University of Tartu
Faculty of Mathematics and Computer Science
Institute of Computer Science

Ilya Kuzovkin

Adaptive Interactive Learning: a Novel Approach to Training Brain-Computer Interface Systems

Master's thesis

Supervisor: Konstantin Tretyakov, M.Sc

Author: .... ".......” May 2013
Supervisor: .... ".......” May 2013

Approved for defense
Professor: .... ".......” May 2013

Tartu 2013

Contents

“Rabbit’s clever,” said Pooh thoughtfully.
”Yes,” said Piglet, ”Rabbit’s clever.”
”And he has Brain.”
”Yes,” said Piglet, ”Rabbit has Brain.”
There was a long silence.
”I suppose,” said Pooh, ”that that’s why he never understands anything.”

— A. A. Milne, Winnie-the-Pooh

Introduction

Brain-Computer Interface (BCI) systems allow interaction between a person's brain and a computer. Among various applications of BCI the most intriguing one is the ability to control external devices with the power of thought. A common way to achieve this lies in following consecutive steps:

Reading the signal. This can be done via various neuroimaging techniques such as electroencephalography (EEG), electrocorticography (ECoG), magnetoencephalography (MEG), functional near-infrared spectroscopy (fNIRS), functional magnetic resonance imaging (fMRI), etc.

Signal processing allows us to eliminate artifacts and noise from the raw signal and transform it into an appropriate representation.

Labeling the signal instances. We associate each piece of a signal with some action (stimulus), that the corresponding brain activity should invoke.

Training a classifier. Sophisticated machine learning algorithms allow us to extract signal-to-action mapping from the acquired data and represent it as a model.

Applying the classifier. By applying the model to the new instances of the signal we can estimate which action is associated with the current mental state. Thus we can use mental states to trigger desired behavior of the machine.

However, the result of such a procedure is often much worse than one might hope for. The reason lies neither in inadequate learning algorithms nor in the lack of data. We believe that the main reason is the inconsistency of the incoming signal. The test subject can not control which neurons in his brain will fire in respond to certain mental actions, therefore the same thought or mental intention often results in considerably different neuron activity and, therefore, in different outgoing signals. One way to partially overcome this issue is to expose the test subject to prolonged training, during which he will learn how to *think in the right way* in order to trigger the action.

In this thesis we propose a system, which augments the existing approach to the training of BCI systems with an interaction between the test subject and the system. This introduces additional information for the machine to use in its venture of understanding brain signals as well as lets the human adapt his mental processes to be “understandable” by the machine. The proposed system allows to identify mental states which will impact the classifier the most and which are easily distinguishable by the machine. Once a set of such states is established, the test subject can use it to control the system.

Chapter 1

Biological and Technological Background

1.1 Electroencephalography

Throughout this thesis we work with data acquired using electroencephalography (EEG). In this section we will see how this technology works and talk about the signal it produces.

1.1.1 Biology of the human brain

There are special cells, called neurons, in our brain. These are electrically excitable cells that process and transmit information via electrical and chemical signaling [ner]. The signals we try to capture are generated by changes in the electrical charge of the membrane of the neuron. Neurons have a *resting potential*, which is the difference in the electrical potential between the interior of a cell and extracellular space. The resting potential fluctuates as a result of the impulses arriving from other neurons through *synapses.* Cell membrane contains *ion channels* where ions of sodium, potassium, chloride and calcium are concentrated during the chemical processes in the cell. Concentration of ions creates cross-membrane voltage differences. Changes in the cross-membrane voltage generate *postsynaptic poten-*

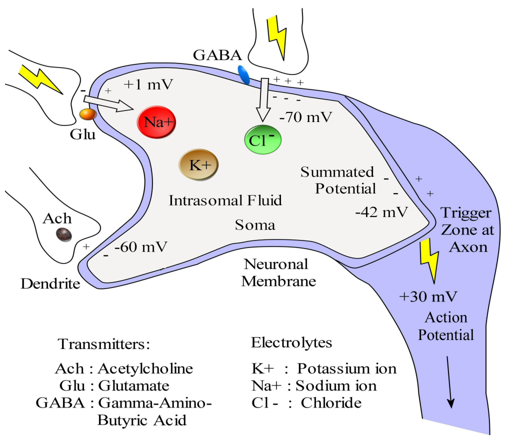


Figure 1.1: Summation of postsynaptic potentials. [Won10]

tials, which cause electrical flow along the membrane and dendrites [Fis91].

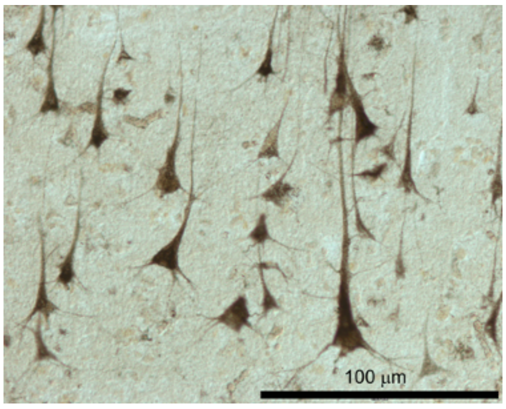


Figure 1.2: Pyramidal cells [aud]

A neurons consists of a cell body, dendrites and an axon. When a summated potential at the trigger zone of the axon reaches the threshold of –43 mV (can vary), it fires the axon by generating an *action potential* of +30 mV that goes along the axon to release the transmitters at the end of it (see Figure 1.1). When a summated potential is below the threshold, the axon rests. This summation can be observed well at the vertically oriented pyramidal cells (Figure 1.2) of the cerebral cortex due to following properties of these cells:

- The dendrites of the pyramidal cells extend through nearly all layers of the cortex, guiding the flow of currents generated by postsynaptic potentials from the deep layers to the more superficial ones.
- These cells are closely packed and oriented parallel to each other, facilitating spatial summation of the currents generated by each neuron.
- Groups of these neurons receive similar input and respond to it with potentials of similar direction and timing.

Despite the fact that most of the currents remain inside the cortex, small fraction penetrates to the scalp, where it causes different parts of the scalp to have different electric potentials. These differences, having amplitudes of usually 10-100 μV are detected by electrodes and constitute the *electroencephalogram* (EEG).

There are different types of neurons in the brain, and about 20 major types of spiking activity [dB05]. Most of those spiking patterns are periodic [Izh03]. This fact makes it reasonable to decompose the raw EEG signal into frequency components as we will show in Section 1.2.

1.1.2 EEG technology

By observing differences between electrical signals coming from the different locations on the scalp, we can monitor brain activity, see which parts of the brain are active during different types of activities, and how high the activity is.

EEG mainly reads *postsynaptic potentials*, which are relatively sustained (a potential persists up to 100 ms). The *action potentials* (actual neuron firings) are very brief (1 ms) and their electrical contribution is small, so we cannot track them on the EEG. If we could be able to see exact moments when a neuron sends the signal, we would obtain much more information about the processes in the brain.

As we have mentioned before, electrodes read potentials, so we can see differences between them. Usually there is one or several *reference electrodes*, which are placed on the most electrically stable areas, such as nose or ears. Other electrodes' potentials are being compared with the reference ones.

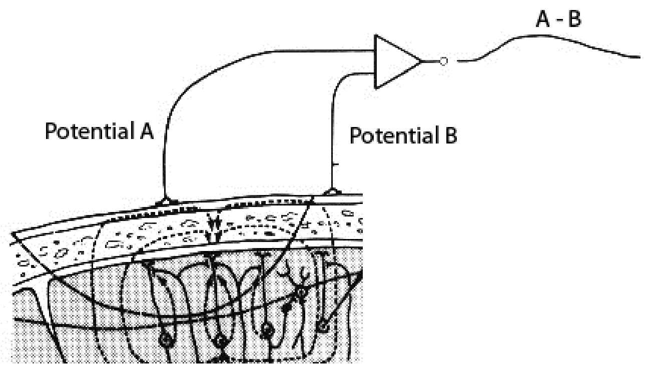


Figure 1.3: Two electrodes on scalp areas with different electric potentials produce the signal. [Fis91]

1.1.3 EEG signal

An EEG signal consists of time series of fluctuating electrical potential (μV) on several channels. Each channel represents an electrode placed on the head. Each EEG machine has a certain *sampling rate*, which indicates how many changes per second the outgoing signal can carry. We work with the Emotiv EPOC [Emo12]. It has a sampling rate of 128 Hz, 14 usual and 2 reference electrodes. Figure 1.4 demonstrates 1000 samples (7.8 seconds) of the raw signal on 14 channels.

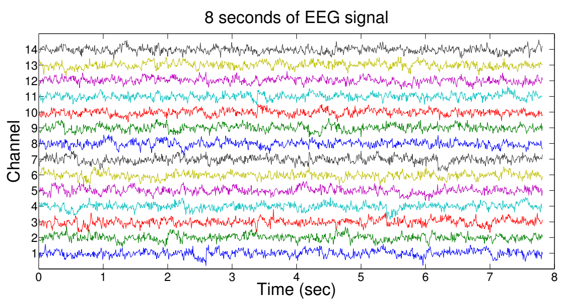


Figure 1.4: EEG data: 14 channels, 8 seconds, 128 samples per second.

The mental state of the test subject will vary over time, therefore we need to represent the signal in such way that different moments (or periods) of time will constitute separate *instances* in the resulting dataset. The naïve representation would be to take each time point of the data sample and compose the instance as voltage reading on each channel. One such instance would be represented with a 14-element vector. However, such representation will loose valuable periodic information, which, as we will see, characterizes brain signals.

As we have seen in Section 1.1.1, neuron fires periodically and, depending on the type of activity, the firing periodicity changes. When the brain region is involved in a certain kind of activity, then all the neurons in this region fire with the same periodicity [GPD85] and those joint fluctuations are reflected in the signal. According to frequency, brain waves are classified into several groups, which are believed to correspond to specific types of activity each [LdS91]. Table 1.1 briefly lists basic types of brain waves and examples of respective activities.

Name	**Type of activity**
Delta up to 4 Hz	• Adult's slow wave sleep • Babies' brain activity • Continuous attention tasks [KABG$^+$06]
Theta 4–8 Hz	• Brain activity of young children • Drowsiness or arousal • Idling • Person is actively trying to repress a response or action [KABG$^+$06]
Alpha 8–13 Hz	• Wakeful relaxation • Eyes are closed • Inhibition of areas of the cortex not in use, plays an active role in network coordination and communication. [PP07]
Mu 8–13 Hz	• Appears in resting neurons responsible for motor activity
Beta 13–30 Hz	• Alert • Active, anxious thinking, concentration • Work
Gamma 30–100+ Hz	• Cross-modal sensory processing (perception that combines two different senses, such as sound and sight) [KC06] [KSO07] [NYC04] • Appears during short term memory matching of recognized objects, sounds, or tactile sensations [HFL10]

Table 1.1: Basic frequency ranges and corresponding activity types.

In this context, the *time–frequency representation* looks like a natural choice, since it allows us to see which frequency components are present in the signal at certain time points. One way to convert a time series to the time-frequency domain is the *windowed Fourier transform*, which decomposes the signal into periodic components.

1.2 Fourier Transform

The frequency representation of a signal x_t where $t = \{0, \ldots, N-1\}$ is computed using the *Fourier transform* as follows:

$$X_k = \sum_{t=0}^{N-1} x_t e^{-i2\pi k \frac{t}{N}} \tag{1.1}$$

where N is the total number of samples, k is the frequency under investigation, and X_k is the energy of frequency k in the signal.

Listing 1.1 provides Matlab code to compute the Fourier transform explicitly on

Listing 1.1: Fourier toy example

```
% Generate signal: 1 second, 128Hz, sinusoid at 20Hz
timepoints = 0:0.0078:0.9922;
signal = sin(2 * pi * 20 * timepoints);

% Prepare vectors
time = 1:length(timepoints);
frac = time / length(timepoints);
freqs = zeros(1,50);

% Perform Fourier transform
for w = 1:50
    component = sum(signal .* (exp(1) .^ (-i * 2 * pi * w * frac)));
    freqs(w) = sqrt(real(component)^2 + imag(component)^2);
end

% Show result
bar(freqs)
```

a toy example. Figure 1.5 shows the initial signal and the frequency components detected in the signal by the transform.

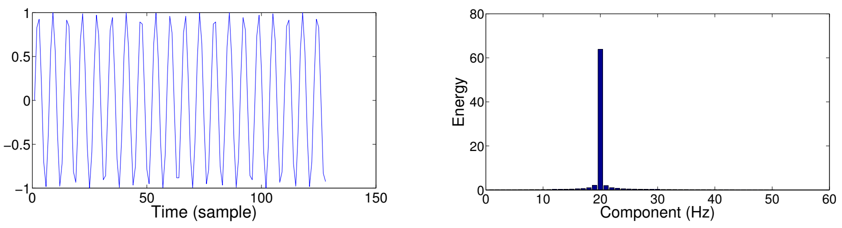


Figure 1.5: Raw signal (on the left) and its decomposition into periodic components using Fourier transform.

To compute the time-frequency representation of a signal we split it into windows and compute the Fourier transform of each window. Figure 1.6 presents the result. This is the representation of a signal we will use throughout this thesis. The rows are frequency components and the columns are points of time where the transform was applied.

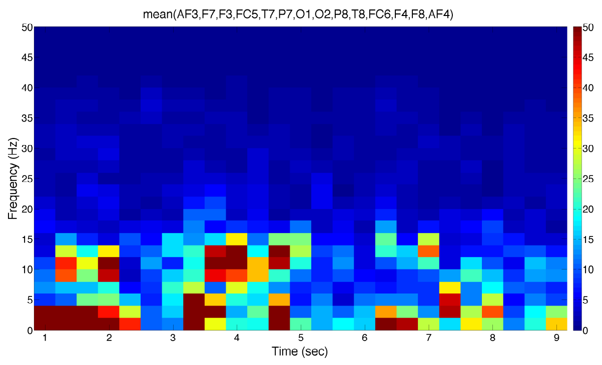


Figure 1.6: Time-frequency domain representation of the EEG signal from Figure 1.4, window length is 0.3 sec.

1.3 Machine Learning

Machine learning is a branch of computer science, that studies algorithms, which are capable of learning from data. The data is represented as a set of *samples*. Each sample is a vector of *features*, which describes the sample's characteristics, and a *label*, which identifies to which *class* the data sample belongs. The ultimate goal is to find a function (also called the *model*), which will be able to map samples from the feature space to the label space. In other words, an algorithm will find dependencies between features and labels, represent those dependencies as a model and using this model the algorithm will be able to correctly classify new data samples, which were not part of the training data.

