle, Benjamin Franklin 45, Col. Condesa CP 06140, Ciudad de México, México
[b] Intelligent Systems Group - Facultad de Ingeniería – Universidad La Salle, Benjamin Franklin 45, Col. Condesa CP 06140, Ciudad de México, México

ARTICLE INFO

Article history:
Received 16 December 2016
Received in revised form 28 January 2018
Accepted 26 February 2018
Available online 6 March 2018

Keywords:
EEG classification
Izhikevich spiking neuron model
Pattern recognition
Coherence
Particle swarm optimization

ABSTRACT

Since the emergence of brain computer interface (BCI), several methods have been applied to associate an electroencephalographic (EEG) recording with a specific mental task. Particularly, in the classification stage, several techniques such as linear Fisher discriminant (LD), feed-forward artificial neural networks (FNN) and radial basis function neural networks (RBF) have been applied successfully with BCI applications. However, in BCI applications, there is a challenge related to avoid long and tedious calibration session for users, this implies that the classification techniques used during the classification stage, have to be trained with a reduced number of EEG recordings. However, most of the classification techniques require several samples to learn accurately an association with a particular mental task. Since the spiking neural models (SNM) have shown their robustness in pattern recognition problems, this paper is focused on demonstrating that they are potential alternatives to classify EEG recordings when they are trained with a reduced number of data samples. To do that, we computed the coherence from a subset of three electrodes to obtain the feature vector of each EEG recording. Then, this information was classified using the SNM. In order to evaluate the robustness, the SNM was trained varying the number of samples. Furthermore, based on the performance and the confidence interval achieved in the classification, we developed two indexes to evaluate and compare the SNM against LD, FNN and RBF. The experimental results over the IIIa, IVa and V data sets from BCI International Competition III, suggest that the SNM are the best option to avoid long calibration sessions.

1. Introduction

A brain computer interface (BCI) is a communication system which enables a subject to control a device only using the brain activity, i.e. without the use of muscles nor peripheral nerves [1]. Then, a correct relationship between the brain activity recorded and a command control must be established to achieve the control of the device.

A generic BCI scheme consists of the next stages: acquisition (in BCI applications, the electroencephalography (EEG) is the most widely used recording technique), pre-processing, feature extraction and classification. Since the emergence of BCI, several processing methods have been proposed focusing on the different stages [2–4]. However, there are some challenges, which must be addressed to achieve real-life BCI applications: select a reduced electrode set, avoid long calibration sessions reducing the number of recordings for training a classifier, increase the information transfer rate and BCI illiteracy [5–7].

To reach an acceptable classification rate in a BCI application, it is necessary to train the classifier with a huge amount of recordings. Collect this amount of recordings implies long calibration sessions for the users, which may result tedious for them. Therefore, during the classification stage, it is desirable to employ a classifier able to learn with a reduced number of recordings in order to avoid long calibration sessions.

One alternative to reduce the number of recordings involves to reduce the dimensionality of the feature vector to satisfy that the number of features be less than the number of recordings. For example in [7–9], the authors described an approach to select a reduced electrode set. However, this solution might not be always practicable, since for some subjects several features to differentiate between two or more mental tasks are necessary. So, a more feasible solution could be obtained if it is used a robust classifier capable to learn with few recordings.

* Corresponding author.
E-mail addresses: rocio.salazar@ulsa.mx (R. Salazar-Varas), ravem@lasallistas.org.mx (R.A. Vazquez).
[1] These authors contributed equally to this work.

https://doi.org/10.1016/j.asoc.2018.02.054
1568-4946/

Artificial neural networks (ANN) have been applied to several pattern recognition problems, including EEG classification for diagnosis of alcoholism [10], detection of epileptic seizure [11] and others [12]. Among the most popular models used in motor imagery EEG classification, we could mention feed-forward neural networks (FNN) and radial basis function neural networks (RBF) [13,14]. However, it is well known that this type of classifier requires a large training data set to get an acceptable accuracy.

Recently, a new type of ANN based on spiking neural models (SNM) has attracted the attention of the ANN scientific community. SNM are considered as the third generation of neural models [15]. They are mathematical abstractions that mimic the behavior of biological neurons in terms of a differential equation. These models are stimulated during a period of time with an input current and at the output, they generate a spike train, similar to real neurons. Although these models are used in the computational neuroscience field to model cortical regions from the brain [16–18], recently they have been applied to solve pattern recognition problems. According to the results reported in [19–25], these models have shown advantages against classical neural networks. In that sense, SNM can be seen as an alternative method to classify patterns obtained from a wide range of problems.

In [26], the authors developed a spiking neural network for epilepsy and epileptic seizure detection using EEG. Although the results were encouraged, the spiking neural network required a large training data set to get an acceptable performance.

In [27,28], the authors described a methodology for recognizing EEG spatio-temporal data using a spiking neural network called NeuCube. The proposed method was useful to predict the response to treatment and dose-related drug effect using EEG data collected from subjects with opiate addiction, patients undertaking methadone maintenance treatment, and non-drug users.

In [29], the authors also proposed a spiking neural network for classifying continuous EEG recordings using wavelet transformation and a network of leaky-integrate-and-fire (LIF) neurons as nodes in a multi-layered structure.

Although spiking neural networks have been applied to classify EEG signals obtained from several mental tasks, these approaches require many samples during the training phase to get a high performance. Based on the results obtained from these previous researches, it is still necessary to evaluate different SNM or learning strategies to determine if they could be considered as robust classifiers to train BCI systems using a reduced number of recordings.

Therefore, the goal of this research is to demonstrate that an SNM is a potential alternative to classify EEG signals when the number of recordings is reduced to avoid long and tedious calibration sessions.

To achieve this goal, we proposed a methodology that integrates the coherence and SNM to classify motor imaginary EEG signals under short calibration sessions. Based on [30], we computed the coherence into a subset of three electrodes to get the feature vector from the EEG signals. Since the coherence is highly sensitive to the frequency at which it is computed, in this paper the coherence was not computed in a specific frequency but in a range of frequencies. After that, based on [19,20], we trained an SNM using a strategy based on particle swarm optimization (PSO). Finally, to evaluate the robustness of the SNM, we developed two indexes based on the performance and the confidences intervals: performance-confidence index (*pci*) and stability index (*si*). In addition, the performance and robustness of the SNM, trained with a different number of samples, was compared against other popular classifiers (used in EEG classification) such as linear Fisher discriminant (LD), feed-forward artificial neural networks (FNN) and radial basis function neural networks (RBF) using the IIIa, IVa and V data set from BCI International Competition III.

This paper is organized as follows: In Section 2, the proposed methodology is exposed, in Section 3 results and discussions are shown and Section 4 presents the conclusions from this work.

2. Methods

As it was mentioned in the Introduction, a generic BCI scheme consists of the next stages: acquisition, pre-processing, feature extraction and classification. The brain activity is recorded during the *acquisition* stage. Next, in the *pre-processing* stage, the EEG signal is enhanced through filtering and amplification. In the *feature extraction*, the most representative information is extracted from the EEG signal to build a feature vector. Finally, in the *classification* stage, the recorded signal is associated with a mental state in order to emit a command control [1].

In this paper, the feature extraction stage is approached through the coherence, and the classification is achieved applying a spiking neural model (SNM). So, in this section both methods are briefly exposed. For more detailed information see [30,19,20].

2.1. Feature vector

Given an EEG data set in a $M \times N \times T$ matrix, where the total number of electrodes is expressed by M, N indicates the total number of data points and T is the total number of trials, the goal is to select a sub set of L electrodes, with $L \ll M$, based on the coherence values. The coherence is a metric that expresses, in the frequency domain, the correlation between two signals. Then, given two EEG signals recorded from electrodes j and k, expressed as $x_j(n)$ and $x_k(n)$, the coherence between them is given by [31]

$$\gamma_{j,k}^2(f) = \frac{|P_{jk}(f)|^2}{P_j(f)P_k(f)}, \tag{1}$$

where f indicates the frequency at which the coherence is computed, $P_{jk}(f)$ is the cross spectral density of $x_j(n)$ and $x_k(n)$, and $P_j(f)$ and $P_k(f)$ are the individual signal's auto-spectral densities.

Then, for each one of $\left(M//L\right)$ subsets, the coherence between all $D = \left(L//2\right)$ possible combinations from the sub set of L electrodes is computed as was expressed in (1), obtaining D coherence values to form a D-dimensional feature vector $\boldsymbol{y}$. To assure that these coherence values express true connectivity, a significance level $\eta_{j,k}$ is established according [32].

The coherence computation is carried out in an iterative process for all T trials for each $i = 1, 2, \ldots, I$ class. If in a subset of L electrodes all D coherence values are significant, i.e.

$$\gamma_{j,k}^2(f) > \eta_{j,k} \quad \forall D = \binom{L}{2} \text{ combinations} \tag{2}$$

the electrode set and the coherence values are stored.

Now take importance the coherence values that allow us discriminate different mental tasks. Considering the case of I different classes, each of one containing T' trials, where $T' \leq T$ are the trials satisfying the condition in (2) and $\{\gamma_{j,k}^2(f)\}_{i,t}$ denotes the coherence between signals for a given t trial and i class, the mean class coherence is

$$\mu_i = \frac{1}{T_i'}\sum_{t=1}^{T_i'}\{\gamma_{j,k}^2(f)\}_{i,t}, \tag{3}$$

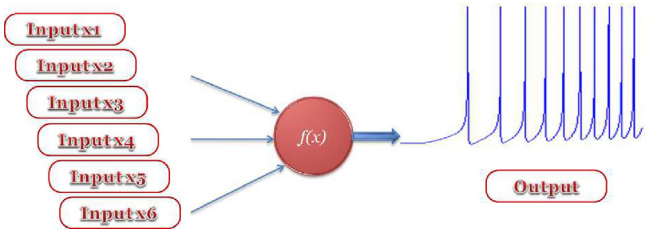


Fig. 1. Scheme of a spiking neuron.

In order to determine if the coherence values allow us to discriminate among different mental tasks and regarding (3), we can apply the following hypothesis test

$$H_o : \mu_1 = \mu_2 = \cdots = \mu_I$$
$$H_a : \text{any negation of } \ H_o. \tag{4}$$

A variety of methods can be applied to reject the null-hypothesis. Independently of that, the L sensors are selected based on the p-values of the statistical test in order to select the sensors which provide the maximal difference among the coherence values for the I different classes. If $p_{j,k}$ defines the p-value corresponding to the coherence between EEG signals recorded from electrodes j and k, the most likely differences between coherence values occurs when

$$p_{j,k} \ll \alpha. \tag{5}$$

where α is the significance level in the hypothesis test.

Finally, the feature vector $\boldsymbol{y}_{\mathrm{OPT}}$ is built by the coherences values computed from the selected L electrodes. These coherence values attain the condition in (2) and the p values from the hypothesis test (4) are a lot less than the significance level (5) in most of its D dimensions.

2.2. Classifying with spiking neuron models

Spiking neural models were proposed since the beginning of XX century [33,15]. Basically, these models are capable of modeling different neurodynamic properties of neurons. Among the most popular, we could mention the integrate-and-fire, Hodgkin-Huxley, resonate-and-fire, FitzHugh-Nagumo and Izhikevich models [34–38].