1.3.1 Supervised and unsupervised learning

There are two major branches of machine learning: supervised and unsupervised.

Supervised learning. Supervised learning relies on labeled data samples. To create a model, a supervised learning algorithm requires some amount of training data, where each sample has the correct label. Formally, given a dataset $D = \{(\mathbf{x_1}, y_1), \ldots, (\mathbf{x_n}, y_n)\}$ of labeled points (where $\mathbf{x_i}$ are the *feature vectors* and y_i are the *labels*), a learning algorithm produces a function

$$f(\mathbf{x_i}) \to y$$

In some cases it is more convenient to use the probabilistic prediction function

$$f(\mathbf{x_i}) \to \mathbf{p}$$

where $\mathbf{p}$ is a vector of probabilities, where for each class c we have the probability that vector $\mathbf{x}$ will be classified as class c.

Unsupervised learning. This family of algorithms uses samples *without* labels as an input. These algorithms analyze the internal structure of the data and based on it assign data samples to different *clusters*. This is achieved by measuring distance between the samples in the feature space using various kinds of similarity measures and placing similar samples in the same cluster.

In our case each sample is a vector of length cf, where c is the number of channels in the raw signal and f is number of frequency components we take from the Fourier transform of each channel. In this thesis we will use $c = 14$ and $f = 25$ (frequencies from 1 to 50 Hz grouped into 2Hz bins), therefore the dimensionality of our samples is 350.

In this section we give a high level description of three supervised learning classification algorithms:

- Support Vector Machines (SVM)
- Naïve Bayes Classifier
- Decision Tree Classifier

and two unsupervised learning algorithms (*clustering algorithms*):

- Self-Organizing Maps (SOM)
- K-means

that we use in our work.

1.3.2 Support vector machines

Support Vector Machine (SVM) [CV95] is a supervised linear classifier learning algorithm. The idea of the algorithm is to find a linear separation boundary between the samples, which correctly separates instances of different classes and has the maximal *margin* – distance from the separating boundary to the closest training sample (see Figure 1.7).

Support vector machines can also work in nonlinear spaces. This is achieved by transforming the instance space using nonlinear mapping. With a nonlinear mapping a straight line in the new space corresponds to a nonlinear boundary in the original space. In such way a linear model constructed in the new space can represent a nonlinear model in the original space [WF05].

For efficient SVM implementation we used a Matlab toolbox based on the `libsvm` library [CL11].

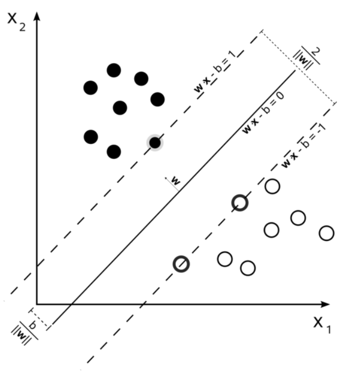


Figure 1.7: Maximal margin plane separating two classes. Samples lying on the boundaries of the separation plane are called support vectors.

1.3.3 Naïve Bayes classifier

The Naïve Bayes classifier algorithm is an example of a statistical modeling technique.

The method [FGG97] is based on the Bayes formula of conditional probability

$$\Pr[A|B] = \frac{\Pr[B|A] \cdot \Pr[A]}{\Pr[B]} \tag{1.2}$$

Let $\mathbf{x}$ be the vector of attributes $(x_1, x_2, \ldots, x_n)$ and y be the class of an instance. Then, by applying the formula (1.2) we express the probability that instance $\mathbf{x}$

belongs to class y as

$$\Pr[y|\mathbf{x}] = \frac{P[\mathbf{x}|y] \cdot \Pr[y]}{\Pr[\mathbf{x}]} \propto \Pr[\mathbf{x}|y] \cdot \Pr[y], \tag{1.3}$$

(we can drop the denominator, because we can normalize the probabilities of all attributes to sum to 1). If we now assume conditional independence of the attributes, then

$$\Pr[\mathbf{x}|y] \cdot \Pr[y] = \Pr[x_1|y] \cdot \Pr[x_2|y] \cdot \ldots \cdot \Pr[x_n|y] \cdot \Pr[y], \tag{1.4}$$

where $\Pr[x_i|y]$ can be estimated from the training data.

The conditional independence assumption is where the term “naïve” comes from. In reality, features usually are correlated with each other. But this assumption gives us a simple model that often performs surprisingly well [WF05].

We used the Matlab built-in implementation of the Naïve Bayes classifier.

1.3.4 Decision tree classifier

A decision tree is a data structure, which consists of *leaves*, which are labelled with a certain class and *decision nodes*, which describe branching conditions based on the features from the feature space. A decision tree learning algorithm constructs the tree from the data. The algorithm uses *entropy* for choosing the branching feature at each of the nodes. The feature, separation along which results in the largest *information gain*, is chosen to be the branching criterion. Information gain and entropies are being calculated based on the training data provided to the algorithm.

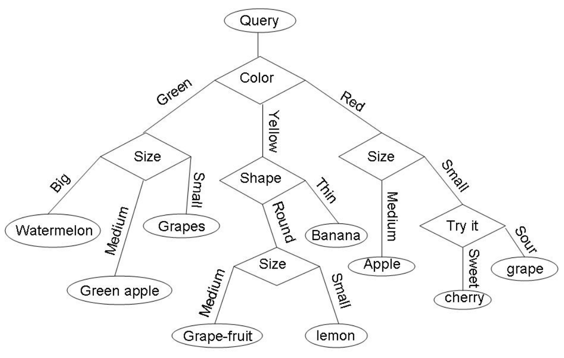


Figure 1.8: Toy example of a decision tree [tre].

After the tree structure is complete, classification of new data samples can be performed. Each sample is propagated along the tree until it reaches any of the leaves. At each node of the tree certain feature of the data sample is tested according to the branching condition of the node. The result of this test determines along which path of the tree the current data sample will move afterwards.

We used the Matlab built-in implementation of the decision tree classifier, which is based on the C4.5 algorithm [Qui93].

1.3.5 Self-organizing maps

A self-organizing map (SOM) [Koh82] is a collection of m units organized into a multidimensional rectangular grid. Most commonly (and also in this work) a two-dimensional grid is used. To each unit is assigned vector $\mathbf{w}(u) \in \mathbb{R}^d$, where d is the dimensionality of the data.

Initializing the map

The first step is to decide on the map size and ratios of the map sides. In the technical report for the SOM Toolbox [VHAP00] authors estimate size of the map using formula $5n^{0.54321}$, where n is the number of samples in the training set. Since there is no explanation about how they came up with this number we assume that it was either picked at random or empirically estimated.

Once the number of units is established we choose the structure of the map. Again, according to [VHAP00] the ratio of the sidelengths of the map should be based on the ratio between two biggest eigenvalues of the covariance matrix of the data. Keeping this property in mind the sidelengths are set so that their product is as close to the estimated number of units as possible. This is essentially the same as taking the two first principal components of the training data. Such choice is motivated by the fact that the two axes of the resulting map will account for as much variance of the data as possible.

After the map is created, each map unit is initialized to the $\mathbf{0}$ vector. Now the main phase of the algorithm starts, during which the map will update weigth vectors according to the incoming data.

Updating weight vectors

Definition 1.1. *The best matching unit* (BMU) for a data sample $\mathbf{x}$ is a unit u, whose weight vector $\mathbf{w}(u)$, which is closest to $\mathbf{x}$:

$$BMU(\mathbf{x}) = \underset{u \in \{1 \ldots m\}}{\operatorname{argmin}} \ \operatorname{distance}(\mathbf{w}(u), \mathbf{x}) \tag{1.5}$$

Usually euclidean distance is used as the distance measure:

$$\operatorname{distance}(\mathbf{w}, \mathbf{x}) = \sum_{i=1}^{|\mathbf{x}|} (\mathbf{x}_i - \mathbf{w}_i)^2 \tag{1.6}$$

After the BMU for the incoming data sample is established we update the map. This procedure is explained by Equation (1.7), where u is a unit of the map, $\mathbf{w}(u)$ is the weight vector, which is assigned to u and which we are going to update with respect to the new data, s is the number of the current iteration and $\mathbf{x_t}$ is the incoming data vector.

$$\mathbf{w}(\mathbf{u})^{s+1} = \mathbf{w}(\mathbf{u})^{s} + \Theta(BMU, u, s)\alpha(s)(\mathbf{x_t} - \mathbf{w}(\mathbf{u})^{s}) \tag{1.7}$$

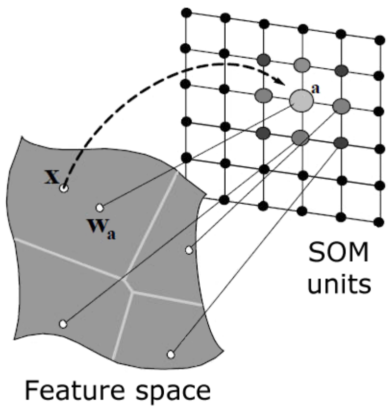


Figure 1.9: Mapping between vectors in the feature space and SOM units. [som]

The update is repeated for each iteration $(1, \ldots, \lambda)$, for each input data vector $(\mathbf{x_1}, \ldots, \mathbf{x_n})$ in the training set and for each unit in the map $(u_1, \ldots, u_m)$. In total this procedure is being repeated up to λnm times, where λ is the iteration limit, n is the number of samples in the training data and m is the size of the map. Not all units are updated with each new input vector, furthermore, not all units among the updated ones are updated equally. There are two functions in Equation (1.7), which are responsible for deciding which units will be updated and how much. $\Theta(b, u, s)$ is called the *neighborhood function*, it determines to what extent unit u is neighbor of b: for b itself $\Theta(b, b, s) = 1$ and for some unit u, which is too far away to be considered to be a neighbor of b $\Theta(b, u, s) = 0$. The parameter s is used to decrease the number of neighbors on later iterations. The function $\alpha(s)$ can be interpreted as *learning rate*. It also takes the current iteration as a parameter and uses that do decrease with time the influence that new samples have on the weight vectors.

At the end of this process units of the resulting map represent centers of the data clusters. Each new data sample is likely to be assigned to the cluster, which is "populated" by similar samples. A second important property of the final map is that some units are closer to each other, and the distance between them (i.e. their weight vectors) indicates the distance between the cluster centers in the feature space.

SOM Toolbox

The self-organizing map implementation is taken from the SOM Toolbox [VHAP00], which is created by a research group at Helsinki University of Technology, the same university where SOM was invented.

1.3.6 K-means

K-means [M+67], is a simple clustering algorithm that can serve as the counterpart for SOM and be our point of reference when analyzing complexity of the EEG signal. Here is a high level description of the steps this algorithm makes in order to separate the data into clusters:

1. Start by selecting k random vectors in the data space, where k is the number of clusters we want to identify. These vectors are called *cluster centers.*
2. Each data sample is assigned to closest cluster center.
3. For each cluster, we update the cluster center to be the average of all the samples assigned to the cluster.
4. Step 2 and 3 are repeated until convergence (i.e. until cluster centers do not change).

1.3.7 Performance evaluation

In the field of machine learning model performance is evaluated on the so-called *test data.* This is a set of data samples, which were collected separately from the training data. Samples from the test set are labelled in the same way as the training data samples are, but we will not give those labels to the algorithm this time. Algorithm should use the model it created from the training data to predict the label for the test set. After that we compare predicted labels with the actual ones and estimate model accuracy.

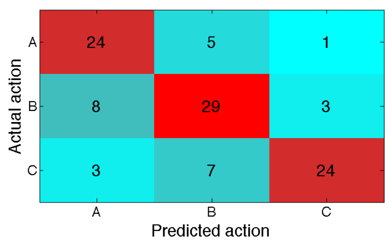


Figure 1.10: Visualization of the example of a confusion matrix.

One way of keeping track of the model's successes and failures is the *confusion matrix.* Data samples, which are classified correctly by the model, contribute to the main diagonal on the confusion matrix. All misclassified samples introduce numbers off the main diagonal.

Definition 1.2. A *confusion matrix* M is an integer matrix of size $a \times a$ where a is the number of classes. Number M_{ij} indicates the number of times a test instance, whose true label is i was classified by the model as belonging to class j.

In our work we will use a modification of the confusion matrix, which we call a *probabilistic confusion matrix.*

Definition 1.3. A *probabilistic confusion matrix* is a matrix $M \in \mathbb{R}^{n \times n}$, where n is the number of actions. For a dataset $D = \{(\mathbf{x_1}, y_1), \dots, (\mathbf{x_n}, y_n)\}$ we define the probabilistic confusion matrix as

$$M_{ij} = \sum_{\mathbf{x} \in D, y=i} f(\mathbf{x})_j$$

where $f(\mathbf{x})$ is the probabilistic prediction function described in Section 1.3.1.

There are three numbers, which can be calculated using the matrix and which we use to characterize model performance:

Precision is a fraction of samples correctly identified as a class c among all samples identified as a class c:

$$\text{Precision}_p = \frac{M_{pp}}{\sum\limits_{a \in classes} M_{ap}} . \tag{1.8}$$

Recall is a fraction of samples of class c correctly identified as such:

$$\text{Recall}_a = \frac{M_{aa}}{\sum\limits_{p \in classes} M_{ap}} . \tag{1.9}$$

F1-score is the harmonic mean of precision and recall. This measure allows us to represent model performance as one number:

$$\text{F-score}_c = 2 \cdot \frac{\text{Precision}_c \cdot \text{Recall}_c}{\text{Precision}_c + \text{Recall}_c} . \tag{1.10}$$

Each measure is computed for each class. Since we can have more than two classes in out data sets, we take a weighted average of F1-scores to describe the whole model's performance. We compute weights of actions in the confusion matrix as shown in Equation (1.11) and calculate dot product of weights and F1-scores (1.12).

$$\text{Weight}_a = \frac{\sum\limits_{p \in classes} M_{ap}}{\sum\limits_{i \in classes} \sum\limits_{j \in classes} M_{ij}} \tag{1.11}$$

$$\text{Weighted F-score} = \sum_{c \in classes} \text{Weight}_c \cdot \text{F-score}_c \tag{1.12}$$

The resulting number reflects how well the model performs.

1.3.8 Principal component analysis

Principal Component Analysis (PCA) is a method dating as far back as 1901 [Pea01]. It allows to transform data points from one feature space to another and gain useful properties along the way. In the original space, features can be linearly correlated to each other. PCA aims at transforming the data so that the new features are linearly uncorrelated. Those features are called *principal components.* The first principal component will account for as much of the variability in the data as possible. The second component has to be orthogonal to the first one and also account for as much variability as possible under the orthogonality constraint. Each following component obeys the same rule.

The result of PCA provides us with following useful information:

- By looking at several first principal components of data samples for particular class we can understand whether the data features are highly correlated or not, which gives an abstract understanding of the complexity of the data.
- We can reduce dimensionality of the feature space by taking only as many features as needed to account for enough variance in the data. In many cases this number can be considerably smaller than the initial number of dimensions in the original feature space.

Chapter 2

Emotiv EPOC and EEG Signal

In this chapter we describe the device we use in our work and study the properties of the signal this device is capable of producing. After that we demonstrate why solving the task of classification of mental states is hard by looking at some real examples of the data.

2.1 Emotiv EPOC

Real data experiments were conducted using the Emotiv EPOC® [Emo12]. The device has 14 electrodes and 2 reference channels. Figure 2.1 shows electrode placement according to the International System of Electrode Placement (*10-20 System*) [TBS+93].

Table 2.1 shows Emotiv EPOC parameters according to the vendor documentation [emo] and compares those parameters with a high-end EEG device BioSemi ActiveTwo® [VRPG90].

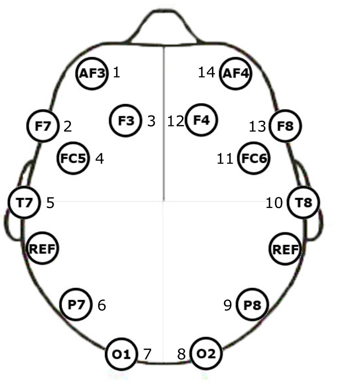


Figure 2.1: Emotiv EPOC electrode placement according to the 10-20 system.

	Emotiv EPOC	**BioSemi ActiveTwo**
Sampling rate	128 Hz	16384 Hz
ADC error margin	0.51μV	31 nV
Frequency bandwidth	0.2 – 45 Hz	up to 3200 Hz

Table 2.1: Comparison of the Emotiv EPOC and Biosemi ActiveTwo EEG devices. ADC stands for analog-to-digital converter.

As we see Emotiv EPOC is a far less sophisticated device and we may assume that the quality of the signal is also quite low. However, there are a few features which make Emotiv EPOC attractive:

- Low cost allows end users to acquire this device for personal use,
- Portability makes it possible to conduct studies outside the laboratory.

If we would be able to achieve reasonable classification accuracy levels with this device, we could extend areas of applications of BCI technology.

2.2 Signal Consistency

As we mentioned in Section 1.3, we represent each data sample by the concatenated vector of frequencies from each channel. We have 14 channels in the data, and we are interested in frequency range from 1Hz to 50Hz, which we group into 2 Hz bins. This gives us 25 features to describe each channel, so our sample belongs to a $14 \times 25 = 350$-dimensional space.

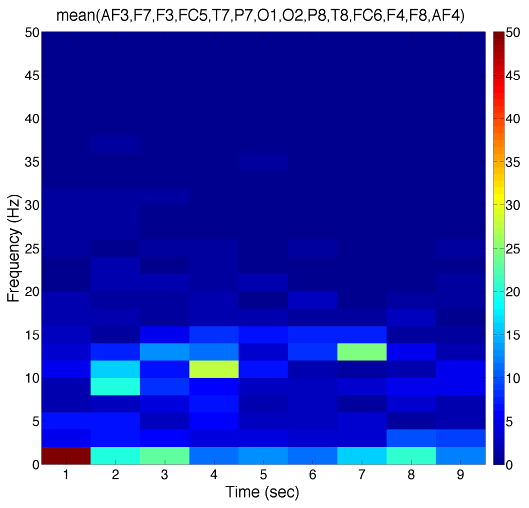


Figure 2.2: 10 seconds of real EEG data during same mental state.

For each data sample our goal is to detect in which one of the fixed number of *mental states* the test subject was when he produced the sample. We must do so solely by analyzing the signal representation (position of the sample in the feature space). The test subject's task is to produce similar brain activity each time he is engaged in a certain mental state. In other words, signal representation of any two time moments when the test subject was thinking "left" must be similar to each other.

Definition 2.1. Intentional attempt to sustain stable brain activity will be referred to as *mental state.*

Figure 2.2 gives an example of real EEG data processed as described in Section 1.2. The picture demonstrates the signal, obtained during 10 seconds when the test subject tried to think "left". We can see from the signal that almost each next sample somewhat differs from previous one. One particular picture can be unconvincing, but according to our studies this is the general case, that signal produced by the test subject in response to the same stimulus can be very different, which makes is hard for an algorithm to identify the stimulus from the signal.

This is our first problem, the *inconsistency of the signal.* The source of this problem is the test subject's inability to produce similar brain activity for similar mental tasks. One can overcome the issue via excessive learning: given long enough time the test subject can learn to think in the *right* way. Efficiency of the learning process then naturally becomes one of the bottlenecks of the system as a whole.

2.3 Signal Distribution Over Different Mental States

Signal samples can be quite different for the same mental state. This naturally leads to an idea that if they are all different, we might be able to map each of them to a certain action. In such case actions would be described by non-intersecting sets of signals and the only thing a learning algorithm would have to do is to divide signals into sets according to actions.

Definition 2.2. An *action* is the desired behavior of a BCI system that the test subject is trying to trigger by sustaining the approproate mental state.

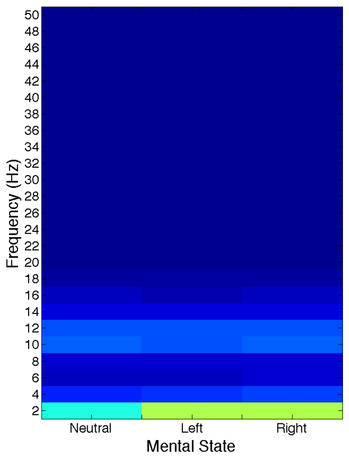


Figure 2.3: Averaged frequency energies for 3 different mental states.

For this idea to work we expect that signals produced for different mental states would end up in different sets. Let us investigate this assumption. Figure 2.3 shows averaged frequency energies over 1000 samples for each of the three mental states. We see that they are not exactly equal, which gives us some hope, but we also see that they are very similar to each other. This suggests that dividing samples into several disjoint sets may be difficult.

We used SOM to separate mental states into different clusters. On the left side of Figure 2.4 is the map built on the artificial data where we introduced three distinguishable signals into randomly generated EEG-like data [YBHC04]. We can see the formation of three distinct clusters in the corners of the map. On the right side of the figure we see the result of applying the exactly same method to real data. What we see is consistent with conclusions we made before – the task of distinguishing mental states is far from trivial. Currently, SOM tends to put all samples from the test subject's brain into the same place, since they are very similar. Our task will be to teach the test subject to produce as different mental states as possible and at the same time adapt the SOM to handle data with high level of internal similarity. This leads us to our second problem: the test subject can not produce *distinguishable mental states.* However, it is known that

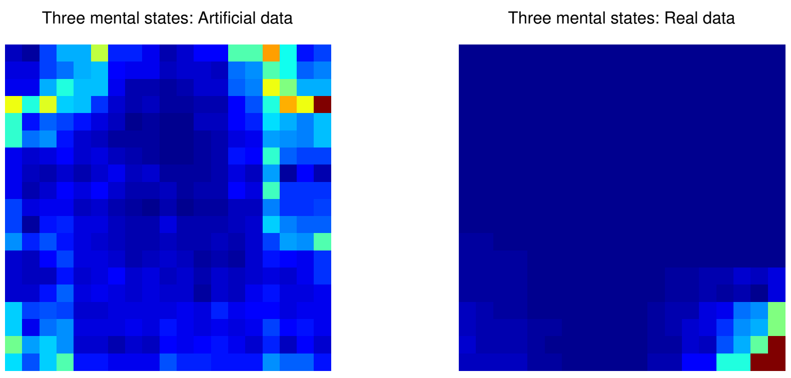


Figure 2.4: Applying self-organizing map on artificial and real data to separate samples from different mental states.

prolonged training makes it possible [SE05]. This again makes learning process the cornerstone of system efficiency.

2.4 Empirical Estimations of Complexity of Real Data

2.4.1 Comparing SOM and K-means on real data

In most cases of unsupervised learning problems, simpler algorithms such as k-means can yield good results. However, EEG data seems to fall into the category of data with a more complex internal structure.

We performed two clusterings on the same dataset (12000 samples) using K-means and SOM. For both methods we computed clustering matrices, where rows are actual clusters given by the labels and columns are the clusters algorithm assigned data samples into.

$$\begin{pmatrix} 3890 & 0 & 0 \\ 0 & 3800 & 0 \\ 0 & 0 & 4796 \end{pmatrix} \qquad \begin{pmatrix} 0 & 3890 & 0 \\ 0 & 3800 & 0 \\ 7 & 4783 & 6 \end{pmatrix} \qquad \begin{pmatrix} 3141 & 511 & 238 \\ 3151 & 443 & 206 \\ 3931 & 586 & 279 \end{pmatrix}$$

This matrix represents actual cluster distribution.

K-means. The algorithm was not able to separate the data: almost all samples are clustered into the second cluster.

SOM. Most of the data samples are clustered into same cluster as well, but other clusters also received small portion of the data samples.

Based on these results we decided to use SOM as primary clustering algorithm in our system.

2.4.2 Analyzing principal components

Additional way to visually access the complexity of the data is provided by Principal Component Analysis (PCA).

On Figure 2.5 we can see the first three principal components of the data. We can see that neither of the components describes any of the three classes (represented by colors).

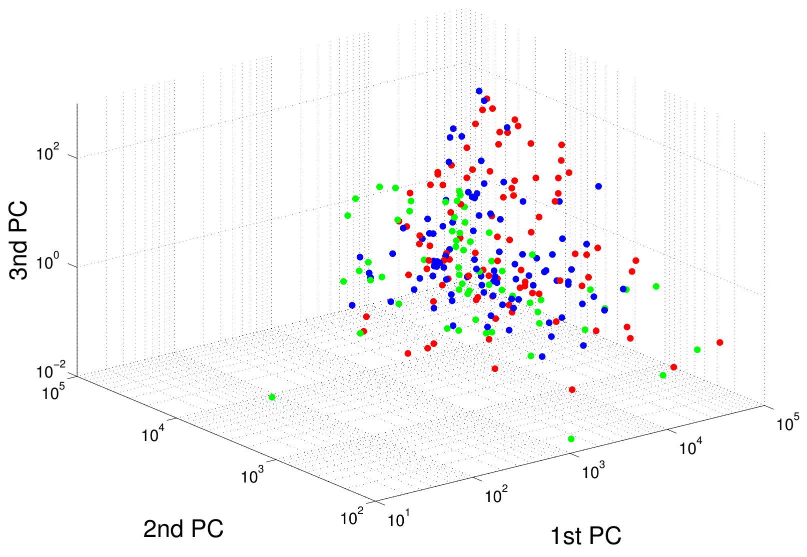


Figure 2.5: Three first principal components of the real data. All axes are in logarithmic scale.

Chapter 3

Adaptive Interactive Learning: A Novel Approach to BCI Training

3.1 Traditional BCI Training

The term “Brain-Computer Interface” denotes a wide variety of systems in which a computer and a human brain are connected in some way and exchange information. The most common meaning implies a computer which tries to perform some actions based on the information coming from the brain. We assume that a *test subject*, who is producing the signal, purposefully tries to tell the computer to perform a certain action. Relying on that assumption, the computer tries to identify which action should be performed to satisfy the test subject.

The common pipeline for training a Brain-Computer Interface is the following:

1. Signal retrieval is a step where quality and possibilities of the hardware play an important role.

2. Signal pre-processing, during which various types of feature extraction approaches are attempted in order to find the best suitable one. This step is rather data-specific, the methods which work well for a particular dataset can work worse for an other one.

3. Classification. Labelled instances of the signal are given to an algorithm to create a model. The variety of methods used here is enormous, but the high-level idea of the whole process is the asame.

Usually, improving BCI performance is attempted by trying different classification algorithms, adjusting algorithm parameters, changing feature extraction methods and so on. Most of the methods rely on the supervised learning paradigm. In this

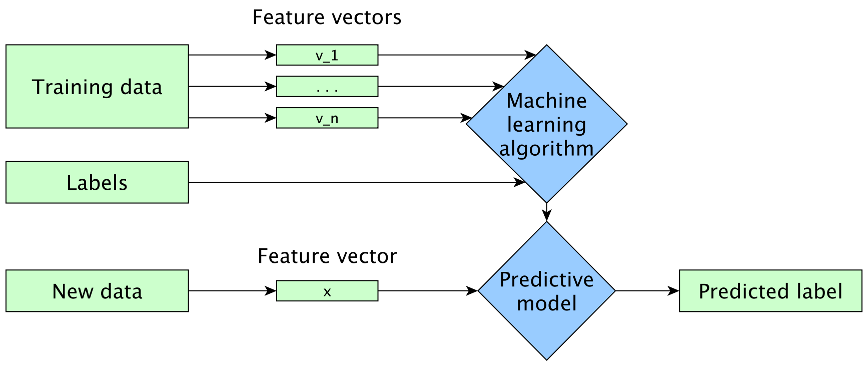


Figure 3.1: Illustration of traditional machine learning approach.

work we will refer to it as the *traditional method*, as a counterpart to the approach we propose.

Over the recent 20 or so years numerous approaches have been proposed for EEG signal classification for BCI systems. Table 3.1 describes the classification task the different authors were trying to solve, information on the hardware they have used, classification method they applied, test subject's training period and the achieved result.
There also exists a quite extensive review paper [LCL+07] on classification algorithms for EEG-based BCI. According to the summary presented there, the state of the art accuracy of mental state classification is ≈ 0.8 for two-class tasks and ≈ 0.7 for three-class tasks.

Despite the usage of elaborate algorithms and plenty of experimental data, test subjects still have difficulties controlling BCI systems even with small numbers of mental states.