Due to its simplicity and versatility, in this paper we are going to use the Izhikevich neuron model which is defined with a differential equation described in (6):

$$\begin{aligned} C\dot{v} &= k(v - v_r)(v - v_t) - u + I \qquad \text{if} v \geq v_{peak} \text{then} \\ \dot{u} &= a\{b(v - v_r) - u\} \qquad\quad v \longleftarrow c, u \longleftarrow u + d \end{aligned} \tag{6}$$

where v represents the membrane potential and u is the recovery current. The membrane capacitance is represented with C, the resting membrane potential with v_r, the instantaneous threshold potential with v_t, and the spike cutoff value with v_{peak}. Finally, a is the recovery time constant, b force to u behaves as an amplifying (when $b < 0$) or as a resonant (when $b > 0$), c define the voltage reset value, and d represents the total amount of currents activated during the spike.

In this research, we are going to use the parameters that produce regular spiking patterns, see Fig. 1.

Based on the work described in [19,20], the next behavior must be reached to apply successfully a SNM in a pattern recognition task: "patterns from the same class produce similar firing rates in the output of the spiking neuron and patterns from other classes produce firing rates different enough to discriminate among the classes". To achieve that, in [19,20] the authors proposed a methodology that can be separated in four stages, see Fig. 2: data

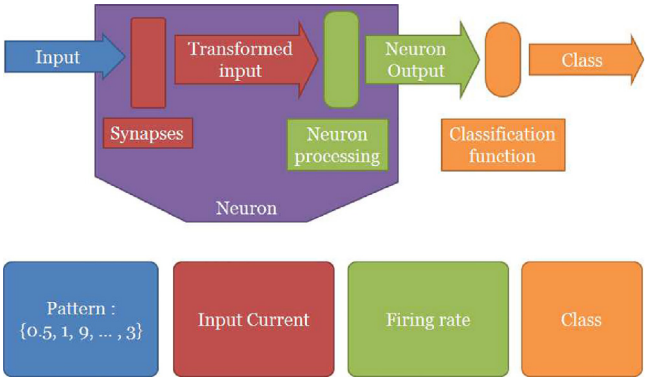


Fig. 2. Stages of the classification using SNM.

adquisition, input current computation, firing rate computation, classification.

Given an input pattern (feature vector) $\mathbf{x}^i \in \mathbb{R}^n$, which corresponds to the presynaptic potential of different receptive fields, it is transformed into an input current by means of the synaptic weights' neuron model $\mathbf{w}^i \in \mathbb{R}^n$ by applying (7)

$$I = \mathbf{x} \cdot \mathbf{w}. \tag{7}$$

Then, this input current is used to stimulate the spiking neuron model during T ms to obtain the spike train and compute the firing rate as described in (8)

$$fr = \frac{N_{sp}}{T}, \tag{8}$$

where N_{sp} is the number of spikes that occur within the time window of length T. After that, the average firing rate of each class $\mathbf{AFR} \in \mathbb{R}^K$ is computed.

Finally, to classify an unknown input pattern $\tilde{\mathrm{x}}$, it is necessary to stimulate the spiking neuron using the current obtained with $\tilde{\mathrm{x}}$, compute its firing rate fr and compare it against the average firing rate of each class by mean of (9)

$$cl = \underset{k=1}{\overset{K}{\arg\min}}(|AFR_k - fr|) \tag{9}$$

The authors suggested that the learning process could be conducted by an optimization strategy for adjusting the synaptic weights of the spiking model to generate different firing rate for each class k. Among these strategies, we could mention particle swarm optimization (PSO) [39], artificial bee colony (ABC) [19], cuckoo search (CS) [40], differential evolution (DE) [21] and genetic algorithms (GA) [20]. According to [41], these learning strategies provide similar results among them during a pattern recognition task. This fact gives us the possibility of choosing any of these strategies taking into account the easiest for implementation, fast convergence and low complexity.

2.3. Training spiking neuron models using PSO

Different evolutionary computation as well as swarm intelligence techniques have been applied as a learning strategy for training artificial neural networks [42,43]. Particle swarm optimization (PSO) is a technique widely used for optimizing non-linear and non-differentiable continuous space functions with a great accuracy. This technique has been widely applied for training and designing ANN [39,44–46]. This algorithm is based on a set of SN particles (population) $\mathbf{s}_i$ that represent solutions (positions) to the optimization problems. During each iteration t of the optimiza-

tion process, each particle updates its current position based on a velocity function according to (10):

$$\mathbf{v}_i(t+1) = \omega \mathbf{v}_i(t) + c_1 r_1(\mathbf{p}_i(t) - \mathbf{s}_i(t)) + c_2 r_2(\mathbf{p}_g(t) - \mathbf{s}_i(t)) \quad (10)$$

where $\mathbf{p}_g$ is a memory that represent social component (best solution of the population), $\mathbf{p}_i$ is the memory of each particle that represent the cognitive component (best solution of the particle), an inertia weight is represented with ω that changes each iteration in the range [1, 0), r_1 and r_2 are uniformly distributed random numbers $U(0, 1)$ called the craziness, and c_1 and c_2 represents the acceleration coefficients. The boundaries of this function are limited according to $[v_{\min}, v_{\max}]$.

Once computed the velocity of each particle i, they update its position according to (11):

$$\mathbf{s}_i(t+1) = \mathbf{s}_i(t) + \mathbf{v}_i(t+1) \quad (11)$$

To asses the quality and know the aptitude of each possible solution, the particles are evaluated by means of a fitness function.

For more details of this algorithms, the reader could refers to [47]. The basic PSO algorithm is described in Algorithm 1.

Algorithm 1. PSO pseudo-code.

```
t = 0
Initialize the swarm population of solutions s_i(t) ∀ i, i = 1, ..., SN
Evaluate the population f(s_i(t)) ∀ i, i = 1, ..., SN
Initialize best local position p_i = f(s_i(t)) ∀ i, i = 1, ..., SN
Initialize best global position p_g = min(f(s_i(t)) ∀ i, i = 1, ..., SN)
for t = 0 to MAXITER do
  for i = 1 to SN do
    Compute velocity v_i(t+1)
    Compute new position s_i(t+1)
    Evaluate the fitness of the particle f(s_i(t+1))
    if f(s_i(t+1)) < p_i then
      Update best local position p_i = f(s_i(t+1))
    endif
  endfor
  if min(f(s_i(t+1)) ∀ i, i = 1, ..., SN) < p_g then
    Update best global position p_g = min(f(s_i(t+1)) ∀ i, i = 1, ..., SN)
  end if
  t = t + 1
end for
```

In the context of the SNM and the EEG classification, each particle from the swarm corresponds to a set of synaptic weights, and the size of particle is determined in terms of the number of features extracted from the EEG recordings that stimulate the SNM. The learning process consists in computing the value of the synaptic weights to maximize the performance of the SNM during the EEG classification task. In order to guide the learning process the fitness function described in (12) was used:

$$f(\mathbf{w}, \mathbf{X}, \mathbf{d}) = 1 - \frac{\sum_{i=1}^{p} c(\mathbf{w}, \mathbf{x}_i, d_i)}{p} \quad (12)$$

where $\mathbf{w}$ represents the synaptic weights of the Izhikevich neuron model, $\mathbf{X}$ represents the set of input patterns, $\mathbf{d}$ represents the set of desired patterns associated to each pattern, p represents the number of patterns, and $c(\mathbf{w}, \mathbf{x}, d)$ represents a function that computes if the pattern was classified correctly and is defined as:

$$c(\mathbf{w}, \mathbf{x}, d) = \begin{cases} 1, & |d - y(\mathbf{w}, \mathbf{x})| = 0 \\ 0, & |d - y(\mathbf{w}, \mathbf{x})| \neq 0 \end{cases} \quad (13)$$

where $\mathbf{x}$ is the input pattern, d is the desired class, y corresponds to the class detected with the SNM, and $\mathbf{w}$ are the synaptic weights of the SNM.

3. Results and discussion

In order to evaluate the performance of the proposed methodology, we employed three different data sets from BCI international competition III, all of them related to imaginary movement:

- Data set IIIa. This data set contains the EEG recording from three subjects, labeled as k3b, k6b and l1b. The tasks asked to the subjects were imagery left hand, right hand, foot or tongue movements. For subject k3b, there are 90 trials for each mental task, for the other two subjects, there are 60 trials for each task. The register was made using 32 electrodes. In this work only left hand and foot imagery movements were used.
- Data set IVa [48]. This data set contains the EEG recording from five subjects labeled as *aa, al, av, aw, ay*. Each subject was asked to imagine two different movements: right foot and right hand. For each mental task, 140 trials were registered using 118 electrodes.
- Data set V [49]. This data set contains the EEG recording from three subjects, labeled as S1, S2 and S3. Each subject was asked to perform three different tasks: imagination of repetitive self-paced left hand movements, imagination of repetitive self-paced right hand movements, and generation of words beginning with the same random letter (in this work, only movement tasks were used). Four sessions were performed, each lasting 4 min in which the subject performed the task about 15 s accordingly with the operator's request. The acquisition was achieved using 32 electrodes. The 15s were segmented in 2s overlapped 1s. In this way, approximately 300 trials were generated for each mental task.

To evaluate the robustness of the SNM, when it is trained with a reduced number of data samples (to avoid long calibration session for the subjects), the classification stage was performed varying the number of samples that composed the training data set from 50% to 10% of the total recordings. The testing data set was always conformed by 50% of the entire recordings. In order to avoid the data in training or the evaluation set influence the classification, a random sub-sampling validation test of 30 iterations was done.

All data sets were preprocessed applying a Butterworth band pass filter of fourth order with cutoff frequencies 1 and 100 Hz. After that, the feature vector $\boldsymbol{y}$ was built in terms of the coherence between the EEG signals. We used $L = 3$ electrodes for each subject, which in the case of the data set IV were obtained from a subset of $M = 16$ electrodes (F7, F8, Fp1, Fp2, F3, Fz, F4, C3, Cz, C4, P3, Pz, P4, O1, Oz, O2). In the case of data sets IIIa and V the $L = 3$ electrodes were selected from the original electrodes set used in the recording. The selection was carried out as described in [30] and summarized in Section 2.1. The significance level α in the hypothesis test described in (4) was set to 0.5.

Instead of computing the coherence for a specific frequency as reported in [30], we computed the coherence within a range of frequencies since the coherence value is highly sensitive to the frequency in which it is computed. The selected frequency range was from 8 Hz to 18 Hz because this is the range in which the (de)synchronization due to the movement tasks occurs. So the feature vector is expressed as $\boldsymbol{y} = [\gamma^2(f_1), \gamma^2(f_2), \ldots, \gamma^2(f_N)]$, where $\gamma^2(f) = [\gamma^2_{1,2}(f), \gamma^2_{1,3}(f), \gamma^2_{2,3}(f)]$, $N = 11$, $f_1 = 8$ Hz and $f_{11} = 18$ Hz. The selected electrodes are listed in Table 1.

In order to train the Izhikevich neural model, we set its parameters as $C = 100$, $v_r = -60$, $v_t = -40$, $v_{peak} = 35$, $k = 0.7$, $a = 0.03$, $b = -2$, $c = -50$, and $d = 100$. To solve the differential equation's model, we set the parameters of the Euler method as $dt = 1$ and simulation time as $T = 1000$. For the training algorithm (PSO), the population size was set to $NP = 40$, the maximum number of generations to $MAXGEN = 1000$, the velocity to $VMAX = 4$ and $VMIN = -4$, the crazyness to $c_1 = 2$ and $c_2 = 2$, ω was varied in the range [0.95–0.4] through

Table 1
Electrodes selected for each subject to build the feature vector [30]. * For this data set, the electrodes name is supposed by the authors based on the scheme reported in the data set description.