3.2 The Novel Approach

A typical question one can hear from the test subject is "How should I think?". This is a natural question, which is also a source of scientific debate. Indeed, which kind of mental activity produces the most recognizable signals? Some [HKI09] propose that the test subject can provide the best feedback by imagining motor activity. Thinking of moving one's left arm can be detected by the learning algorithm more easily than the attempt to imagine some abstract notion of "left". Another approach is to come up with thoughts from different domains: recalling an image of something, imaging muscle tension, performing calculations in the mind, experience anger, recalling certain piece of music, etc. Since functional roles of the different areas of the brain are believed to be different [HIT+03] [GFF+93] [BKM+91], thus triggering listed types of mental activities will result in increasing brain activity in different areas, which can be detected by the machine and there-

Classification task	**Device**	**Method**	**Period**	**Results**
Distal or proximal movement of body parts (finger, shoulder): 2 classes [ŠSS03]	-	HMM	-	Accuracy 0.8
Imagining left or right hand movements: 2 classes [LC02]	-	PCA, HMM, LDS	-	Accuracy 0.775
Hand opening, closing + neutral: 3 classes [EE04]	256 Ag/AgCl electrodes	ICA	30 days	Accuracy 0.85
Imagining motor activity, left, right index finger, foot: 3 classes [MGPF99]	128 Hz, 56 Ag/AgCl electrodes, 0.15 - 60 Hz	Common Spatial Filters	1 hour	Accuracy 0.89
Left and right: 2 classes [HP00]	-	Time-dependent NN	1 hour	0.86
Left and right: 2 classes [TGP04]	128 Hz, 0 - 50 Hz	Graz-BCI [PNG+00]	-	AUC 0.81
Left and right: 2 classes [PKN+96]	-	NN	2 days	Accuracy 0.89
Left and right index finger movement: 2 classes [GEV03]	128 Hz	SVM	-	Accuracy 0.86
Left hand, right hand, foot imaginary movements: 3 classes [TAM07]	1000 Hz, 128 electrodes	Logistic Regression	-	Accuracy ≈ 0.5

Table 3.1: List of several articles where classification of the EEG data was attempted. HMM is Hidden Markov Models, PCA is Principal Component Analysis, LDS is Linear Dynamic System, ICA is Independent Component Analysis, NN is Neural Network, AUC is Area Under Curve, SVM is Support Vector Machines.

fore used to differentiate between mental states. There are studies indicating that certain types of activity suits BCI purposes better than other [CS03], however those results cannot be extrapolated to every human being and not everyone will be able to adopt specified mental activities to use them as triggers for the certain actions of the machine under control.

This leaves us with the necessity to explore each test subject's mental state space separately to be able to find suitable thoughts. This process is usually disguised inside the learning phase and is performed by the method of trial and error: the test subject blindly makes attempts to satisfy the machine.

In this thesis we propose a scheme, which analyzes the test subject's brain activity in real time and assists him in finding out which mental states are suitable and which are not. We see the learning process not as a one-directional flow of information from the brain to the machine, but as a duplex interaction scheme, during which both parties adapt their behavior depending on the received feedback. In the process of exploring the test subject's mental state space our system also creates a predictive model, which can be used in BCI. We will refer to our approach as the *adaptive method*, since the test subject and the algorithm can adapt their behavior in real time.

3.2.1 Interactive adaptive learning

As the name suggests, our scheme facilitates interaction between the human and the machine and allows them to adapt with respect to the information gained through interaction. Human sends out his brain signal in response to the stimulus provided by the machine, receives the feedback about how successful his mental efforts are and tries to adapt them accordingly. The machine creates a predictive model, monitors the incoming signal with respect to the model, and, depending on the results, provides the feedback, updates the predictive model and makes the decision if the model became inadequate and must be recomputed to incorporate latest data from the human.

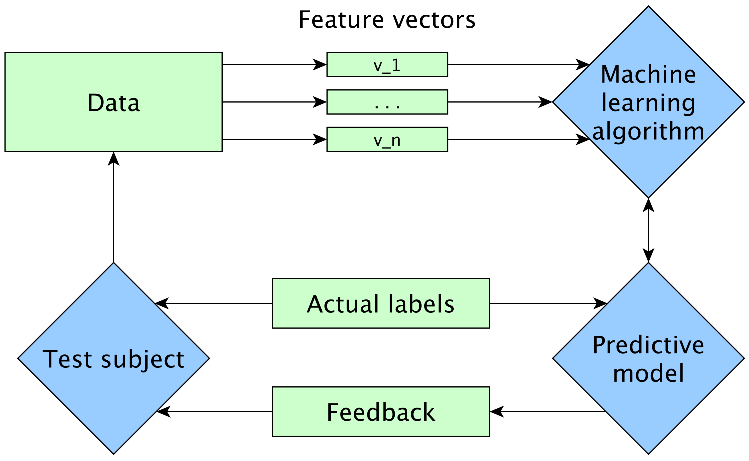


Figure 3.2: Illustration of interactive learning approach.

In the next sections we describe our realization of the interactive learning approach.

3.2.2 Predictive SOM

In our approach we rely on an extension to SOM which we refer to as *Predictive SOM*. Predictive SOM is built in the same way as the usual SOM described in Section 1.3.5, but each unit of the map has an additional vector $\mathbf{p}(u) \in \mathbb{R}^a$ where a is the number of actions. This vector holds action probability distribution for the unit u. It shows what is the probability that a signal $\mathbf{x}$, which was classified into unit u, has been produced in response to the action a.

Initialization

The only parameter we have to decide on is the size of the map. A map too small will not be able to incorporate a sufficient variety of mental states, causing quite different mental efforts to be classified into the same unit. A map too large will distribute the data sample too thinly and will consider even quite similar signals to originate from different mental states. By running the simulation with the different sizes of the map we empirically estimated that the adequate size of the map should be calculated as $10 \cdot a$, where a is number of actions (stimuli) we are going to train.

Update procedure

In predictive SOM there are two sets of vectors to update. Each unit's weight vector $\mathbf{w}(u)$ is updated as described in Section 1.3.5. Vectors $\mathbf{p}(u)$ are updated as follows:

$$\mathbf{p}^{s+1}(u) = \mathbf{p}^{s}(u)(1-\alpha) + \mathbf{c}\alpha \quad (3.1)$$

where s is iteration number, $\alpha \in (0,1)$ is a parameter, which specifies how fast the contribution of the older data samples deteriorates, and $\mathbf{c}$ is a bit vector, where for each class we have value 0 or 1. There can be only one non-zero value in the vector $\mathbf{c}$ and its position indicates the actual class.

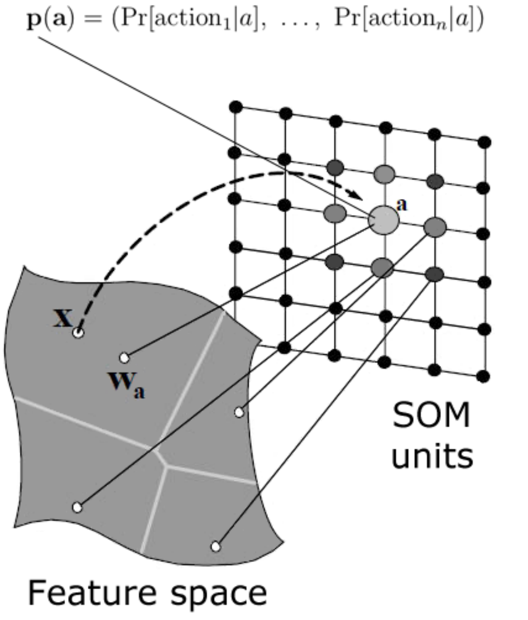

$\mathbf{p}(\mathbf{a}) = (\Pr[\text{action}_1|a],\ \ldots,\ \Pr[\text{action}_n|a])$
a
X
$\mathbf{W_a}$
SOM units
Feature space

Figure 3.3: SOM with additional probability vectors.

Classification procedure

The probability vector $\mathbf{p}(u)$ can be used for classification. For a sample $\mathbf{x}$ we want to classify, we identify SOM's BMU (best matching unit, see Definition 1.1) u for this sample, and predict the class of this sample by choosing the most probable class in the vector $\mathbf{p}(\mathbf{u})$. We can also regard probabilistic SOM as a probabilistic prediction model, where the prediction for sample $\mathbf{x}$ is a probability distribution over all possible classes.

3.2.3 Implementation of the interactive training process

The process starts by recording the test subject's brain signal for a fixed short period of time. Then the signal is transformed as described in Section 1.1.3. This initial data is used to create the initial SOM as discussed in Section 1.3.5. The probability vectors $\mathbf{p}(u)$ are initialized with uniform distribution.

After that the main phase of the learning process begins. The system shows sequences of stimuli to the test subject. Each stimulus is shown for a fixed period of time, during which several data samples are recorded and processed. Each sample is classified according to the existing model and the feedback is shown to the test subject. Figure 3.4 demonstrates the signal processing step and shows how the feedback message is produced in both successful (green) case, and misclassification (red) case.

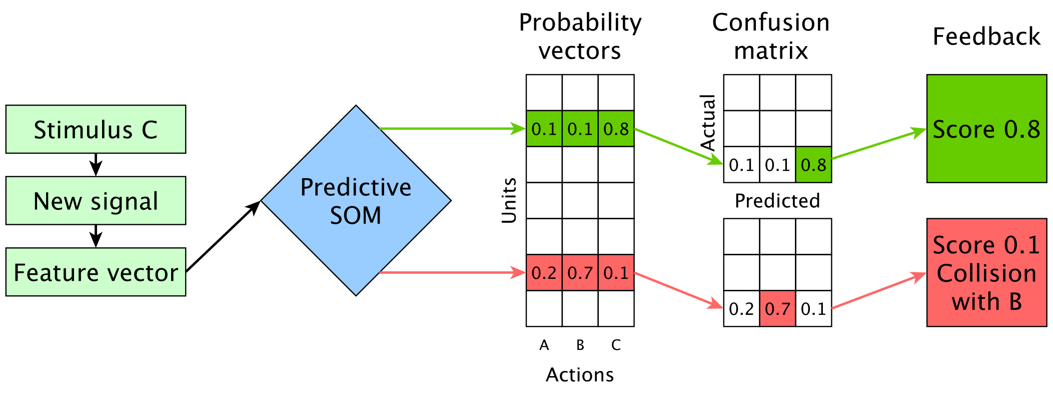


Figure 3.4: Sample classification pipeline. Path in the case of a correctly classified sample is show in green, a misclassified sample case in red.

As the new sample $\mathbf{x}$ for action a arrives, predictive SOM classifies it into unit u and reports success if $\mathbf{p}(u)_a = \max(\mathbf{p}(u))$ and collision otherwise. Feedback f is a number in the range from 0 to 1 and $f = \mathbf{p}(u)_a$. If the estimated probability of action a hitting unit u is high then we report a high score, since the test subject was able to produce a suitable data sample and the machine correctly assigned a SOM unit to it. If the probability is low, then either the test subject evoked a mental state, which is not suitable or the machine has misclassified the sample.

Evaluating model performance

Model's performance is being tracked using a probabilistic confusion matrix C as described in Section 1.3.7. The probability vector $\mathbf{p}(u)$ is added to the a'th row of the confusion matrix, where a is the stimulus, which was shown to the test subject. After the system has shown the stimulus for the action a to the test subject, he produces a signal s. Figure 3.4 illustrates the confusion matrix update procedure. In case of a correct prediction the matrix update $\mathbf{C_a} = \mathbf{C_a} + \mathbf{p}(u)$ mostly contributes to the main diagonal of the confusion matrix and the overall F1 score grows. In the case of misclassification the biggest contribution goes off the main diagonal, which brings the F1 score down.

Examples below demonstrate the effect on the confusion matrix and the resulting F1 score in three major cases:

a) **Random: unit u receives approximately almost same amount of samples from actions other than action a.**
Assume that we have 3 actions in total, $a = 3$ and a unit's u action distribution vector $\mathbf{p}(u) = \left(\frac{1}{3}, \frac{1}{3}, \frac{1}{3}\right)$. Assume the current confusion matrix and F1 score are

$$\text{F1}\begin{pmatrix} 10 & 2 & 1 \\ 1 & 8 & 2 \\ 1 & 2 & 10 \end{pmatrix} = 0.7582\ .$$

Now we add $\mathbf{p}(u)$ to the a'th row of the probabilistic confusion matrix:

$$\text{F1}\begin{pmatrix} 10 & 2 & 1 \\ 1 & 8 & 2 \\ 1.33 & 2.33 & 10.33 \end{pmatrix} = 0.7472\ .$$

Adding $\frac{1}{3}$ to main diagonal will make the score better, but at the same time it will be also penalized for the fact that this is a pure accident and with the same chance the update could have happened off the main diagonal.

b) **Misclassification: almost none of the samples that unit u receives are produced during action (stimulus) a.**
This time the signal s for action a was put into unit u, where it has been never put before and which is associated with by other actions. Vector $\mathbf{p}(u)$ in this case is $\left(\frac{9}{10}, \frac{1}{10}, 0\right)$ and the resulting confusion matrix after update is

$$\text{F1}\begin{pmatrix} 10 & 2 & 1 \\ 1 & 8 & 2 \\ 1.9 & 2.1 & 10 \end{pmatrix} = 0.7376\ .$$

We see that F1 score decreased more than in the previous case.

c) **Correct prediction: most of the samples unit u receives are produced during action (stimulus) a.**
This is a positive example, the vector $\mathbf{p}(u) = \left(\frac{1}{10}, 0, \frac{9}{10}\right)$ and updating the third row of the confusion matrix makes the F1 score larger:

$$\text{F1}\begin{pmatrix} 10 & 2 & 1 \\ 1 & 8 & 2 \\ 1.1 & 2 & 10.9 \end{pmatrix} = 0.7618\ .$$

To get a better picture during training, we track two slightly different confusion matrices.

The overall confusion matrix W is computed from all the samples processed since the last update of the model. It thus reflects the global situation. When the F1 score of this matrix drops below the threshold we consider the model to be inadequate and rebuild it using all new data accumulated during the learning process. It is important to mention, that since SOM does not rely on the labels of the instances, the test subject can freely investigate his mental states space, with no danger of producing "wrong" data samples.

The recent confusion matrix R remembers only the results from the fixed number of last data samples. It reflect the more recent situation. This matrix carries two functions:

1. By observing only recent performance we can finish the learning process as soon as model becomes good enough and we do not have to wait until global confusion matrix W will also reflect this fact.
2. When the system has to pick next stimulus to show to the test subject it chooses the one, which has recently received less positive feedback than the other stimuli.