Data set	Subject	Electrodes
Data	*aa*	Fz, C3, Cz
set	*al*	C3, Cz, P3
IVa	*av*	C3, C4, P3
	aw	F3, C4, P3
	ay	C3, P3, Pz
Data	*S1*	C3, C4, Cz
set	*S2*	C3, C4, Cz
Va	*S3*	C3, C4, Cz
Data	*kb3**	FCz, C3, C4
set	*kb6**	Cz, P2, P3
IIIa	*lb1**	C3, Cz, Pz

the generations and the boundaries were set to $XMAX = 10$ and $XMIN = -10$.

In addition, the performance of the SNM was compared against a linear discriminant classifier (LD) and two popular artificial neural networks: feed-forward neural network (FNN) and radial basis function neural network (RBF).

It was demonstrated in [50] that winner-take-all strategies are computationally powerful compared with standard neural models with threshold and sigmoidal gates, furthermore, instead of using more than a single layers of nonlinear neurons (sigmoidal) to represent complex functions it is enough employing a winner-take-all strategies as nonlinear unit. In order to obtain the best accuracy with the FNN, we evaluate, compare and design different architectures (SNB, SFNNB, FNN1B, FNN2B, FNN1 and FNN2) composed of different number of layers, number of neurons and type of neuron in the output layer. These architectures were trained using a Levenberg-Marquardt learning algorithm (during 5000 epochs) with a learning rate of 0.1 and goal error of 0.

The architecture SNB was composed of one single sigmoidal neuron where outputs greater than 0.5 were assigned to class 1, otherwise class 0.

The architecture SFNNB is composed of a hidden layer with 2 to 10 sigmoidal neurons and an output layer with a sigmoidal neuron where outputs greater than 0.5 were assigned to class 1, otherwise class 0. For each subject and dataset, we maximize the accuracy determining the best number of neurons for the hidden layer.

For the architectures FNN1B, FNN2B, FNN1 and FNN2, the number of neurons that composes the hidden layers *hn* was computed according to Eq. (14)

$$hn = (n_f + n_c) \tag{14}$$

in which n_f is determined in terms of the number of features that contains the input pattern and n_c is determined with the number of classes of the dataset.

For the architecture with two hidden layers, the neurons were distributed as follow: $hn_1 = 0.6hn$ and $hn_2 = 0.4hn$, respectively.

For the architectures FNN1B, FNN2B, sigmoidal neurons were used in the hidden layers and one sigmoidal neuron was used in the output layer. To determine the class to which an input pattern belongs, we establish a threshold value (*th*) to indicate that outputs greater than *th* belongs to class 1, otherwise class 0. To select the *th*, we tested different values in the range [0.1–0.9] using an increment of 0.1, and then selected the best *th* for each subject and configuration.

For the architectures FNN1, FNN2, hyperbolic tangential neurons were used in hidden and output layers, and the number of neurons in the output layer *on* was determined in terms of the number of classes of the data set n_c. To determine the class to which an input pattern belongs, we applied a winner-take-all approach, where the neuron with the highest output value is set to 1 and the others to 0; if neuron one is set to 1, it indicates that the pattern belongs to class 1, if neuron two is set to 1, indicates that pattern belongs to class 2.

Table 2
Average performance achieved with different FNN architectures.

Data training [%]	Classifier	Data set IVa	Data set V	Data set IIIa
50	SNB	49.8 ± 2.5*	49.8 ± 1.9*	49.7 ± 4.9*
	SFNNB	50.8 ± 2.9*	50.4 ± 1.9*	51.2 ± 4.5*
	FNN1B	65.7 ± 7.2*	55.1 ± 5.5*	62.8 ± 7.6*
	FNN2B	65.3 ± 7.1*	57.7 ± 4.7*	61.7 ± 8.4*
	FNN1	70.5 ± 4.8	62.4 ± 3.7	**66.0 ± 5.1**
	FNN2	**71.1 ± 4.4**	**63.5 ± 3.1**	64.8 ± 6.1
40	SNB	49.9 ± 2.8*	49.9 ± 2.0*	50.1 ± 4.7*
	SFNNB	50.9 ± 2.8*	50.6 ± 2.1*	50.9 ± 4.2*
	FNN1B	64.6 ± 8.3*	55.8 ± 4.7*	62.2 ± 6.3*
	FNN2B	65.9 ± 7.1*	58.3 ± 5.1*	63.3 ± 7.3
	FNN1	69.8 ± 5.0	**63.6 ± 2.7**	**64.8 ± 6.7**
	FNN2	**70.3 ± 5.0**	62.8 ± 2.7	64.5 ± 5.3
30	SNB	49.6 ± 2.6*	49.4 ± 2.2*	49.2 ± 3.9*
	SFNNB	50.7 ± 2.9*	50.5 ± 2.8*	51.2 ± 4.0*
	FNN1B	63.8 ± 7.7*	58.2 ± 3.9*	59.6 ± 8.9*
	FNN2B	65.0 ± 6.8*	58.4 ± 4.7*	61.2 ± 8.8
	FNN1	68.0 ± 5.3	62.2 ± 3.4	63.6 ± 6.3
	FNN2	**69.5 ± 5.3**	**62.7 ± 3.0**	**64.1 ± 7.0**
20	SNB	49.9 ± 2.7*	50.2 ± 2.1*	50.4 ± 4.5*
	SFNNB	50.8 ± 2.7*	50.5 ± 1.8*	50.8 ± 5.2*
	FNN1B	64.1 ± 6.6*	56.0 ± 5.9*	61.5 ± 8.0
	FNN2B	63.1 ± 8.3*	56.8 ± 5.0*	61.6 ± 9.8
	FNN1	66.4 ± 5.3	59.1 ± 3.9	57.8 ± 7.8
	FNN2	**68.1 ± 5.6**	**59.8 ± 3.6**	**61.7 ± 7.1**
10	SNB	49.9 ± 2.7*	50.2 ± 1.8*	49.8 ± 4.8*
	SFNNB	50.7 ± 2.6*	50.5 ± 2.1*	50.8 ± 4.2*
	FNN1B	61.1 ± 7.7*	55.5 ± 5.6*	58.7 ± 7.9
	FNN2B	61.0 ± 7.8*	55.9 ± 5.1*	59.4 ± 7.3
	FNN1	**65.3 ± 7.1**	57.7 ± 4.6	**61.7 ± 8.3**
	FNN2	63.8 ± 7.3*	**58.4 ± 4.7**	58.9 ± 8.6

Finally, a random sub-sampling validation test of 30 iterations was done to compute the average performance of the different architectures.

In order to determine which configuration was the most suitable to recognize the EEG signals and assess if there are significant differences among the accuracies achieved with the different ANN architectures, we applied an ANOVA test. Since the ANOVA test showed that there are significant differences, a multiple comparison test was performed. In Table 2, we present the average accuracy for all subjects for each data set. The maximum accuracy is remarked with bold font and the accuracies that present significant difference based on the ANOVA and multiple comparison tests are followed by an asterisk.

After evaluating the experimental results, the ANOVA test corroborates that the winner-take-all strategy in the output layer provides significantly different results compared with those obtained with the other architectures. Based on these results, we selected the FNN based on the winner-take-all strategy for comparing the results against those obtained with the proposed methodology.

We adopted two different approaches for training the RBF. In the first approach (RBF1), the hidden layer *hn* was composed of 2 to 10 Gaussian neurons where the center and the width were selected using a kmeans clustering algorithm. The output layer was composed of one linear neuron where outputs greater than 0.5 were assigned to class 1, otherwise class 0. For each subject and dataset, we maximize the accuracy determining the best number of neurons for the hidden layer.

The second approach (RBF2) used an incremental architecture. When the learning stage begins, the hidden layer has no neurons

Table 3
Average performance achieved with the two approaches for designing RBF.

Data Training [%]	Classifier	Data set IVa	Data set V	Data set IIIa
50	RBF1	56.0 ± 4.2*	55.6 ± 3.6*	53.6 ± 5.5*
	RBF2	70.8 ± 4.9	62.5 ± 2.4	66.1 ± 5.5
40	RBF1	55.6 ± 4.5*	55.3 ± 3.5*	52.7 ± 5.2*
	RBF2	71.5 ± 4.1	61.8 ± 2.4	61.4 ± 6.0
30	RBF1	54.6 ± 4.1*	54.7 ± 3.1*	51.7 ± 4.7*
	RBF2	72.3 ± 4.0	62.3 ± 2.6	61.9 ± 5.9
20	RBF1	54.0 ± 4.7*	54.2 ± 3.9*	52.4 ± 5.6*
	RBF2	70.4 ± 4.8	61.8 ± 2.8	62.9 ± 5.6
10	RBF1	53.0 ± 4.1*	52.7 ± 3.6*	51.4 ± 5.0*
	RBF2	65.9 ± 6.6	58.8 ± 3.6	59.8 ± 8.2

and *nc* linear neurons in the output layer. Neurons in the hidden layer are added each iteration during the learning process until a number of neurons or specific error is reached. In order to determine the maximum number of neurons and the width of the Gaussian, we tested different values for the width in the range [0.1–2.0] using an increment of 0.1 and a maximum number of neurons computed as $mnn(hma) = (((n_p - n_c)/20) \times hma) + n_c$ where n_p represents the number of patterns, n_c the number of classes and *hma* is a value from 1 to 20. The goal error was set to 0.01. For each subject and dataset, we maximize the accuracy determining the best number of neurons for the hidden layer and best width. As equal as FNN1 and FNN2, to determine the class to which an input pattern belongs, we applied a winner-take-all approach, where the neuron with the highest output value is set to 1 and the others to 0.

Finally, a random subsampling validation test of 30 iterations was done to compute the average performance of the RBF1 and RBF2.

In order to evaluate if there are significant differences between the results achieved with RBF1 and those obtained with RBF2, we applied a hypothesis test with a significance level $p = 0.01$. Table 3 shows the average performance for all subjects for all data sets. As equal as in Table 2, the cases with significant differences are indicated with an asterisk. From these experiments, we observed that the RBF2 provides the best results which are significantly different from those achieved with RBF1. Based on these results, we selected the RBF2 for comparing the results against those obtained with the proposed methodology.

Table 5
Performance achieved for data set V.