Measuring the test subject's performance

Conceptually, we can see the probability vectors $\mathbf{p}(u)$ as reflection of the test subject's efforts to explore his mental state space. On Figure 3.5 we visualize an example of all probability vectors as one matrix where probability values are coded with color.

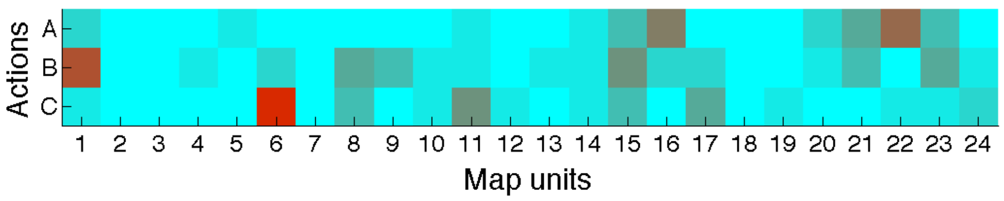


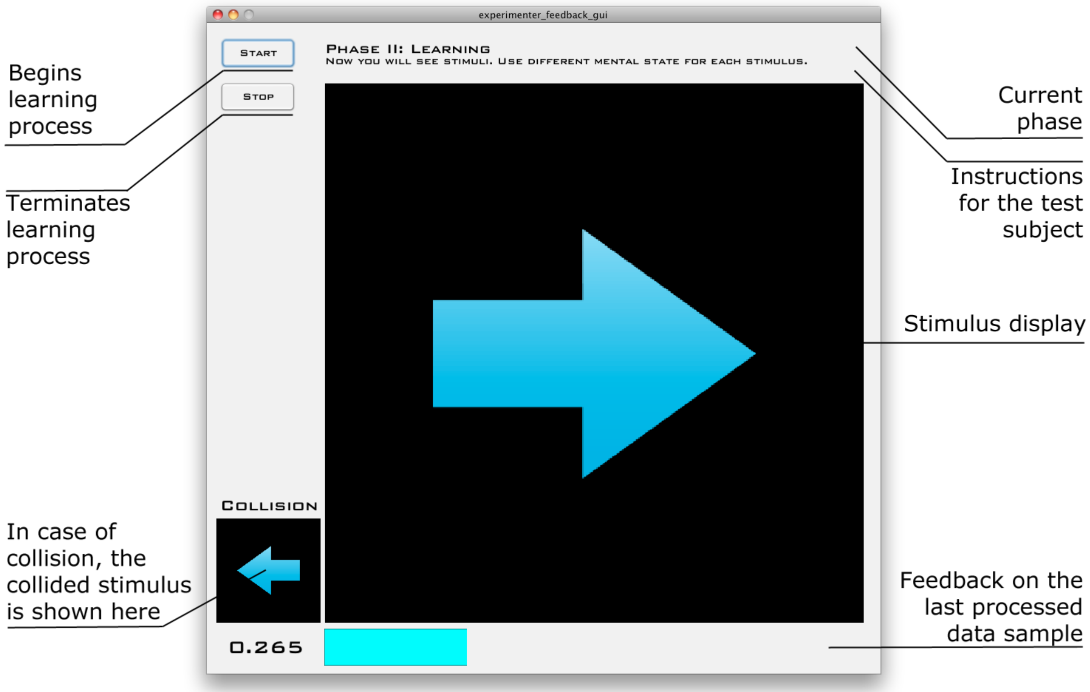
Figure 3.5: Visualization of the probability vectors for predictive SOM with 3 actions 24 map units.

We were always talking about the action distribution in the units. Just as an observation we would like to notice that, simultaneously, the unit distribution over the actions is being formed in the process of learning. For example on Figure 3.5 action A will be represented by the units 5, 16 and 22; action B by the units 1, 4, 13 and 15; action C by the units 6, 11, 17 and 19.

Experimenter

The experimenter is the main GUI application, which incorporates all the logic described in this section. It interacts with the test subject, shows stimuli, provides the feedback, tracks performance and makes decisions about recomputing the model or entering evaluation phase when the model has achieved success threshold. This later phase is also facilitated by the Experimenter.

Figure 3.6 demonstrates the Experimenter window during second phase. Top part of the screen contains an instruction for the test subject. Most space is occupied by the stimulus display. In the bottom part of the screen we can see the feedback score, collided stimulus if there is one and the color gauge, which visualizes the score.


experimenter_feedback_gui
Start
Stop
Phase II: Learning
Collision
0.265
Begins learning process
Terminates learning process
In case of collision, the collided stimulus is shown here
Current phase
Instructions for the test subject
Stimulus display
Feedback on the last processed data sample

Now you will see stimuli. Use different mental state for each stimulus.

Figure 3.6: Main window of the Experimenter application.

Real-time processing

All computations are made in real time facilitating transmission of relevant information between human and machine and providing adaptive learning process. We use Matlab® as our primary development environment. This choice is justified by the availability of powerful toolboxes for EEG signal processing designed for use with Matlab. FieldTrip [OFMS11] is one of them. This toolbox has module for communicating with the Emotiv EPOC. It allows to transmit data from the device into Matlab environment in real time.

3.2.4 A scheme to summarize

The scheme on Figure 3.7 depicts all processes going on in the system. Time flows from top to bottom, boxes contain states or milestone processes, arrows denote transitions between the states. The transitions can be triggered by the system internal logic, results of the calculations or by the instructions issued by the graphic user interface (GUI) of the system.

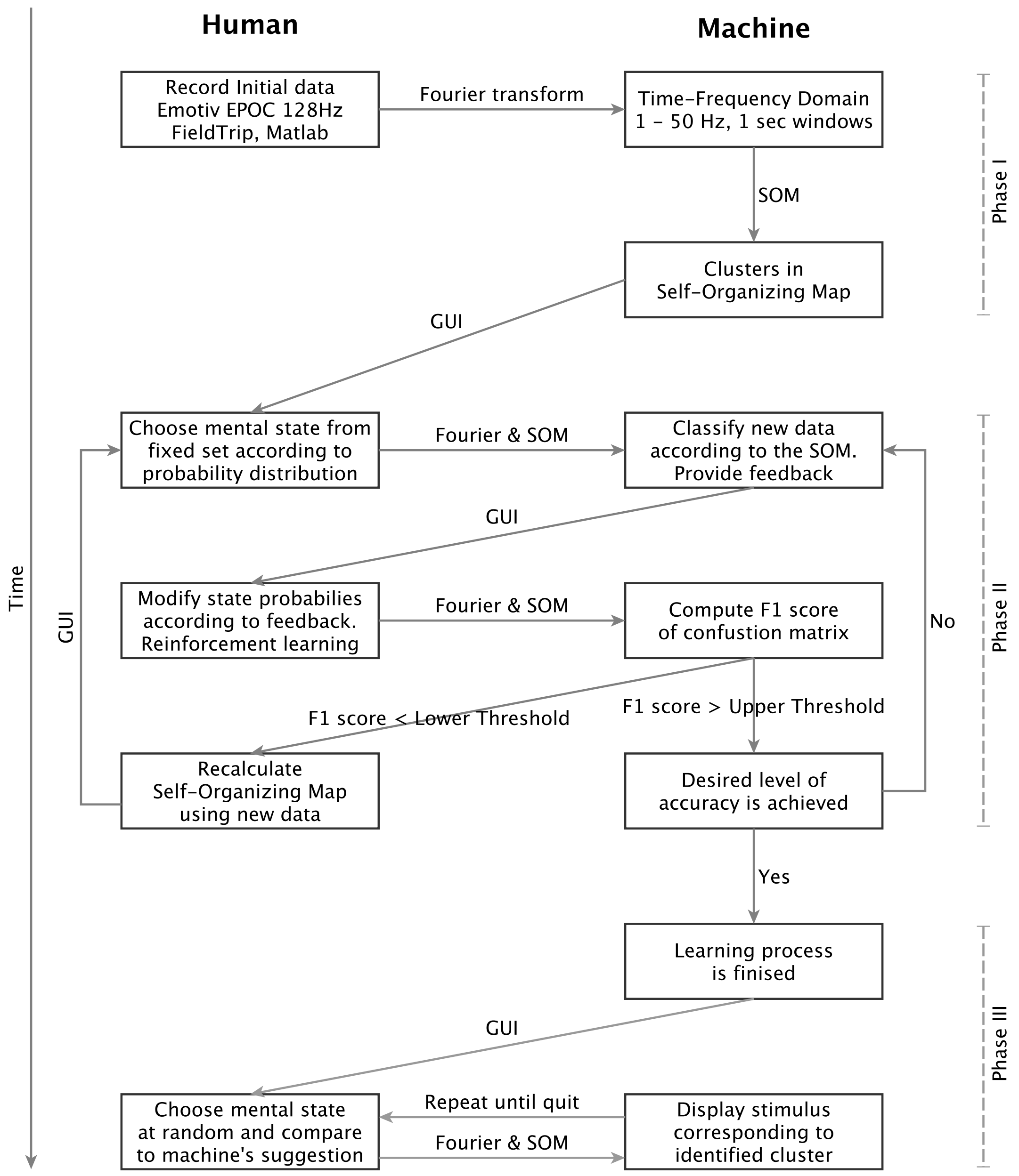

Human
Machine
Record Initial data Emotiv EPOC 128Hz FieldTrip, Matlab
Fourier transform
Time–Frequency Domain 1 – 50 Hz, 1 sec windows
SOM
Clusters in Self–Organizing Map
Phase I
GUI
Choose mental state from fixed set according to probability distribution
Fourier & SOM
Classify new data according to the SOM. Provide feedback
GUI
Modify state probabilies according to feedback. Reinforcement learning
Fourier & SOM
Compute F1 score of confustion matrix
No
Phase II
Time
GUI
F1 score < Lower Threshold
F1 score > Upper Threshold
Recalculate Self–Organizing Map using new data
Desired level of accuracy is achieved
Yes
Learning process is finised
GUI
Phase III
Choose mental state at random and compare to machine's suggestion
Repeat until quit
Fourier & SOM
Display stimulus corresponding to identified cluster

Figure 3.7: Detailed diagram of the interaction scheme.

Chapter 4

Experiments on Artificial Data

Before we test our model on real data we would like to test and fine-tune it on artificial data. In this section we describe how simulated experimental setup works and how it helps to formalize our goals and assumptions.

4.1 Brain Signal Simulation

4.1.1 Generating EEG-like data

To generate data with noise and component composition similar to real EEG readings we use supplementary material to [YBHC04], which includes Matlab code of the EEG-like signal generator the authors used in their experiments. Artificial data is being generated according to the *Event Related Potentials* theory – classical theory on the nature of EEG signals [Luc05].

Figure 4.1 show a snapshot of a running 10-second spectrum window on channels `F3`, `P7` and `T8`.

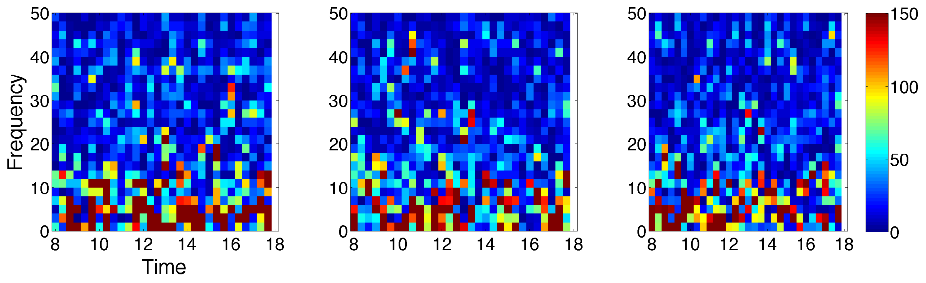


Figure 4.1: Running spectrum of generated EEG-like data without mental states introduced.

4.1.2 Introducing components into noisy data

In our simulation we exploited the abstract notion of mental state space, and defined a fixed number of mental states the simulated "test subject" can use. Mental states are defined by introducing 40 Hz sinusoidal component of amplitude 40 (same amplitude as the amplitude of the noise in the EEG signal generator) into the predefined channels of the signal. Table 4.1 shows exactly which channels are mapped to which mental states.

#	Area	Channels	Corresponding electrodes
0	All (noise only)	1-14	`AF3, F7, F3, FC5, T7, P7, O1, O2, P8, T8, FC6, F4, F8, AF4`
1	Left frontal lobe	1-4	`AF3, F7, F3, FC5`
2	Frontal lobe	1-4, 11-14	`AF3, F7, F3, FC5, FC6, F4, F8, AF4`
3	Right frontal lobe	11-14	`FC6, F4, F8, AF4`
4	Left hemisphere	1-7	`AF3, F7, F3, FC5, T7, P7, O1`
5	All	1-14	`AF3, F7, F3, FC5, T7, P7, O1, O2, P8, T8, FC6, F4, F8, AF4`
6	Right hemisphere	8-14	`O2, P8, T8, FC6, F4, F8, AF4`
7	Left parietal and occipital lobes	5-7	`T7, P7, O1`
8	Parietal and occipital lobes	5-10	`T7, P7, O1, O2, P8, T8`
9	Right parietal and occipital lobes	8-10	`O2, P8, T8`

Table 4.1: Nine artificial mental states are defined by the channels where the sinusoidal component will be introduced to.

Mental states can overlap, making it more complex for the machine to distinguish between them. By overlapping we simulate existence of mental states, which can seem different to the test subject, but actually are quite similar in terms of the produced signal.

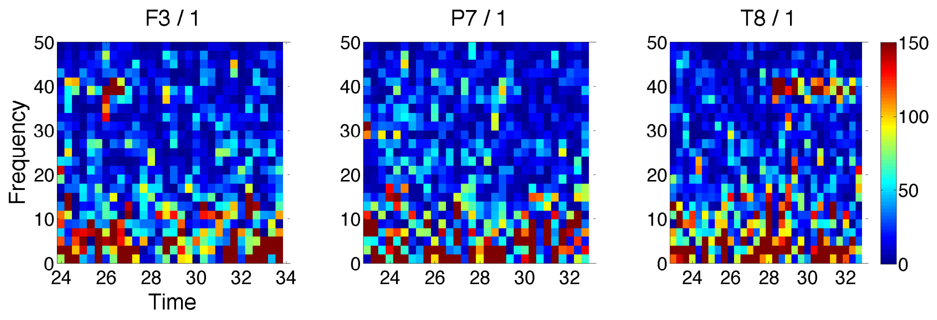


Figure 4.2: Running spectrum of generated EEG-like signal with introduced artificial mental states.