Data Training [%]	Classifier	S1	S2	S3	Average
50	SNM	72.6 ± 2.2	59.4 ± 2.8	56.6 ± 2.8	62.9 ± 2.6*
	LDA	75.6 ± 1.4	60.9 ± 2.6	57.61 ± 2.8	**64.7 ± 2.2**
	FNN1	71.1 ± 3.9	58.4 ± 3.7	57.6 ± 3.6	62.4 ± 3.7*
	FNN2	73.3 ± 2.7	58.0 ± 3.8	59.3 ± 2.8	63.5 ± 3.1
	RBF2	72.3 ± 1.9	56.4 ± 3.3	58.7 ± 2.1	62.5 ± 2.4*
40	SNM	71.9 ± 2.1	60.0 ± 2.9	56.6 ± 2.2	62.8 ± 2.4*
	LDA	71.7 ± 1.6	61.3 ± 2.7	57.4 ± 2.2	**63.5 ± 2.1**
	FNN1	72.5 ± 2.0	58.8 ± 3.0	59.4 ± 3.2	63.6 ± 2.7
	FNN2	71.9 ± 1.9	58.7 ± 2.8	57.8 ± 3.5	62.8 ± 2.7*
	RBF2	70.5 ± 1.8	57.0 ± 2.4	58.0 ± 3.0	61.8 ± 2.4*
30	SNM	71.6 ± 2.5	60.5 ± 2.5	56.2 ± 3.4	62.8 ± 2.8
	LDA	71.8 ± 1.6	61.1 ± 1.9	56.5 ± 3.4	**63.1 ± 2.3**
	FNN1	71.3 ± 2.9	57.2 ± 4.0	58.0 ± 3.4	62.2 ± 3.4
	FNN2	72.0 ± 2.5	58.6 ± 3.3	57.6 ± 3.1	62.7 ± 3.0
	RBF2	72.4 ± 2.0	55.7 ± 3.5	58.8 ± 2.3	62.3 ± 2.6
20	SNM	70.3 ± 2.7	59.2 ± 2.8	54.7 ± 3.0	61.4 ± 2.8
	LDA	71.4 ± 2.2	56.2 ± 2.5	56.1 ± 3.0	61.3 ± 2.6
	FNN1	71.1 ± 2.4	56.2 ± 3.3	56.0 ± 3.3	61.1 ± 3.0
	FNN2	69.4 ± 3.0	54.5 ± 4.9	55.6 ± 3.1	59.8 ± 3.6*
	RBF2	71.5 ± 2.2	57.0 ± 3.2	56.8 ± 3.1	**61.8 ± 2.8**
10	SNM	69.5 ± 2.8	57.0 ± 4.2	54.5 ± 2.7	**60.3 ± 3.2**
	LDA	65.8 ± 4.0	53.3 ± 2.7	54.0 ± 2.7	57.7 ± 3.1 *
	FNN1	66.8 ± 4.2	55.8 ± 4.7	54.8 ± 2.8	59.1 ± 3.9
	FNN2	67.6 ± 4.0	53.9 ± 5.3	53.8 ± 4.9	58.4 ± 4.7*
	RBF2	68.9 ± 2.9	53.4 ± 4.2	54.2 ± 3.6	58.8 ± 3.6

Tables 4–6 show the average performance value and standard deviation for all subjects achieved with the SNM as well as a comparison against other classifiers.

Table 4 shows the average performance obtained with the classifiers using data set IVa (which contains 140 recordings for each class). The average performance obtained with the SNM for all sub-

Table 4
Performance achieved for data set IVa

Data Training [%]	Classifier	aa	al	av	aw	ay	Average
50	SNM	68.7 ± 3.7	90.7 ± 2.4	60.5 ± 3.5	72.9 ± 3.4	72.3 ± 1.6	**73.0 ± 2.9**
	LDA	67.2 ± 3.3	84.1 ± 3.3	55.9 ± 3.6	69.2 ± 3.4	67.4 ± 3.5	68.8 ± 3.4*
	FNN1	65.2 ± 6.7	89.4 ± 2.1	57.9 ± 4.1	69.9 ± 3.8	70.0 ± 7.3	70.5 ± 4.8*
	FNN2	63.0 ± 6.1	90.1 ± 3.0	59.3 ± 5.1	72.5 ± 5.0	70.4 ± 2.9	71.1 ± 4.4*
	RBF2	58.0 ± 7.7	90.7 ± 2.2	51.6 ± 7.4	71.1 ± 3.7	82.5 ± 3.5	70.8 ± 4.9*
40	SNM	67.1 ± 4.2	89.6 ± 4.2	60.4 ± 4.5	71.8 ± 3.9	71.5 ± 2.4	**72.1 ± 3.8**
	LDA	65.1 ± 3.9	83.6 ± 3.6	55.5 ± 3.6	68.3 ± 3.5	66.2 ± 3.5	67.7 ± 3.6*
	FNN1	63.0 ± 7.6	88.7 ± 2.9	58.7 ± 3.5	68.8 ± 5.2	69.7 ± 5.5	69.8 ± 5.0*
	FNN2	62.7 ± 7.1	89.5 ± 3.4	59.3 ± 4.0	70.3 ± 5.8	69.6 ± 4.8	70.3 ± 5.0
	RBF2	62.3 ± 6.7	89.7 ± 2.4	55.6 ± 4.8	68.2 ± 4.4	81.7 ± 2.1	71.5 ± 4.1
30	SNM	66.0 ± 4.9	88.7 ± 4.3	59.3 ± 4.4	72.3 ± 3.9	71.0 ± 2.4	71.5 ± 4.0
	LDA	62.1 ± 3.8	81.5 ± 4.2	54.8 ± 3.6	66.3 ± 4.1	64.0 ± 4.4	65.7 ± 4.0*
	FNN1	59.9 ± 6.7	87.3 ± 7.5	56.4 ± 4.0	68.4 ± 4.7	67.8 ± 3.5	68.0 ± 5.3*
	FNN2	61.5 ± 6.3	88.9 ± 3.4	58.3 ± 5.2	67.9 ± 7.2	71.1 ± 4.2	69.5 ± 5.3*
	RBF2	64.3 ± 5.0	88.9 ± 2.8	59.5 ± 5.2	69.0 ± 4.5	79.6 ± 2.6	**72.3 ± 4.0**
20	SNM	65.8 ± 4.4	87.2 ± 4.9	60.0 ± 4.3	69.9 ± 3.6	68.9 ± 2.8	**70.4 ± 4.0**
	LDA	58.2 ± 5.0	76.3 ± 5.5	53.1 ± 4.1	63.0 ± 4.5	61.6 ± 4.1	62.4 ± 4.6*
	FNN1	60.4 ± 7.3	87.1 ± 2.6	54.6 ± 5.4	67.9 ± 4.5	69.2 ± 2.7	67.8 ± 4.5*
	FNN2	60.1 ± 7.7	86.9 ± 4.0	57.7 ± 6.6	67.8 ± 4.3	68.2 ± 5.1	68.1 ± 5.6*
	RBF2	61.5 ± 6.3	87.4 ± 4.3	58.8 ± 5.6	70.4 ± 4.4	73.8 ± 3.5	70.3 ± 4.8
10	SNM	61.5 ± 7.4	84.1 ± 4.8	58.3 ± 5.4	67.1 ± 7.4	67.4 ± 3.0	**67.7 ± 5.6**
	LDA	50.4 ± 5.8	49.0 ± 9.3	50.1 ± 4.8	49.5 ± 5.7	50.2 ± 7.3	50.6 ± 6.6*
	FNN1	60.9 ± 6.2	81.0 ± 6.2	56.3 ± 5.5	67.2 ± 6.3	66.5 ± 2.3	66.4 ± 5.3
	FNN2	57.2 ± 6.7	78.8 ± 8.0	53.4 ± 8.7	63.9 ± 8.6	65.9 ± 4.4	63.8 ± 7.3*
	RBF2	53.7 ± 7.0	85.4 ± 6.1	57.2 ± 5.4	64.0 ± 10.9	69.4 ± 3.5	65.9 ± 6.6

Table 6
Performance achieved for data set IIIa.

Data Training [%]	Classifier	S1	S2	S3	Average
50	SNM	85.3 ± 3.6	60.6 ± 4.7	58.3 ± 4.9	**68.1 ± 4.4**
	LDA	79.2 ± 4.4	54.2 ± 7.9	51.7 ± 4.7	61.7 ± 5.6 *
	FNN1	84.9 ± 2.9	58.4 ± 6.5	54.8 ± 5.8	66.0 ± 5.1
	FNN2	85.3 ± 3.2	57.9 ± 7.4	51.2 ± 7.6	64.8 ± 6.1*
	RBF2	85.6 ± 2.8	61.1 ± 5.2	51.6 ± 8.4	66.1 ± 5.5
40	SNM	84.6 ± 4.3	60.6 ± 6.0	58.7 ± 4.8	**68.0 ± 5.0**
	LDA	78.2 ± 4.2	56.1 ± 6.8	48.9 ± 6.2	61.1 ± 5.7*
	FNN1	84.7 ± 5.5	57.7 ± 6.4	52.1 ± 8.1	64.8 ± 6.7*
	FNN2	82.9 ± 4.3	58.7 ± 5.3	52.0 ± 6.2	64.5 ± 5.3*
	RBF2	78.6 ± 4.4	55.3 ± 8.2	50.3 ± 5.6	61.4 ± 6.0*
30	SNM	82.9 ± 3.7	60.0 ± 6.6	58.9 ± 4.8	**67.3 ± 5.0**
	LDA	72.6 ± 4.2	51.4 ± 6.1	50.1 ± 8.0	58.0 ± 6.1*
	FNN1	82.6 ± 4.9	57.3 ± 6.9	50.8 ± 7.1	63.6 ± 6.3*
	FNN2	81.2 ± 7.8	58.1 ± 6.1	53.2 ± 7.1	64.1 ± 7.0*
	RBF2	84.7 ± 4.3	48.1 ± 6.7	52.8 ± 6.7	61.9 ± 5.9*
20	SNM	82.7 ± 4.2	58.2 ± 6.2	57.8 ± 5.7	**66.2 ± 5.4**
	LDA	50.7 ± 9.1	50.7 ± 6.3	51.1 ± 6.7	50.8 ± 7.4*
	FNN1	77.9 ± 9.8	53.3 ± 7.7	50.3 ± 5.5	60.5 ± 7.7*
	FNN2	78.9 ± 9.2	54.2 ± 7.0	52.1 ± 5.0	61.7 ± 7.1*
	RBF2	83.0 ± 4.6	56.7 ± 6.6	49.1 ± 5.5	62.9 ± 5.6*
10	SNM	79.1 ± 7.7	56.6 ± 5.9	55.1 ± 6.3	**63.6 ± 6.6**
	LDA	51.1 ± 11.6	51.1 ± 5.5	47.8 ± 6.6	50.0 ± 7.9*
	FNN1	70.6 ± 11.1	52.5 ± 7.1	50.4 ± 5.3	57.8 ± 7.8*
	FNN2	72.6 ± 11.0	51.4 ± 9.0	52.7 ± 5.9	58.9 ± 8.6*
	RBF2	74.8 ± 13.3	55.5 ± 6.8	49.1 ± 4.5	59.8 ± 8.2*

jects decreased from 73.0% to 67.7% when the number of recordings for training was diminished from 50% (70 EEG recordings for each task) to 10% (14 EEG recordings for each task). The SNM provided the best average accuracy, except when 30% of the recordings were used during training, where the RBF2 provided the best results. Although the average performance of RBF2, FNN1 and FNN2 also was acceptable, this type of neural network requires several neurons to obtain acceptable results. Furthermore, in the critical case, when the classifiers were trained with 10% of the samples, the SNM provided the best results for three of the subjects (*aa*, *av*, *aw*) and the RBF2 achieved the best results for *al* and *ay*. Nonetheless, the principal advantage of the SNM against RBF and FFN is related to the number of neurons required to obtain acceptable results. Finally, in average the LDA (which is commonly used in BCI applications) provided the lowest performance, even when the classifier was trained with an acceptable number of data samples.

Table 5 shows the average accuracy achieved with the classifiers for the data set V which contains 300 recordings (a big amount of recordings compared against the other data sets) for each class. In this set of experiments, we observe that LDA achieved the best accuracy when the 50–30% of the total recordings (150–60 recordings for each class) were used during training stage. The RBF2 achieved the best performance when it was trained with 20% of the recordings. In the worst case, when 10% of data samples (30 recordings for each class) were used to train the classifiers, a decrease in the performance can be observed for most of the classifiers, in this condition, the SNM achieved the best accuracy higher than 60%. Although with this data set the other classifiers provided a better accuracy, the SNM showed its robustness against the other classifiers when the number of samples used to train was drastically reduced.

Table 6 shows the results obtained with data set IIIa, which contains 60 recordings for each class. In average, the SNM provided the best results in all training conditions from 68.1% to 63.6%. Except for subject S1 (when it was trained with 50%, 30% and 20% of the total recordings) and subject S2(when it was trained with 50% of the total recordings) where the RBF2 provided the best results.

Table 7
Results of the ANOVA test of the accuracy values.

Data set	Data training [%]	F values	*p* values
IVa	50	20.3, 117.4, 23.71	7.73E−16, 3.03E−308, 6.67E−56
	40	19.90, 963.39, 13.28	1.45E−15, 2.20E−288, 1.93E−31
	30	50.12, 854.81, 7.69	2.90E−37, 1.06E−272, 5.5E−17
	20	86.47, 751.90, 6.36	4.58E−60, 3.04E−256, 1.74E−13
	10	175.05, 247.60, 16.30	6.97E−105, 5.44E−134, 7.07E−39
V	50	13.50, 1368, 7.7	2.29E−10, 2.25E−188, 9.81E−10
	40	13.40, 1575.30, 7.3	2.45E−10, 5.67E−200, 4.34E−9
	30	2.0, 1293.2, 9.9	0.0981, 8.48E−184, 1.13E−12
	20	5.2, 1198.4, 5.4	4.34E−4, 1.12E−177, 1.85E−6
	10	7.89, 616.14, 2.47	3.74E−6, 1.21E−127, 0.01
IIIa	50	17.9, 1323.9, 3.4	1.33E−13, 1.06E−185, 9.18E−04
	40	20.2, 1022, 1.9	2.86E−15 4.15E−165, 0.064
	30	20.8, 1000.6, 10.4	1.01E−15, 1.82E−163, 1.99E−13
	20	65.43, 475.68, 37.97	2.62E−43, 3.26E−110, 1.14E−45
	10	35.43, 231.92.9, 14.63	1.23E−25, 2.78E−69, 5.40E−19

Table 8
Average of the area under ROC curve for all data sets.

	Classifier	Data set IVa	Data set V	Data set IIIa
50	SNM	**81.1 ± 3.1**	**71.9± 2.8**	**77.0 ± 4.5**
	LD	72.3 ± 3.7*	69.5 ± 7.9*	63.12 ± 6.4*
	FNN1	77.0 ± 5.7*	67.8 ± 5.0*	70.2 ± 5.2*
	FNN2	77.4 ± 4.6*	69.6 ± 3.7*	70.2 ± 7.2*
	RBF2	77.0 ± 4.2*	67.0 ± 2.6*	70.6 ± 5.4*
40	SNM	**80.10 ± 3.5**	**72.1 ± 2.9**	**75.6 ± 5.9**
	LD	70.4 ± 3.8*	68.4 ± 7.9*	60.3 ± 6.4*
	FNN1	76.3 ± 4.8*	69.0 ± 3.3*	69.2 ± 7.1*
	FNN2	76.8 ± 5.0*	68.7 ± 3.0*	70.6 ± 6.2*
	RBF2	76.4 ± 4.0*	66.6 ± 2.4*	68.6 ± 5.2*
30	SNM	**80.11 ± 4.4**	**71.6 ± 3.2**	**74.9 ± 6.5**
	LD	67.1 ± 4.5*	66.9 ± 8.4*	58.1 ± 6.9*
	FNN1	74.3 ± 6.4*	67.6 ± 3.9*	68.4 ± 7.4*
	FNN2	76.1 ± 5.4*	68.1 ± 3.6*	68.3 ± 7.3*
	RBF2	77.6 ± 4.4*	66.4 ± 2.6*	68.2 ± 6.4*
20	SNM	**79.1 ± 4.9**	**71.9 ± 3.9**	**74.0 ± 6.0**
	LD	60.6 ± 4.9*	65.1± 8.1*	49.0 ± 7.1*
	FNN1	73.5 ± 5.5*	66.2 ± 3.9*	65.0 ± 10.4*
	FNN2	74.6 ± 6.2*	64.9 ± 4.7*	67.0 ± 7.6*
	RBF2	76.8 ± 3.6	66.3 ± 2.8*	67.4 ± 6.7*
10	SNM	**76.8 ± 6.4**	**69.5 ± 4.5**	**69.6 ± 9.2**
	LD	49.6 ± 6.9*	59.4± 9.0*	47.9 ± 10.1*
	FNN1	72.8 ± 5.6*	63.3 ± 4.8*	61.4 ± 12.4*
	FNN2	71.2 ± 9.0*	62.1 ± 5.9*	63.2 ± 12.8*
	RBF2	71.0 ± 6.7*	62.8 ± 4.0*	66.0 ± 8.4

To ensure that the differences in the accuracy values described above are statistically significant, we applied an ANOVA test of two ways regarding the different classifiers and the subjects. In Table 7 are shown the *F* and *p* values. As it can be seen, in all cases it is possible to say that there are differences statistically significant in at least one classifier. Therefore, a multiple comparison test was performed to find those classifiers that report these differences. In the last column of Tables 4–6, the maximal average accuracy value for each condition is marked with bold font, and the accuracy values which result statistically significant different are indicated with an asterisk. As it can be seen, most of the cases the SNM achieved the maximum values, and these values present differences statistically significant to the achieved with the others classifiers.

On the other hand, to evaluate the sensitivity of the classifier, the area under ROC curve was computed, see Table 8. In this case, the ANOVA test was also applied to detect if the differences in the average area are statistically different. The results are shown in Table 9. Once more, the maximal value for each condition is remarked in bold font and the values which are statistically different to it are

Table 9
Results of the ANOVA test of the area under ROC curve values.

Data set	Data training [%]	*F* values	*p* values
IVa	50	60.2, 1027.2, 24.5	6.23E−44, 6.50E−297, 1.16E−57
	40	88, 1107.6, 13.7	5.82E−61, 4.67E−307, 1.33E−32
	30	133.53, 770.93, 7.58	1.92E−85, 2.03E−259, 1.08E−16
	20	242.61, 593.11, 8.88	3.76E−132,6.75E−227, 4.12E−20
	10	291.38, 231.19, 22.89	2.9E−149, 7.34E−128, 3.65E−54
V	50	22.9, 1348.7, 8	3.33E−17, 3.29E−187, 4.43E−10
	40	34.3, 1607.6, 11.9	7.26E−25, 1.17E−201, 2.15E−15
	30	27.8,1296.5, 15.2	1.35E−20, 5.26E−184, 9.6E−20
	20	36.53, 915.44, 4.1	2.42E−26, 1.28E−156, 9.89E−5
	10	45.44, 545.67, 5.14	5.81E−32, 2.67E−119, 3.8E−6
IIIa	50	51.2, 1211.8, 3.9	2.46E−35, 1.43E−178, 1.98E−4
	40	56.6, 1186.4, 4.5	1.97E−38,7.14E−177, 2.88E−5
	30	63.72, 716.59, 7.41	2.21E−42,2.17E−138, 2.84E−9
	20	113.92, 457, 22.88	2.25E−66, 1.24E−107, 3.09E−29
	10	36.95, 182.38, 11.87	1.31E−26, 3.00E−58, 2.30E−15

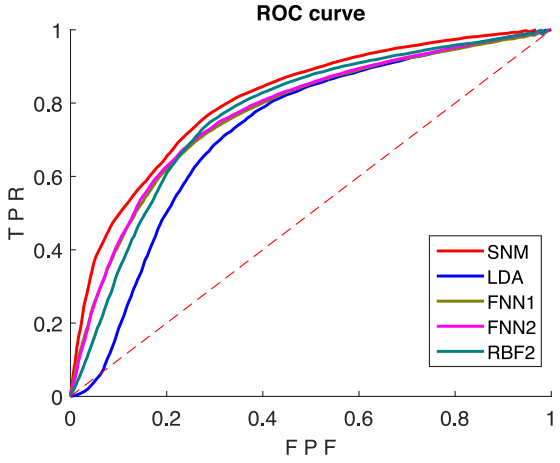


Fig. 3. Average ROC curve obtained with six different classifiers (50% of data for training) for data set IVa.

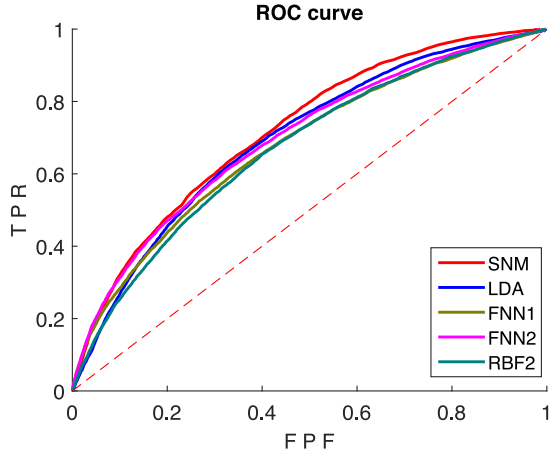


Fig. 4. Average ROC curve obtained with six different classifiers (50% of data for training) for data set V.

marked with an asterisk. As it can be seen, the SNM provided the best area under ROC with the three data sets and all conditions, and these values present differences statistically significant to the achieved with the others classifiers.

Figs. 3–5 show the average ROC curve for all classifiers when 50% of the data are used to training. As it can be seen, the curve obtained by the SNM classifier always describes a better trajectory than those described by the other classifiers. Combining these results with those exposed in Table 9, it is possible to say that the sensibility of the SNM is better than the showed by the other classifiers. Although the artificial neural networks also provided an acceptable ROC curve, it is important to remark that they need several neurons to obtain these results whereas the SNM need only one neuron.

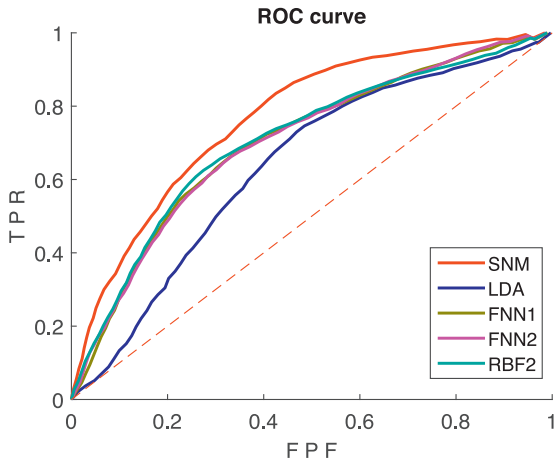


Fig. 5. Average ROC curve obtained with six different classifiers (50% of data for training) for data set IIIa.

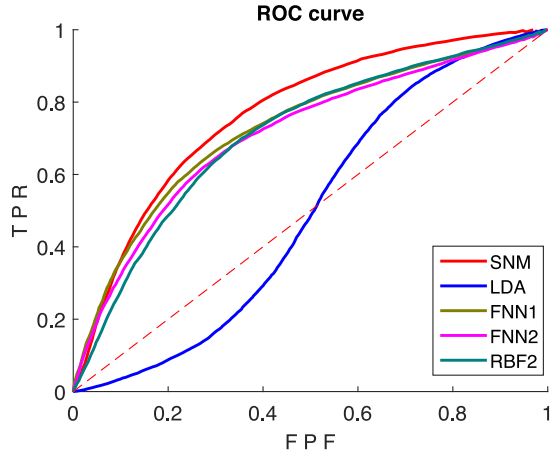


Fig. 6. Average ROC curve obtained with six different classifiers (10% of data for training) for data set IVa.