On Figure 4.2 we can see the result of combining the generated signal from Section 4.1.1 and introduced components. During the first 5 seconds we activated mental state `1`, which introduces the components into the channels `AF3, F7, F3, FC5`. As we can see, channel `F3` indeed has elevated activity on frequency of 40 Hz.

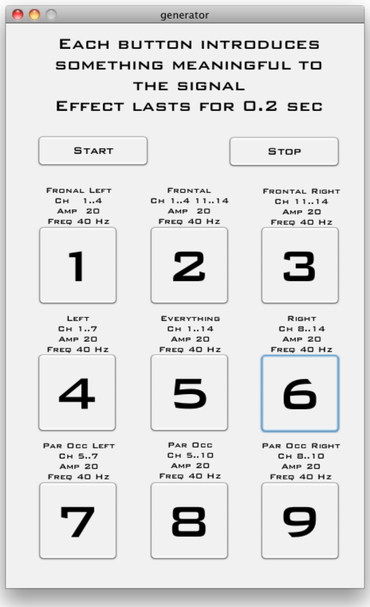
During the last 5 seconds we activated mental state `9` and again the activity on the 40 Hz band can be observed on the channel `T8` as expected.

4.1.3 Generator

The application, which implements the process of generating EEG-like signal and allows to manually activate any of the mental states is called `Generator`. We have assigned nine mental states described in Section 4.1.2 to nine buttons, which are numbered accordingly. Figure 4.3 depicts the generator application.

The generator writes the resulting signal into the same signal buffer as the Emotiv EPOC, which makes it possible to use the simulated signal with the Experimenter application.

Please see the Appendix A for the instructions on how to run this application.



EACH BUTTON INTRODUCES SOMETHING MEANINGFUL TO THE SIGNAL
EFFECT LASTS FOR 0.2 SEC

generator
START
STOP
FRONAL LEFT CH 1..4 AMP 20 FREQ 40 HZ
FRONTAL CH 1..4 11..14 AMP 20 FREQ 40 HZ
FRONTAL RIGHT CH 11..14 AMP 20 FREQ 40 HZ
1
2
3
LEFT CH 1..7 AMP 20 FREQ 40 HZ
EVERYTHING CH 1..14 AMP 20 FREQ 40 HZ
RIGHT CH 8..14 AMP 20 FREQ 40 HZ
4
5
6
PAR OCC LEFT CH 5..7 AMP 20 FREQ 40 HZ
PAR OCC CH 5..10 AMP 20 FREQ 40 HZ
PAR OCC RIGHT CH 8..10 AMP 20 FREQ 40 HZ
7
8
9

Figure 4.3: Generator GUI allows to introduce mental states into the signal buffer.

4.2 Comparing Adaptive and Traditional Learning Methods in the Artificial Setting

The models created using different models can be evaluated based on the F1-score (see Section 1.3.7). If we assume that mental states are separable, then both models can achieve desired F1-score value if they are given enough data and parameters are correct. We set up the *success threshold* and define the *model performance* as a number of samples the model needs to process in order to achieve this threshold.

Definition 4.1. *Success threshold* is a real number in the range $0 \leq t \leq 1$. We say that a model has achieved the desired performance if F1-score of the confusion matrix is $\geq t$.

Definition 4.2. *Model performance* is an average number of data samples that a model requires in order to reach the success threshold.

4.3 Traditional Simulator

In this section we describe how we simulate the traditional approach and the behavior of the test subject during the traditional learning process. We describe the assumptions we made in order to be able to formalize the process and the simulated test subject.

4.3.1 Simulating traditional behavior

The first thing the test subject has to do is to choose the potential mental states, which he is able to consciously activate, switch between them and sustain for some period of time. In Section 3.2 we have discussed the possible ways he can make this decision. Regardless of the approach, the test subject will end up with a fixed number of mental states and the mapping between them and the stimuli. It does not matter, which mental states are mapped to which actions, therefore we can make this choice randomly in each run of our simulation.

The simulated system has $k = 3$ actions $\{A, B, C\}$. The simulated test subject has $l = 9$ mental states $\{m_1, \dots m_9\}$ we have defined in Table 4.1. Each action is associated with $\lfloor k/l \rfloor$ mental states. Since the real test subject is not in full control of the signal he emits, we allow overlaps in our simulation: the same mental state can be assigned to one or more actions. An adaptive scheme would help to resolve such an overlap, but in the traditional setting the test subject will not receive the necessary information to do that. Table 4.2 gives an example of the mapping between the actions and the mental states.

	Mental states								
	$\mathbf{m_1}$	$\mathbf{m_2}$	$\mathbf{m_3}$	$\mathbf{m_4}$	$\mathbf{m_5}$	$\mathbf{m_6}$	$\mathbf{m_7}$	$\mathbf{m_8}$	$\mathbf{m_9}$
Actions	A		A		A				
			B	B					B
		C				C	C		

Table 4.2: Mental states $\{m_1, \dots, m_9\}$ are mapped to the actions $\{A, B, C\}$.

When the simulated test subject sees a stimulus $a \in \{A, B, C\}$ he randomly choses one of the mental states mapped to this action and produces the corresponding signal.

4.3.2 Traditional simulator implementation

The traditional model will benefit from every piece of labelled data it can get. For this reason we recompute the model not when it reaches the threshold, but on a regular basis. In our implementation we chose to recompute the model after every 50 samples of new data.

Classification is done using SVM algorithm with parameters fine-tuned using parameter search.

The performance of the model is measured in terms of F1 score of the confusion matrix. The matrix is being updated in a trivial manner: each correctly classified sample adds 1 to the main diagonal, a misclassified sample adds 1 off the main diagonal, into the cell M_{ap} where a is the index of the actual action and p is the index of the predicted action.

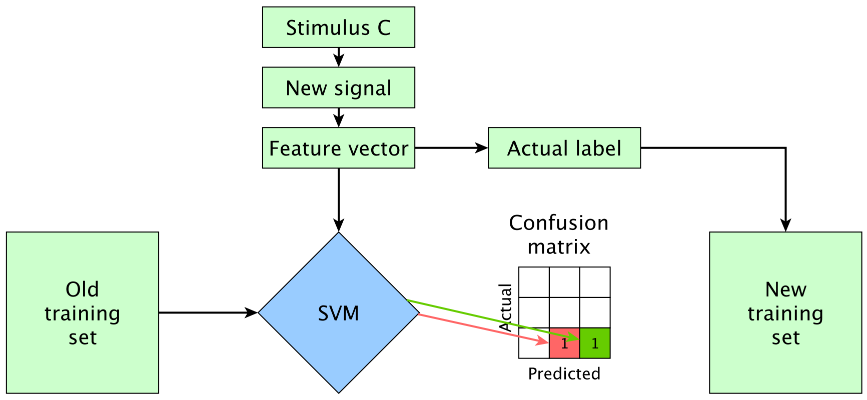


Figure 4.4: In the traditional simulator each new data sample is treated as test sample and after evaluation is added to the training set.

Usually, supervised learning algorithms are being evaluated on a separate data set, called *training set*. In our implementation each new sample first is being treated as a test sample, meaning that we evaluate the model using this sample and after that the same sample with a correct label in being added to the training set. Figure 4.4 illustrates this process.

Listing 4.1: Traditional Behavior Simulator: example output

```
 1 Action 1
 2 Picked mental state 7
 3
 4 Confusion matrix for whole lifetime of the most recent model
 5       1      0      0
 6       0      8      1
 7       2      2      6
 8
 9 Last cycles confusion matrix
10      12      1     18
11       0     15      5
12       3      3     14
13
14 If 0.90000 < 0.58286 => Learning complete
15
16 Number of samples 95
17 Recompute counter 20
```

Listing 4.1 provides an example output of the traditional simulator. Line 1 shows the stimulus, which is currently "shown" to the simulated test subject. Line 2 says which of the mental states was picked by the test subject in response to the stimulus. The matrix on the lines `5-7` is a confusion matrix of the predictions done since the last model update. The matrix on the lines `10-12` is the one, which is used to calculate F1 score and decide whether the learning process is complete.

Please see Appendix A for instructions on how to run this application.

4.3.3 Results

Trivial case

First we perform the most simple type of an experiment, in which we do not add EEG-like noise to the simulated data. Thus, the training data consists purely of the introduced sinusoidal components. This trace is trivial, because it should be very easy to classify the instances in such setting. However, as we mentioned in Section 4.3.1, mental states overlap, which can make it difficult to distinguish one from another. As we can see on Figure 4.5 the traditional simulator does not always converge. This is the behavior we expected to get and a good illustration of the issue we are facing in the real learning process: without feedback the test subject does not know whether chosen mental states are adequate, and without adapting his behavior he continues to produce non-classifiable samples, which leaves the model in the state of stagnation no matter how many data samples are provided to it. Few runs which do converge are the cases, where random initialization of state-to-action mapping happened to assign distinguishable mental states to the actions and, therefore, the model was able to discriminate.

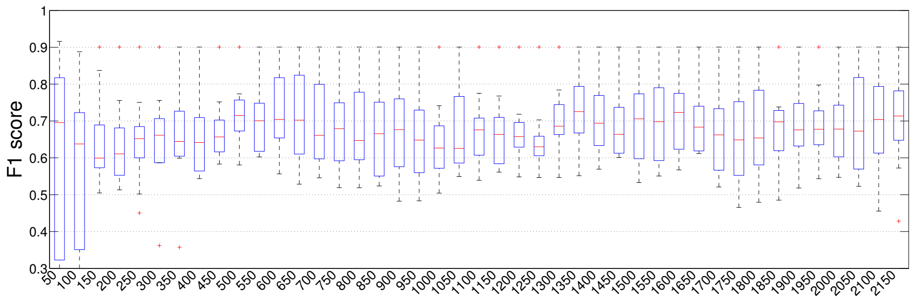


Figure 4.5: Model performance and trace of the changes in F1 score for the traditional simulator on noiseless data. Average F1 score for last 500 samples for this experiment is 0.6875. Data is collected over 10 simulator runs.

Noisy artificial data

Noisy artificial data is closer to the real task and harder for the learning algorithm. The results show that the period of stagnation begins later, since model needs more

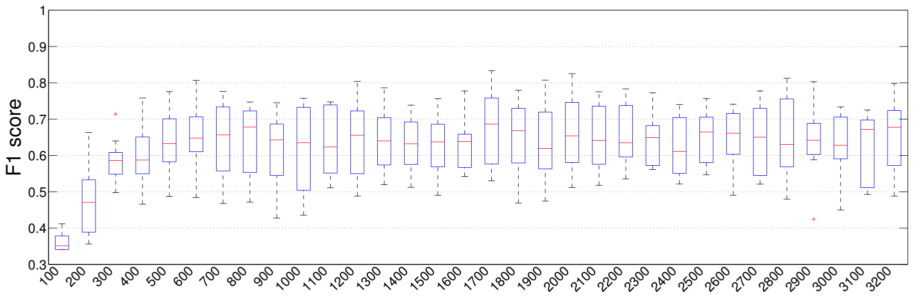


Figure 4.6: F1 score trace of the traditional simulator on noisy artificial data. Data is collected over 10 simulator runs.

data to clear out the noise. After a certain level has beed achieved the model does not improve any more and the maximal achievable F1-score is lower. The average F1 score for last 500 samples is 0.6423. The difference is statistically significant. Comparing the last 500 F1 score readings between the trivial and the noisy cases yields p-value $< 2.2 \times 10^{-16}$. For the same reasons as in the trivial case, the model does not reach the desired F1 score level of 0.9. Presence of noise does not make the task easier.

4.4 Adaptive Agent Simulation

4.4.1 Simulating adaptive behavior

Behavior of an adaptive test subject is very different from the traditional one. Since he receives feedback, he has the ability to alter the mapping between the mental states and the actions. The positive feedback will encourage him to continue on using certain mental states for the certain actions, while negative feedback will allow to understand which mental states are not suitable, or are in conflict with some other mental states. The simulated test subject's behavior is described by the *probability matrix*, which holds the probabilities for invoking a certain mental state in response to a certain action.

Definition 4.3. The *probability matrix* M is an integer matrix of size $a \times m$ where a is the number of actions and m is the number of mental states in the imaginary mental state space. Each column represents the probability distribution for certain mental state to be invoked in response to the stimulus.

The probability matrix is initialized with uniform mental state distribution for each of the actions. This corresponds to the test subject's zero knowledge of his mental state space in the beginning of the process. When the mental state m is being evoked in response to the action a we adapt the simulated test subjects behavior by incrementing or decrementing the probability score of the action a

for the mental state m. If the stimulus predicted by the model coincides with the actual stimulus shown to the test subject we increment the probability score by 1. If the sample was misclassified we decrement the probability score by 1. Over time our model will "learn" whether engaging mental state m in response to the action a will give positive feedback or not.

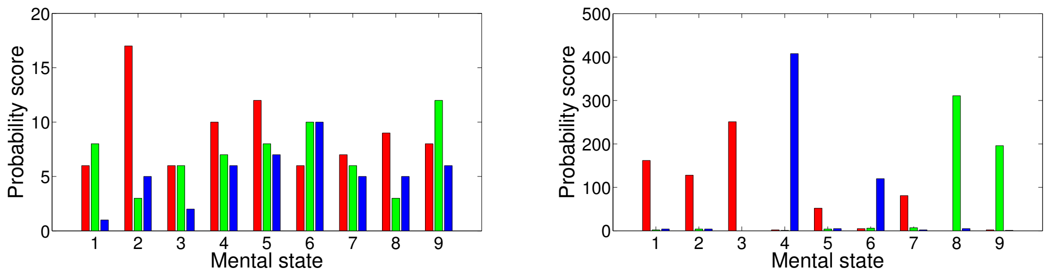


Figure 4.7: Probability matrix of the adaptive behavior simulator in the beginning (left) and in the end (right) of the learning process.

Figure 4.7 provides a visualization of one such matrix in the beginning and in the end of the learning process. In the end of the process we can see how every mental state is mapped to one particular action. Mental states 5 and 7 have received very small amount of the positive feedbacks during the learning process, which indicates that those mental states are not suitable, since they are either not distinguishable enough, or they conflict with the other states. For example we know from Table 4.1 that 5th mental state introduces the components into all channels at once, therefore it is reasonable to expect that activating this mental state will frequently cause conflicts and the test subject will be better off not engaging this mental state at all. And as we see from the figure above the adaptive test subject has successfully learned this fact and diminished the probability of activating 5th mental state for any of the actions.