Moreover, when the data used to training the classifiers is drastically reduced to 10%, the SNM keeps its ability to ranks correctly the feature vectors, this observation is supported by the ROC curves showed in Figs. 6–8.

These results suggest that the SNM can be consider such as an alternative classification tool for the BCI applications in short calibration sessions, even without selecting a specific frequency when the feature is in the frequency domain (in this case the feature used was the coherence). Although in some cases the RBF2 achieved the best accuracy for some subject, in average, we observed that the SNM was more robust when was trained 10% of the total recordings, in contrast to the other tested architectures, where the accuracy diminished to less than the 60%.

In order to compare the performance stability of the SNM against the other classifiers, we computed the confidence intervals (CI) with a significance level of $\alpha = 0.05$. Table 10 shows results obtained with data set IVa. In general, the SNM reached the minimum confidence intervals width which express that the variability in the

Table 10
Average performance, confidence interval's (CI) width and proposed *pci* for all data sets.

Classifier		Data set IVa			Data set V			Data set IIIa		
		Mean	Width of CI	*pci*	Mean	Width of CI	*pci*	Mean	Width of CI	*pci*
SNM	50	73.0	16.2	100	62.9	14.3	79.2	68.1	24.3	100.0
	40	72.1	21.2	87.6	62.8	13.2	93.0	68.0	27.9	91.9
	30	71.5	22.2	84.2	62.8	15.5	74.4	67.3	27.8	90.3
	20	70.4	22.2	81.9	61.4	15.7	64.2	66.2	29.8	83.4
	10	67.7	31.1	57.9	60.3	18.0	48.9	63.6	36.9	61.1
LD	50	68.8	19.0	85.0	64.7	12.4	98.3	61.7	31.4	68.1
	40	67.7	20.1	80.6	63.5	11.9	92.8	61.1	31.7	64.4
	30	65.7	22.3	71.7	63.1	12.8	87.4	58.0	33.8	49.7
	20	62.4	25.8	57.5	61.3	14.3	58.7	50.8	40.9	15.2
	10	50.6	36.5	8.2	57.7	17.3	31.1	50.0	43.8	8.8
FNN1	50	70.5	26.6	73.1	62.4	20.7	53.5	66.0	28.1	86.4
	40	69.8	27.5	69.7	63.6	15.2	80.8	64.8	37.1	64.0
	30	68.0	29.3	62.2	62.2	19.1	57.6	63.6	34.8	65.4
	20	67.8	25.0	70.7	61.1	16.8	58.3	60.5	42.6	40.6
	10	66.4	29.5	58.2	59.1	21.5	28.1	57.8	43.5	31.5
FNN2	50	71.1	24.5	78.6	63.5	17.2	73.8	64.8	33.6	71.3
	40	70.3	27.8	70.2	62.8	15.2	75.4	64.5	29.2	79.9
	30	69.5	29.3	65.5	62.7	16.4	70.9	64.1	38.7	58.7
	20	68.1	30.8	59.4	59.8	20.2	37.8	61.7	39.2	51.1
	10	63.8	40.5	30.2	58.4	26.3	6.6	58.9	48.0	24.9
RBF2	50	70.8	27.1	72.7	62.5	13.5	79.2	66.1	30.5	81.5
	40	71.5	22.6	83.4	61.8	13.3	75.6	61.4	33.6	62.0
	30	72.3	22.4	85.6	62.3	14.4	75.0	61.9	32.7	65.2
	20	70.4	26.8	72.5	61.8	15.8	66.4	62.9	30.9	71.8
	10	65.9	36.6	42.7	58.8	19.8	32.1	59.8	45.5	32.6

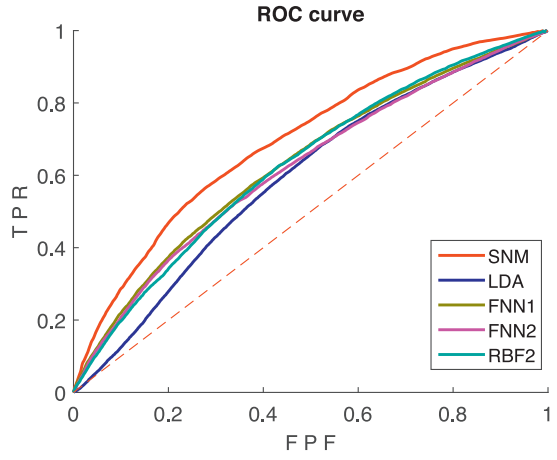


Fig. 7. Average ROC curve obtained with six different classifiers (10% of data for training) for data set V.

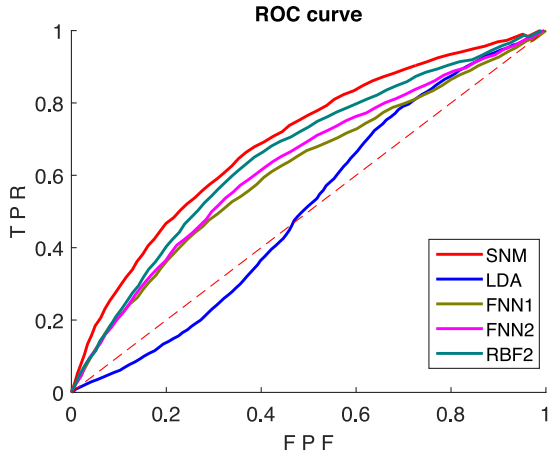


Fig. 8. Average ROC curve obtained with six different classifiers (10% of data for training) for data set IIIa.

results obtained with this classifier was lower than with LDA, RBF2, FNN1 and FNN2.

Table 10 also shows the results obtained data set V. We observed that for all classifiers the confidence interval's width was low since the data set contains several recordings which is an advantage for the stability of statistics. As we expected, the LDA obtained the minimum confidence interval's width followed by the SNM.

For the results obtained with the data set IIIa, Table 10 shows that the SNM provided the best results achieving the minimum confidence interval's width even when the number of recordings was reduced. This results also confirm that SNM is stable through all experiment conditions (diminish the data training from 50% to 10%).

In addition, to compare the behavior of the SNM against the other classifiers in terms of the performance and the confidence interval's width, we proposed a performance-confidence index (*pci*) that combines both measures and express an unique value in the interval [0 100], where 100 corresponds to the best classifier. This index is defined as

$$pci = w_p \left(\frac{p - p_m}{p_M - p_m} \right) + (1 - w_p) \left(1 - \frac{c - c_m}{c_M - c_m} \right) \quad (15)$$

where p is the performance of the classifier, p_m and p_M is the minimum and maximum performance obtained from the set of the experimental results respectively, c is the width of the confidence interval, c_m and c_M is the minimum and maximum confidence interval's width computed from the experimental results, and w_p is a parameter that weight the performance and the confidence interval's width. It is important to mention that if it is necessary to compare the behavior of the classifiers against the ideal classifier the next parameters must be used: $p_M = 100$, $p_m = 100/I$ where I is the number of classes, $c_M = 2 \cdot z_{\alpha/2} \cdot std$ where $z_{\alpha/2}$ is the statistics value according of the confidence level α, std is the expected value of the standard deviation, and $c_m = 0$.

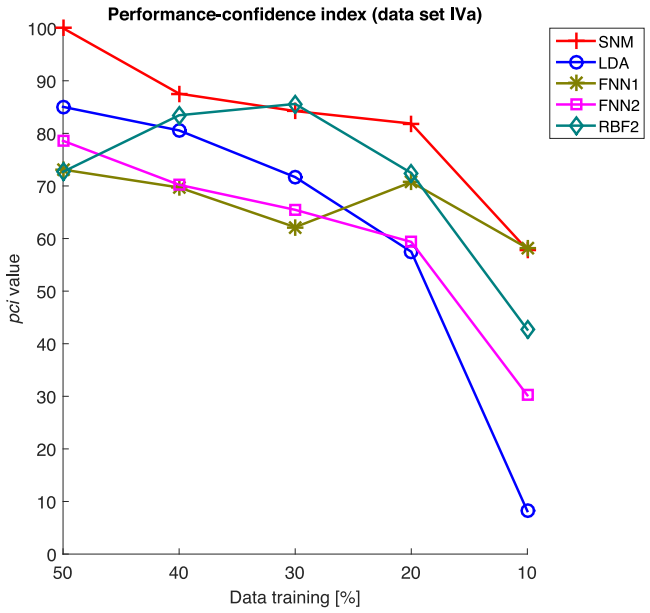


Fig. 9. Behavior of the proposed *pci* obtained with six different classifiers for data set IVa.

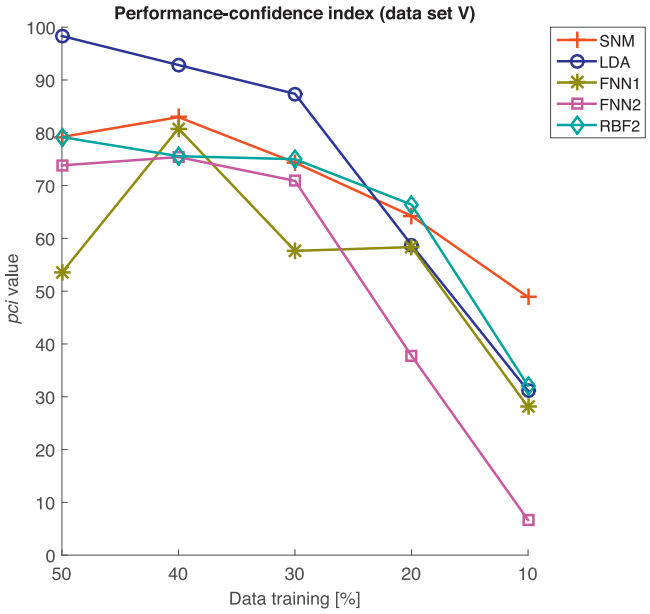


Fig. 10. Behavior of the proposed *pci* obtained with six different classifiers for data set V.

Table 10 shows the results obtained with the *pci* index. As it can be seen, for the data set V (which contains several recordings), the highest *pci* is obtained with LDA when 50–30% of the data samples were used during training stage. However, for 20% and 10% the SNM achieved the highest *pci*. Moreover, for the data sets IVa and IIIa, the SNM provided the best *pci* values for all training conditions.

This result suggests that the SNM is the best option when a reduced number of recordings are collected for each subject during calibration sessions. The behavior of the *pci* when the number of samples in data training decreases can be seen in Figs. 9–11.

Finally, based on the *pci* index, we developed a stability index (*si*) to asses the effect of reducing the size of the data train in the

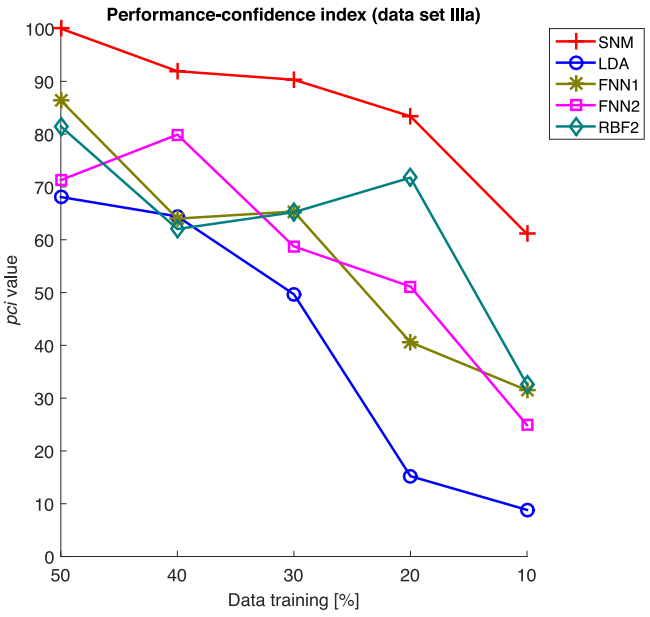


Fig. 11. Behavior of the proposed *pci* obtained with six different classifiers for data set IIIa.