4.4.2 The implementation of the adaptive simulator

The adaptive simulator has several parameters, which can be changed to tune the simulation process. Table 4.3 lists those parameters and explains their purpose.

The simulation process begins with generating initial data for each of the actions, after that each new sample is being processed, issuing the feedback and evaluating current state of the model. Listing 4.2 provides example output for the working simulator after yet another sample has been processed.

Line `1` shows the index of the action the simulator is currently training, on line `2` we see the mental state the simulated test subject have chosen in response to the current stimulus. Line `3` shows the BMU for the input vector $\mathbf{x}$. Line `6` holds the distribution (not normalized) of the actions in the current unit. Depending on the success of the predictive model line `7` provides the feedback $= \{1, -1\}$.

Listing 4.2: Adaptive Behavior Simulator: example output

```
 1 Action 2
 2 State  4
 3 Unit   26
 4
 5 Learning vector               0   1   0
 6 Unit probability vector      35   6   2
 7 Learning probability score   -1
 8
 9 Confusion matrix for whole lifetime of the most recent model
10    49.0181   14.8157   19.1662
11    17.3839   39.6657    7.9505
12    24.2423   18.1684   37.5893
13
14 Last cycles confusion matrix
15    19.1909    5.5027   10.3064
16     4.5717   15.9333    4.4950
17     7.3857    3.3878   19.2265
18
19 If 0.50000 > 0.55212 => Recompute
20 If 0.90000 < 0.60350 => Learning complete
21
22 Recompute counter 228
23 Number of samples 642
```

In the current example we can see that the most probable action for unit 26 is the first action. But since we are training action 2 at the moment, choosing unit 26 is considered to be a mistake and the system issues negative feedback. It will decrease the probability of evoking mental state 4 in response to action 2.

Parameter	**Description**
`actions`	The list of actions we want to train.
`action_duration`	The range where the first number indicates minimal duration of the stimulus (in seconds) and second indicates the maximal duration. System will randomly choose duration from this range for each stimulus. Default value is `[10 10]`, which means that all stimuli will be shown for 10 seconds.
`f1_high`	Upper threshold after which learning process should be considered successful and simulation will end.
`f1_low`	Lower threshold for recomputing the model. Whenever F1-score of the confusion matrix will drop below this value system will assume the model to be inadequate and recompute it. By default this value is given as a function of the number of actions n: $\mathrm{F1}_{low} = \frac{1}{n} * 1.5$, which means that model is considered adequate until its performance is $\frac{3}{2}$ times better than random.
`mental_states`	The list of the mental states. Each element of the list is a set of the channels where sinusoidal signal is introduced when particular mental state is activated.

Table 4.3: The parameters of the adaptive simulator implementation.

Lines `10-12` hold the confusion matrix, which accounts for all the data samples processed since the last update of the model. We can see on line `22` that the last time this happened 228 samples ago. Confusion matrix on lines `15-17` accounts only for the last 90 samples, thus it reflects the more current situation.

Lines `19` and `20` reflect the logic described in Section 3.2.3. If the F1 score of the global matrix will fall below the threshold `f1_low` = 0.5, then the model will be recomputed. If the F1 score of the recent matrix will exceed the threshold `f1_high` = 0.9, then the learning process is complete.

4.4.3 Results

Trivial case

First we measure system performance in the ideal settings. The experiment described in this section is performed on noiseless data. Each mental state introduces sinusoidal component into designated channels, and, as before, some channels do overlap. The adaptive learner is expected to get rid of the overlapping mental states and map each mental state to only one action. Figure 4.8 demonstrates the final state of the probability matrix after the desired level of model performance is achieved. We can see how each state settled on one particular action.

We measure system performance as defined in Section 4.2: performance of the model is reflected by the number of samples the system needs to process before the required level of accuracy is achieved. Box plot on Figure 4.13 depicts change of model performance over time. The data is taken over 10 simulator runs. The simulator has achieved F1-score ≥ 0.9 after 1900 samples in most cases.

It is worth noticing that all of the runs converged and the growth rate was quite stable over the runs.

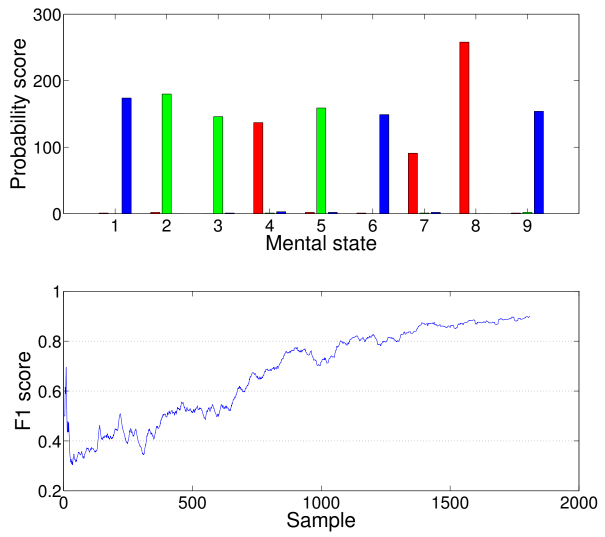


Figure 4.8: Final probability matrix (top) after F1 score $\geq$ 0.9 is achieved. The trace of the changes in F1 score (bottom).

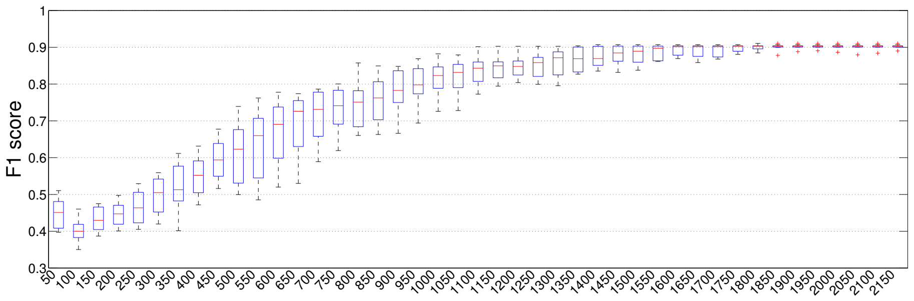


Figure 4.9: Model performance and trace of the changes in F1 score for the adaptive simulator (10 simulator runs).

Artificial noisy data

After the experiment on the trivial case has confirmed that the model works and does converge as expected, we proceed to experiments on more realistic data. The data for this experiment is generated as described in Section 4.1. Figure 4.10 demonstrates the performance.

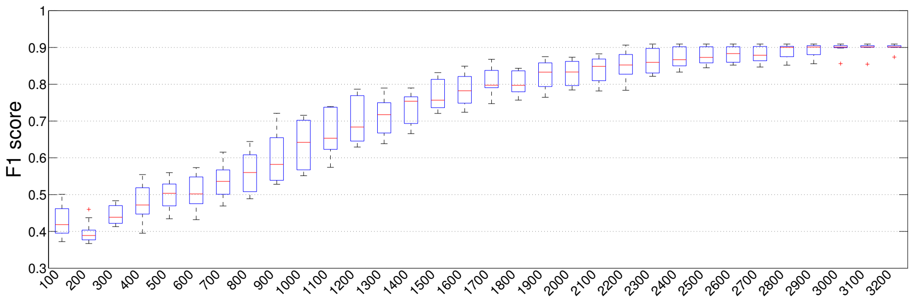


Figure 4.10: F1 score trace of the adaptive simulator on the noisy artificial data (10 simulator runs).

In comparison with the traditional approach the adaptive simulator shows much better performance. The presence of noise, as expected, made the situation worse for both methods, but whilst the traditional method got stuck on F1 score of 0.64, adaptive one was able to achieve the F1 score of 0.9, although it took more time.

4.5 Dependency Between the Number of Mental States and System Performance

In the previous sections we made some quite abstract assumptions when we simulated the test subject's brain. Probably the least realistic one is that the simulated test subject has 9 mental states. In this section we ran several experiments to see how adaptive and traditional models would behave if we changed the following parameters of the simulation:

a) The total number of mental states.
b) The total number of actions to train.
c) Both parameters simultaneously.

Experiments are conducted in the trivial (noiseless) setting.

Increasing the number of mental states

The increase in the number of mental states increased the time the adaptive simulator needed to converge. This is expected behavior, since the number of mental states directly affects the dimensions of the probability matrix, thus requiring more samples to diminish unsuitable states and select the suitable ones.

The level at which stagnation of the traditional simulator begins becomes higher with the larger number of mental states. This is explained by the fact that the chance to pick suitable mental states from a larger pool is higher.

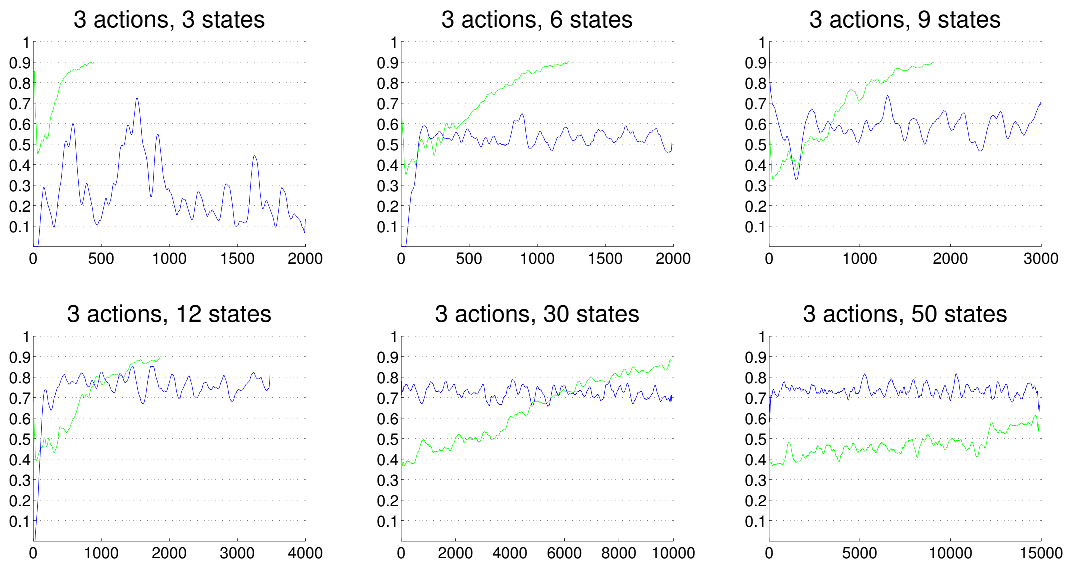


Figure 4.11: Trace of F1 score change for the traditional (blue) and the adaptive (green) simulators under the condition of increasing the number of states.

Increasing the number of actions

With a larger number of actions the chance of a conflict between the mental states increases. The adaptive simulator requires more time to converge, but converges in all experiments we conducted.

The traditional simulator does not converge and with the increase in the number of actions the maximal achievable F1 score decreases. This is the expected behavior, since a larger number of conflicts will cause more difficult cases for the classifier.

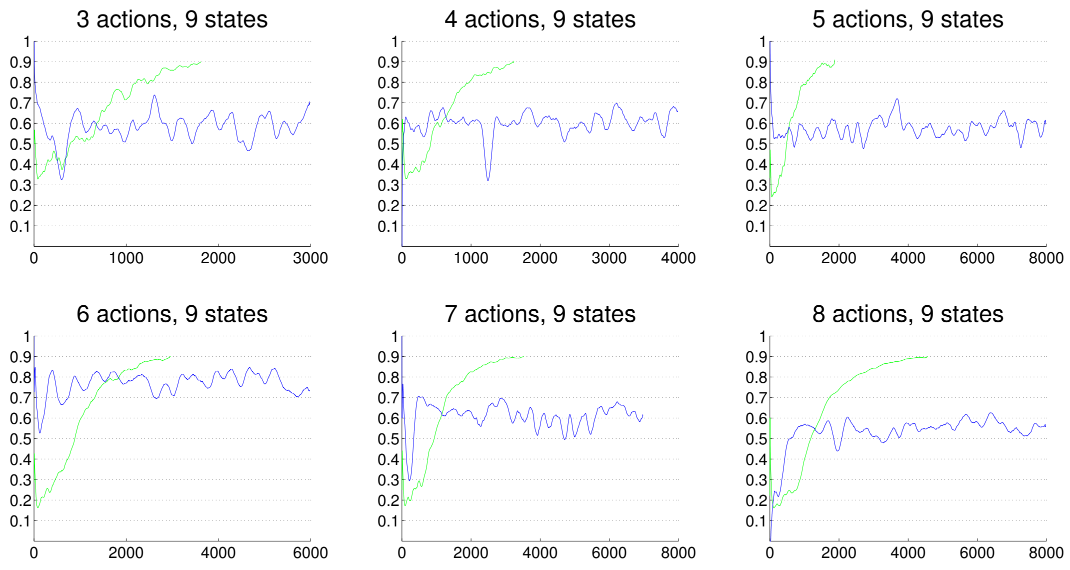


Figure 4.12: Trace of F1 score change for the traditional (blue) and the adaptive (green) simulators under the condition of increasing the number of actions.

Increasing the number of mental states and the number of actions

To check if some unexpected effect will occur when the number of mental states and of actions is large, we performed one more series of experiments.

Results are consistent with everything seen so far: higher number of actions and mental states causes the adaptive behavior to spend more time to converge, but it does converge eventually. The traditional simulator does not converge and the level of stagnation seems to be stable if the number of mental states and the number of actions are linearly correlated.