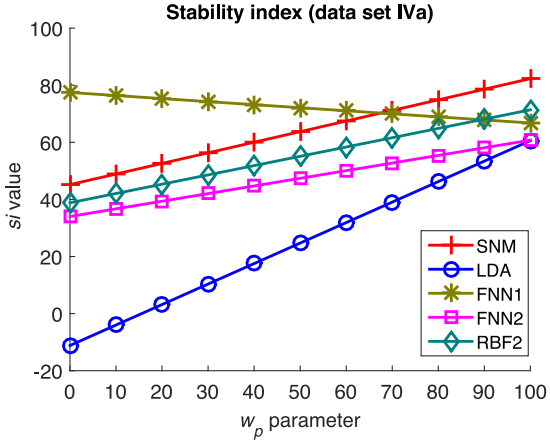


Fig. 12. Behavior of *si*-index for different values of weight factor *w* with six different classifiers. Data set IVa.

SNM and determine if the classifier could be useful to avoid long calibration session with subjects. This index is defined as:

$$si = \begin{cases} w \cdot \left(\frac{pci_a - pci_m}{pci_M - pci_m}\right) + (1-w)\left(1 - \frac{sd - sd_m}{sd_M - sd_m}\right), & pci_a \geq 50 \\ \left[w \cdot \left(\frac{pci_a - pci_m}{pci_M - pci_m}\right) + (1-w)\left(1 - \frac{sd - sd_m}{sd_M - sd_m}\right)\right] \cdot 0.5, & pci_a \leq 50 \end{cases} \quad (16)$$

where pci_a is the average *pci* index obtained with the classifier using different sizes of the training set, $pci_m = 0$ and $pci_M = 100$ is the minimum and the maximum *pci* index value, *sd* is the standard deviation of the *pci* index obtained with the classifier using different sizes of the training set, $sd_m = 0$ and $sd_M = 28$ is the minimum and the maximum standard deviation of the *pci* index value, and *w* is a weight factor for the *pci* index and the standard deviation. In other words, for the case when $w > 0.5$ the performance is more important than stability and when $w < 0.5$ the stability is more important than the performance. Therefore, the *si* index is useful to rank the classifiers according to its stability and performance through different training conditions.

Fig. 12 shows the results obtained from data set IVa in terms of *si* index. As the reader can observe, the best *si* index was obtained

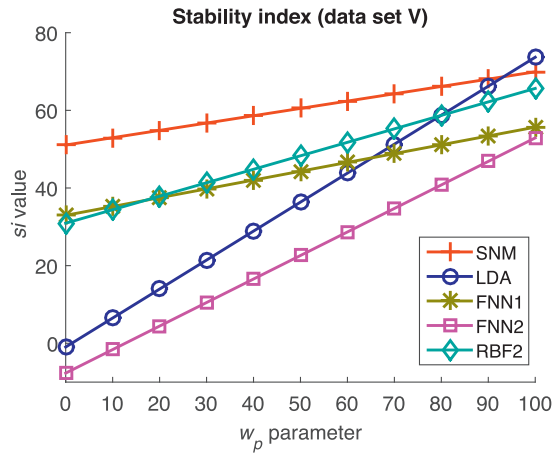


Fig. 13. Behavior of *si*-index for different values of weight factor *w* with six different classifiers. Data set V.

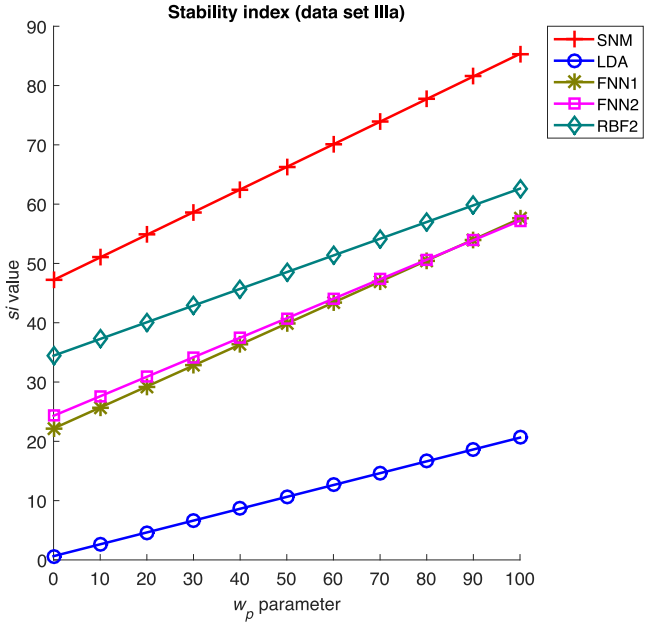


Fig. 14. Behavior of *si*-index for different values of weight factor *w* with six different classifiers. Data set IIIa.

with the SNM, even if we want to weight the stability or the performance. Although FNN1 and FNN2 did not provide acceptable results in terms of the performance, the *si* suggest that this type of neural network provided better stability than RBF2 and LDA. However, when we weight the performance, the *si* suggest that RBF2 was better than FNN1, FNN2 and LDA.

The results obtained from data set V are shown in Fig. 13. As it can be seen, the SNM achieved the best *si* index when $w < 90$. Although the previous analysis based on performance and *pci* suggest that LDA is the best option for this data set, the *si* index shows that the SNM presents more stability in both cases, when stability or performance is weight. Only when $w > 90$ i.e. the performance is the most important, the LDA was the best classifier.

Finally, Fig. 14 shows the behavior of *si* for data set IIIa. As it can be seen, the best *si* index was obtained with the SNM, even if we want to weight the stability or the performance.

Based on the exposed results, we analyzed the behavior obtained with the SNM and showed its robustness when the number of recording is reduced. Although the performance of the SNM was not the best for some subjects and training configurations, in general the values obtained for the ROC area, *pci* and *si* index corroborates the stability of the model, showing that the SNM is the best option to avoid long calibration session with subjects with an acceptable accuracy.

Table 11
Average number neurons in the hidden layer that composed the FNN and RBF for each data set.

Data	Data set IVa		Data set V		Data set IIIa	
Training %	FNN1	RBF2	FNN1	RBF2	FNN1	RBF2
50	35	98	35	139	35	21
40	35	85	35	82	35	10
30	35	43	35	28	35	12
20	35	25	35	21	35	18
10	35	14	35	6	35	6

It is also important to remark that the main advantage of the SNM compared with FNN and RBF is focused on the number of neurons, whereas FNN and RBF that provides the best accuracy require in average more than 38 neurons to obtain acceptable results, the SNN use only one neuron, see Table 11. This advantage could be also important for real time BCI applications.

Another advantage showed in this paper is that it is not necessary to select a specific frequency to achieve acceptable results. This is a relevant fact due to the coherence is highly sensitive to the frequency in which is computed, and restrict the coherence value to specific frequency could provoke to lose the accuracy in the classifications since the highly variability in the brain activity.

4. Conclusions

In this paper, we described the advantage of using a spiking neural model (SNM) to discriminate electroencephalography signals from motor imagery movements when it is necessary to avoid long calibration session. For this purpose, we combined a method for computing a feature vector, from a EEG recording in terms of the coherence, with a method for training a SNM by means of the particle swarm optimization algorithms.

The proposed methodology was tested with three data sets from the BCI International Competition III and the results were compared against LDA classifiers and two different artificial neural networks: Feed-forward neural networks (FNN) and radial basic function neural network (RBF).

The fact that we used data set with different number of samples allow us to explore the behavior of the classifiers in different conditions. For the data sets or conditions where the classifier was trained with several samples, the classical LDA and SNM provided an acceptable performance. However, when the classifiers were trained with data sets composed of a reduce number of samples, the SNM, RBF and FNN provided better results.

In few cases, the RBF and the FNN achieved a better accuracy than the SNM, however, those models in average require more than 30 neurons to obtain that results, whereas the SNM only one neuron. The SNM achieved the best results for most of the subjects. To ensure that the differences in the accuracy values were statistically significant, we applied an ANOVA test of two ways regarding the different classifiers and the subjects, where in most of the cases the SNM achieved the maximum values, and these values presented differences statistically significant to the achieved with the others classifiers. Furthermore, to evaluate the sensitivity of the classifier, the area under ROC curve was computed, where the SNM provided the best area under ROC with the three data sets and all conditions and kept its ability to ranks correctly the feature vectors (this observation is supported by the ROC curves), and these values presented differences statistically significant to the achieved with the others classifiers.

These results suggest that the SNM could be used as a generic classifier and also an interesting option for real-time BCI applications. However, if we want to guarantee the best accuracy for all subjects, we have to tune a specific classifier for each subject.

On the other hand, it was also important to evaluate the robustness of the classifiers in terms of their stability through all configurations and when the number of samples during training is reduced. In that sense, the performance and the confidences intervals were used to propose two different indexes for comparing the performance and stability of the classifiers: performance-confidence index (*pci*) and stability index (*si*).

For the case of the *pci* index, in general the SNM obtained the best results, but for some subjects and training conditions, the RBF and LDA provided better results. These results indicate that for some subjects, the SNM is more stable than the other classifiers.

From the experiments, we also observed that the SNM provided the best *si* index (when the performance is the most important, when the stability is the most important or when an equilibrium between performance and stability is the most important). Furthermore, this index is useful to discover the advantage of some classifiers, not only in terms of the performance but in terms of the robustness with a reduced number of training samples.

Therefore, based in these indexes and the ROC area, we conclude that the results obtained with the proposed methodology are very promising and suggest that the SNM is the best option to avoid long calibration session with subjects with an acceptable accuracy.

Furthermore, this methodology does not require to select a specific frequency to achieve acceptable results, which is very important because some feature extraction methods are highly sensitive to the frequency, this is the case of the coherence. Due to the variability of the brain signals is more convenient to use a frequency range than a specific frequency.

Nowadays, to improve these results, we are exploring new theories to compute the feature vector based on the fractal theory as well as new technique to design spiking neural networks.

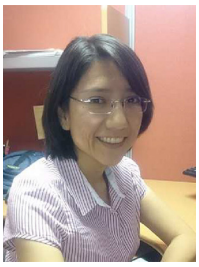
Acknowledgements

The authors would like to thank Universidad La Salle México for the economic support under grant number NEC-03/15 and IMC-08/16.

References

[1] J.R. Wolpaw, N. Birbaumer, D.J. McFarland, G. Pfurtscheller, T.M. Vaughan, Brain–computer interfaces for communication and control, Clin. Neurophysiol. 113 (2002) 767–791.
[2] N.M. Norani, W. Mansor, L. Khuan, A review of signal processing in brain computer interface system, in: IEEE EMBS Conference on Biomedical Engineering and Sciences (IECBES), IEEE, 2010, pp. 443–449.
[3] A. Bashashati, M. Fatourechi, R.K. Ward, G.E. Birch, A survey of signal processing algorithms in brain–computer interfaces based on electrical brain signals, J. Neural Eng. 4 (2007) R32.
[4] F. Lotte, M. Congedo, A. Lécuyer, F. Lamarche, B. Arnaldi, A review of classification algorithms for EEG-based brain-computer interfaces, J. Neural Eng. 4 (2007) R1.
[5] S.N. Abdulkader, A. Atia, M.-S.M. Mostafa, Brain computer interfacing: applications and challenges, Egypt. Inform. J. 16 (2015) 213–230.
[6] F. Popescu, B. Blankertz, K. Muller, Computational challenges for noninvasive brain computer interfaces, IEEE Intell. Syst. 23 (2008) 78–79.
[7] M. Arvaneh, C. Guan, K.K. Ang, C. Quek, Optimizing the channel selection and classification accuracy in EEG-based BCI, IEEE Trans. Biomed. Eng. 58 (2011) 1865–1873.
[8] W. Ting, Y. Guo-zheng, Y. Bang-hua, S. Hong, EEG feature extraction based on wavelet packet decomposition for brain computer interface, Measurement 41 (2008) 618–625.
[9] Q. Wei, Y. Wang, X. Gao, S. Gao, Amplitude and phase coupling measures for feature extraction in an EEG-based brain–computer interface, J. Neural Eng. 4 (2007) 120.
[10] S. Patidar, R.B. Pachori, A. Upadhyay, U.R. Acharya, An integrated alcoholic index using tunable-q wavelet transform based features extracted from {EEG} signals for diagnosis of alcoholism, Appl. Soft Comput. 50 (2017) 71–78.
[11] R. Dhiman, J. Saini, Priyanka, Biogeography based hybrid scheme for automatic detection of epileptic seizures from {EEG} signatures, Appl. Soft Comput. 51 (2017) 116–129.
[12] K. Majumdar, Human scalp {EEG} processing: Various soft computing approaches, Appl. Soft Comput. 11 (2011) 4433–4447.
[13] J. Yang, H. Singh, E.L. Hines, F. Schlaghecken, D.D. Iliescu, M.S. Leeson, N.G. Stocks, Channel selection and classification of electroencephalogram signals: an artificial neural network and genetic algorithm-based approach, Artif. Intell. Med. 55 (2012) 117–126.
[14] K. Aslan, H. Bozdemir, C. Şahin, S.N. Oğulata, R. Erol, A radial basis function neural network model for classification of epilepsy using EEG signals, J. Med. Syst. 32 (2008) 403–408.
[15] W. Maass, T.U. Graz, Networks of spiking neurons: the third generation of neural network models, Neural Netw. 10 (1997) 1659–1671.
[16] M.E. Hasselmo, C. Bodelón, B.P. Wyble, A proposed function for hippocampal theta rhythm: separate phases of encoding and retrieval enhance reversal of prior learning, Neural Comput. 14 (2002) 793–817.
[17] S.J. Thorpe, R. Guyonneau, N. Guilbaud, J.-M. Allegraud, R. VanRullen, Spikenet: real-time visual processing with one spike per neuron, Neurocomputing 58–60 (2004) 857–864, Computational Neuroscience: Trends in Research 2004.
[18] R. Vazquez, B. Girau, J. Quinton, Visual attention using spiking neural maps, in: The 2011 International Joint Conference on Neural Networks (IJCNN), pp. 2164–2171.
[19] R.A. Vazquez, B.A. Garro, Training spiking neural models using artificial bee colony, Comput. Intell. Neurosci. 2015 (2015), Article ID 947098.
[20] A. Cachón, R.A. Vazquez, Tuning the parameters of an integrate and fire neuron via a genetic algorithm for solving pattern recognition problems, Neurocomputing 148 (2015) 187–197.
[21] R. Vazquez, Izhikevich neuron model and its application in pattern recognition, Aust. J. Intell. Inf. Process. Syst. 11 (2010) 35–40.
[22] A. Belatreche, L. P. Maguire, T. M. McGinnity, Pattern recognition with spiking neural networks and dynamic synapse, in: International FLINS Conference on Applied computational intelligence, pp. 205–210.
[23] J. Wade, L. Mcdaid, J. Santos, H. Sayers, Swat: a spiking neural network training algorithm for classification problems, IEEE Trans. Neural Networks 21 (2010) 1817–1830.
[24] N. Kasabov, Evolving spiking neural networks for spatio-and spectro-temporal pattern recognition, in: 2012 6th IEEE International Conference Intelligent Systems (IS), pp. 27–32.
[25] R.A. Vázquez, Pattern recognition using spiking neurons and firing rates, in: Á.F.K. Morales, G.R. Simari (Eds.), Advances in Artificial Intelligence – IBERAMIA 2010, 12th Ibero-American Conference on AI, Bahía Blanca, Argentina, November 1–5, 2010, Proceedings, volume 6433 of Lecture Notes in Computer Science, Springer, 2010, pp. 423–432.
[26] S. Ghosh-Dastidar, H. Adeli, Improved spiking neural networks for EEG classification and epilepsy and seizure detection, Integr. Comput.-Aided Eng. 14 (2007) 187–212.
[27] E. Capecci, N. Kasabov, G.Y. Wang, Analysis of connectivity in neucube spiking neural network models trained on {EEG} data for the understanding of functional changes in the brain: a case study on opiate dependence treatment, Neural Netw. 68 (2015) 62–77.
[28] M.G. Doborjeh, G.Y. Wang, N.K. Kasabov, R. Kydd, B. Russell, A spiking neural network methodology and system for learning and comparative analysis of EEG data from healthy versus addiction treated versus addiction not treated subjects, IEEE Trans. Biomed. Eng. 63 (2016) 1830–1841.
[29] P. Goel, H. Liu, D. Brown, A. Datta, On the use of spiking neural network for EEG classification, Int. J. Knowl.-Based Intell. Eng. Syst. 12 (2008) 295–304.
[30] R. Salazar-Varas, D. Gutiérrez, An optimized feature selection and classification method for using electroencephalographic coherence in brain-computer interfaces, Biomed. Signal Process. Control 18 (2015) 11–18.
[31] S. Sanei, J.A. Chambers, EEG Signal Processing, Wiley, 2008.
[32] L. Faes, G.D. Pinna, A. Porta, R. Maestri, G. Nollo, Surrogate data analysis for assessing the significance of the coherence function, IEEE Trans. Biomed. Eng. 51 (2004) 1156–1166.
[33] W. Gerstner, W.M. Kistler, Spiking Neuron Models: Single Neurons, Populations, Plasticity, Cambridge University Press, 2002.
[34] A.L. Hodgkin, The local electric changes associated with repetitive action in a non-medullated axon, J. Physiol. 107 (1948) 165–181.
[35] L. Abbott, Lapicque's introduction of the integrate-and-fire model neuron (1907), Brain Res. Bull. 50 (1999) 303–304.
[36] R. FitzHugh, Impulses and physiological states in theoretical models of nerve membrane, Biophys. J. 1 (1961) 445–466.
[37] R.M. Rose, J.L. Hindmarsh, The assembly of ionic currents in a thalamic neuron I. The three-dimensional model, Proc. R. Soc. Lond. B: Biol. Sci. 237 (1989) 267–288.
[38] E.M. Izhikevich, Dynamical Systems in Neuroscience: The Geometry of Excitability and Bursting, Computational Neuroscience, MIT Press, 2007.
[39] R.A. Vázquez, B.A. Garro, Training spiking neurons by means of particle swarm optimization, in: Proceedings of the Second International Conference on Advances in Swarm Intelligence – Volume Part I, ICSI'11, Springer-Verlag, Berlin, Heidelberg, 2011, pp. 242–249.
[40] R. Vazquez, Training spiking neural models using cuckoo search algorithm, in: 2011 IEEE Congress on Evolutionary Computation (CEC), pp. 679–686.
[41] R.A. Vazquez, G. Sandoval, J. Ambrosio, How to generate the input current for exciting a spiking neural model using the cuckoo search algorithm, in: X.-S.

Yang (Ed.), Cuckoo Search and Firefly Algorithm: Theory and Applications, Springer International Publishing, Cham, 2014, pp. 155–178.
[42] B.A. Garro, H. Sossa, R.A. Vázquez, Design of artificial neural networks using differential evolution algorithm, in: K.W. Wong, B.S.U. Mendis, A. Bouzerdoum (Eds.), Neural Information Processing. Models and Applications – 17th International Conference, ICONIP 2010, Sydney, Australia, November 22–25, 2010, Proceedings, Part II, volume 6444 of Lecture Notes in Computer Science, Springer, 2010, pp. 201–208.
[43] B.A. Garro, H. Sossa, R.A. Vázquez, Artificial neural network synthesis by means of artificial bee colony (ABC) algorithm, in: Proceedings of the IEEE Congress on Evolutionary Computation, CEC 2011, New Orleans, LA, USA, 5–8 June, 2011, IEEE, 2011, pp. 331–338.
[44] B.A. Garro, R.A. Vazquez, Designing artificial neural networks using particle swarm optimization algorithms, Comput. Intell. Neurosci. 2015 (2015), Article ID 369298.
[45] B.A. Garro, H. Sossa, R.A. Vázquez, Evolving neural networks: A comparison between differential evolution and particle swarm optimization, in: Y. Tan, Y. Shi, Y. Chai, G. Wang (Eds.), Advances in Swarm Intelligence - Second International Conference, ICSI 2011, Chongqing, China, June 12–15, 2011, Proceedings, Part I, volume 6728 of Lecture Notes in Computer Science, Springer, 2011, pp. 447–454.
[46] B.A. Garro, H. Sossa, R.A. Vázquez, Back-propagation vs. particle swarm optimization algorithm: which algorithm is better to adjust the synaptic weights of a feed-forward ANN? Int. J. Artif. Intell. 7 (2011) 208–218.
[47] R.C. Eberhart, Y. Shi, J. Kennedy, Swarm Intelligence (The Morgan Kaufmann Series in Evolutionary Computation), 1st ed., Morgan Kaufmann, 2001.
[48] G. Dornhege, B. Blankertz, G. Curio, K.-R. Muller, Boosting bit rates in noninvasive EEG single-trial classifications by feature combination and multiclass paradigms, IEEE Trans. Biomed. Eng. 51 (2004) 993–1002.
[49] J.d.R. Millan, On the need for on-line learning in brain–computer interfaces, in: Proceedings. 2004 IEEE International Joint Conference on Neural Networks, vol. 4, IEEE, 2004, pp. 2877–2882.
[50] W. Maass, On the computational power of winner-take-all, Neural Comput. 12 (2000) 2519–2535.



Rocio Salazar is a scientific researcher member of Digital Signal Processing group in La Salle University, Mexico. She received her Ph.D. degree in Biomedical Engineering and Physics from Center for Research and Advanced Studies (Cinvestav), Nuevo Leon, Mexico in 2015. During her Ph.D. she was accepted as a visiting student in the Brain–Machine Interface Systems Lab at Miguel Hernndez University, Elche, Spain. Her current research interests are focused on biomedical signal processing as well as feature extraction and classification of EEG signals oriented to Brain-Computer Interface applications.

Roberto A. Vazquez received his B.Sc. degree from the School of Computer Sciences, National Polytechnic Institute (ESCOM-IPN), Mexico City, Mexico, 2003. He received his M.Sc. degree from the Center for Computing Research, National Polytechnic Institute (CIC-IPN), Mexico City, Mexico, 2005 and his Ph.D. degree at the Center for Computing Research, National Polytechnic Institute (CIC-IPN), Mexico City, Mexico, 2009. He is a full time professor at Faculty of Engineering, La Salle University, Mexico City, Mexico. His main research interests are Artificial Intelligence, Neurocomputing, Computational Neuroscience, Associative Memories, Pattern Recognition and Image Analysis.