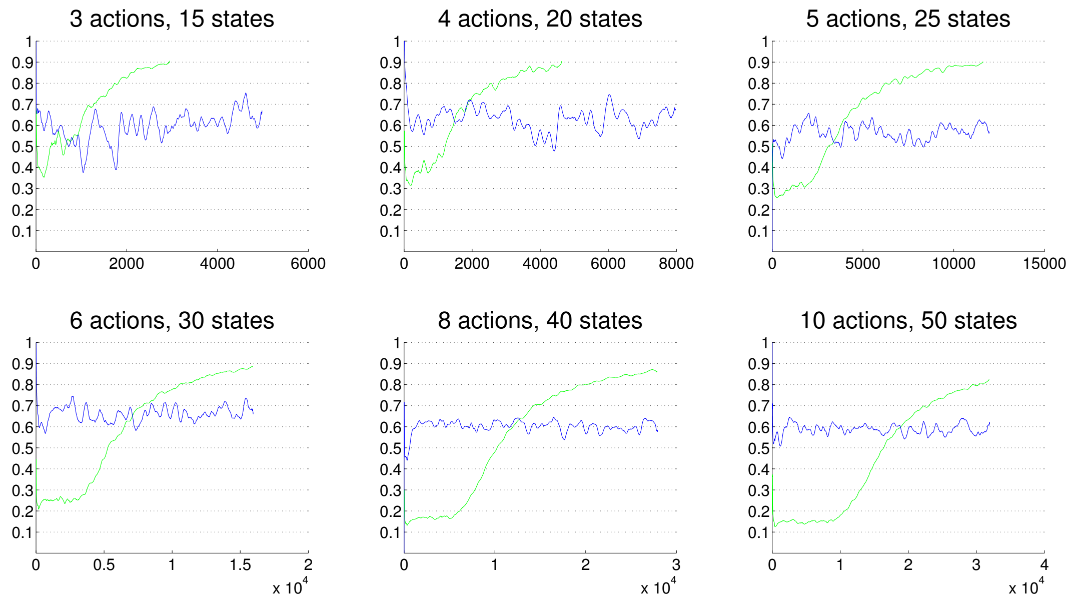


Figure 4.13: Trace of F1 score change for the traditional (blue) and the adaptive (green) simulators under the condition of increasing both the number of actions and the number of the mental states.

4.6 Comparison

In this chapter we have performed a series of experiments to compare the traditional approach and adaptive interactive learning. Table 4.4 summarizes the results we got.

	Traditional approach	**Adaptive approach**
Noiseless setting	Does not converge. Stagnation at F1 $\approx$ 0.6875	Converges. F1 $\geqslant$ 0.9 after $\approx$1900 samples.
Noisy setting	Does not converge. Stagnation at F1 $\approx$ 0.6423	Converges. F1 $\geqslant$ 0.9 after $\approx$3000 samples.
Increasing number of mental states	Does not converge. Maximal achievable F1 score increases.	Converges. Number of samples needed to converge increases.
Increasing number of actions	Does not converge. Maximal achievable F1 score descreases.	Converges. Number of samples needed to converge increases.
Increasing both mental state and action amounts	Does not converge. Maximal achievable F1 score seems to remain the same.	Converges. Number of samples needed to converge increases.

Table 4.4: Comparison of traditional and adaptive approaches on artificial data.

Chapter 5

Experiments on Real Data

The results, in which we are of the most interest to us, are the ones received from the experiments on the real data. In this chapter we describe the experiments we performed on the test subjects and the results we got.

5.1 Experimental Setup

The data is collected using the Emotiv EPOC device. Test subject is sitting in a chair behind the display, on which the stimuli are presented. Each electrode's placement is adjusted to report a perfect contact (green color in the Emotiv EPOC software) and respond with visible signal fluctuations to eye blinks and teeth clenching. These actions trigger muscle activity, which presents very specific markers on the raw EEG signal, for this reason they are traditionally used to make sure that the EEG device captures the signal.

The goal is to create a model to discriminate between 3 mental states: thinking "left", neutral state and thinking "right". Figure 5.1 presents the stimuli we use during the experiments.

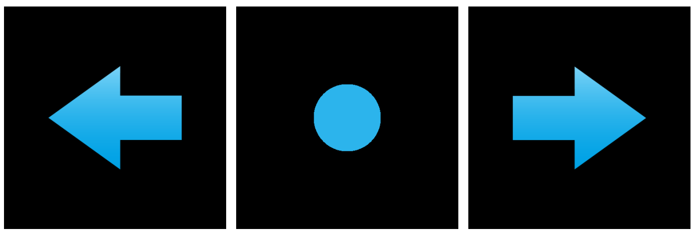


Figure 5.1: The stimuli presented to the test subject: "left", "neutral" and "right".

Each test subject was asked to complete three series of trials:

1. Training data for the traditional method: 7 minutes of stimuli presented in

random order, each stimulus shown for 10 seconds.

2. Test data for the traditional method: 3 minutes of stimuli presented in random order, each shown for 10 seconds.

3. Adaptive learning process: the duration is not fixed, the order of the stimuli is based on the probabilistic confusion matrix described in Section 3.2.3. System finds the minimal value on the main diagonal of the matrix and displays the stimulus, which corresponds to the row of the matrix where the minimal value was found. Each stimulus is shown for 10 seconds.

5.2 Traditional Method

We use Support Vector Machines (SVM) as the classification algorithm. The choice was made based on the empirical data: we tried three different algorithms to find out which one will work best. In addition to the SVM we tried Naïve Bayes and C4.5 classification algorithms.

For the SVM we have performed a search in the parameter space to estimate the best achievable result for the data. Here is the list of the parameters we were searched through:

- window size (seconds): {0.3, 0.5, 1, 2}
- kernel type: {linear, polynomial, RBF}
- polynomial degree for the polynomial kernel: {2, 3, 4, 5}
- regularization cost: {0.001, 0.01, 0.1, 1, 10, 100}

Figure 5.2 depicts the traditional simulator application.

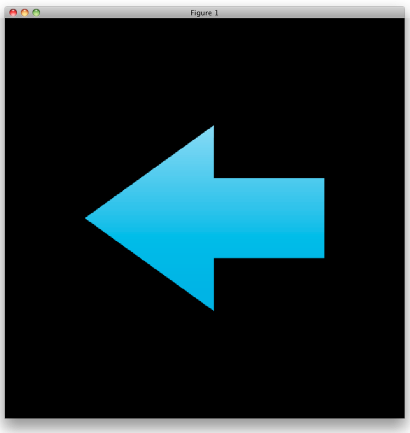


Figure 5.2: The GUI of the traditional experimenter has only one component: the stimulus display.

5.3 Novel Method

Figure 3.6 depicts the main window of the adaptive experimenter application. The experiment does not have a fixed time limit. It continues as long as the test subject is willing to continue or until the upper threshold for F1 score is reached. This would indicate that a sufficiently good model was found.

5.4 Results

In this section we present the final results we have achieved on the real data. We have conducted two types of experiments. During the first one the test subject was allowed to engage facial muscle activity in response to the stimuli. This type of response has clear effect on the EEG reading. The second experiment is aimed purely at the recognition of mental states. Test subject was asked to attempt not to move facial or any other muscles in association with the certain stimulus.

5.4.1 Facial expressions

Face muscle activity highly affects the EEG readings and can be observed with a naked eye on the raw signal. In this series of experiments we mapped facial expressions to the actions. The test subject's task was to train a 3-class classifier using facial expressions.

Figure 5.3 presents the results. The circular marker denote the result achieved using traditional approach and the triangular ones denote the adaptive approach. Each test subject is marked with different color. On the X-axis we can see the number of samples the algorithm needed to reach the F1 score displayed on the Y-axis.

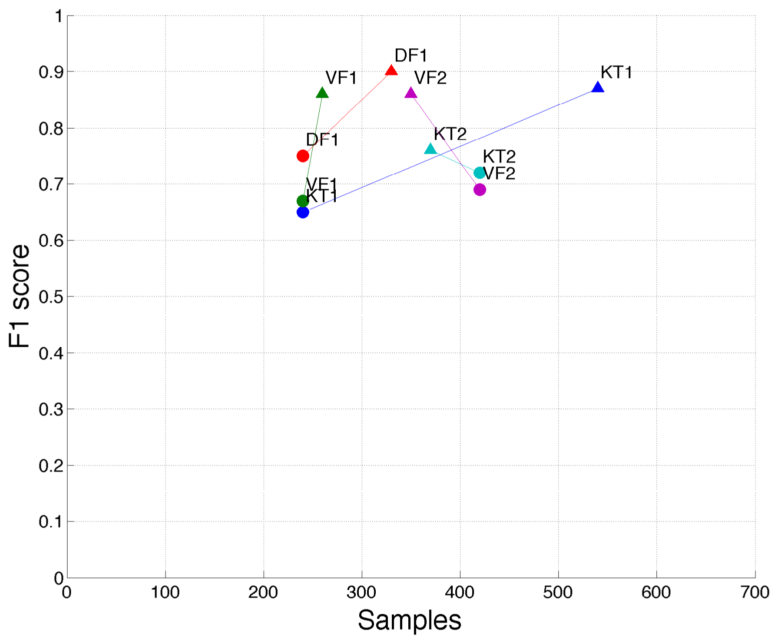


Figure 5.3: 3 class training results using facial expressions via traditional (circe) and interactive (triangle) approach.

We have conducted the experiment on too few test subjects to claim the statistically significant increase, but it seems that the adaptive model allows the test subject to find suitable actions and therefore leads to a higher performance score.

5.4.2 Mental states

Compared to the facial expressions experiment the task of distinguishing mental states if much harder. It also requires more time before the test subject begins to understand how his efforts affect the machine. In Section 4.5 we have seen that with increasing number of mental states the simulated test subject needs several thousands of samples to converge. The problem we encountered is that the task of sustaining a certain mental state is hard and the test subjects tire soon. To avoid this problem in the future we might be able to rearrange the learning process somehow.

Figure 5.4 shows the results we achieved on a small group of test subjects. Conducting a more broad series of experiments is part of the planned future work.

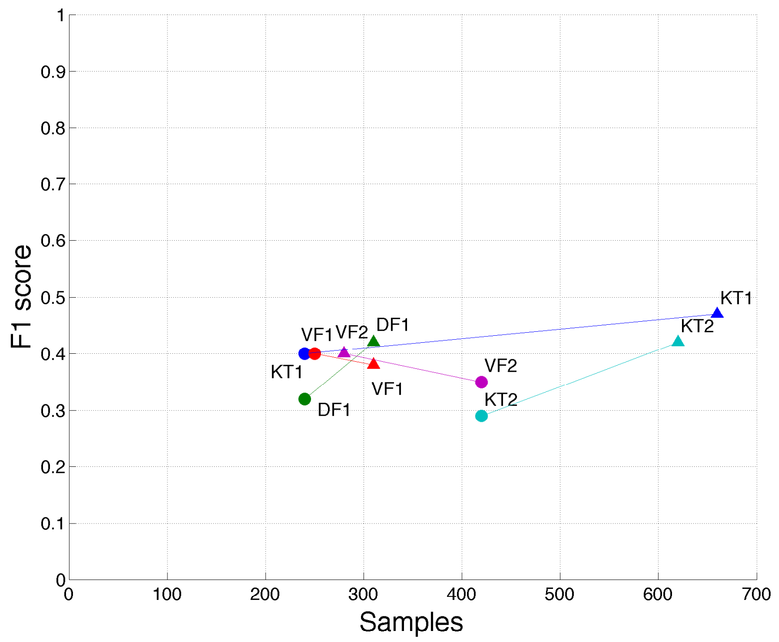


Figure 5.4: 3 class training results using power of thought via traditional (circle) and interactive (triangle) approach.

Chapter 6

Discussion

In this thesis we proposed a new conceptual and algorithmic approach to the process of training Brain-Computer Interface systems. It relies on the idea of interaction between the test subject and the machine and the ability of those two agents to adapt accordingly to the information received via feedback.

We believe that the novel approach we propose has several advantages over the traditional one:

- The test subject receives the feedback on how suitable his mental states and how good he is in sustaining them.
- Test subject can adapt his behavior during the learning process based on the feedback he receives.
- During the learning process the test subject can not significantly hinder the creation of a model by producing mislabeled samples.
- There is no need to pre-estimate the time needed for the learning process. Learning will finish as soon as the desired level of performance is achieved.
- The learning algorithm has only one parameter: the size of the SOM.

The experiments on the simulated data has shown the weaknesses of the traditional approach and the advantages of using the adaptive approach instead.

The results on real data show a small increase in the created model's performance, which is yet to be verified by a more broad series of experiments. However the results we achieved already are at least as good and even slightly better than the results of the traditional approach, which, together with the other advantages listed above speak in favor of the adaptive interactive training.

Future work

In the process of studying the topic we have encountered on numerous occasions that some subtopics and details require additional research and more experimental work. In this section we list the ideas which we plan to investigate and which could improve the learning algorithm and the training process further.

a) The current way of deciding when it is time to update the model is quite naïve. It relies only on the performance score of the model. A more reasonable alternative would consist of several models being created simultaneously. The system would track the performance of each of the models, which are created on different subsets of the data with different parameters, and if any of the alternative models becomes better than the current one then the system would replace the current model with the best one. Such an approach would also allow us to get rid of a parameter our algorithm has. The size of the SOM can be estimated during the learning process and changed if needed.

b) A smarter way to update the model is possible. The current solution recomputes the model from scratch using more data. This results in re-initializing the probability vectors and therefore loosing some information. A smarter way to change the model would rely on updating rather than recomputing. In terms of SOM this would mean realigning the weight vectors incrementally.

c) Perhaps some unsupervised online learning algorithms other than SOM should be attempted.

d) SOM itself can be used as a visualization of the feedback. Currently the feedback consists of the score and the collided stimulus. Another approach would be to allow the test subject to see the whole SOM and after each new processed sample highlight the unit of the map where this sample was put into and the probability score. In this way the test subject will gain the information about how hit mental efforts are aligned on the map.

e) Attempt different strategies for updating the probability vectors.

f) Rebuild the system from the Matlab code to a self-contained modular application, which can be extended by different learning algorithms, model update approaches and other pieces of internal logic.

g) Test the system on a more sophisticated EEG device.

h) Conduct a more extensive series of experiments to achieve statistically significant results.

Summary

A Brain-Computer Interface is a system which allows communication between a human and a computer. Using various neuroimaging techniques the brain activity is recorded and transmitted to the computer, where the signal is analyzed with the help of machine learning methods. The ultimate goal of BCI is to empower the human with the ability to control the external device with the power of thought.

However, distinguishing mental states of a human is a challenging task and standard machine learning alone is not enough to solve the problem. Acceptable level of performance can be achieved after a long training process, during which the human learns how to produce suitable mental states and the machine creates a model, which is able to classify the signal.

In this thesis we proposed a conceptually new approach to the process of training a BCI system. It relies on the idea of the interaction between the test subject and the machine and the ability of those two agents to adapt their behavior accordingly to the information they receive during the learning process. The approach is proposed as a counterpart to the traditional BCI training, where the test subject does not receive any feedback.

Another novelty in comparison to the traditional approach is using an unsupervised learning algorithm (SOM) as the core of the learning system. The original concept of self-organizing maps is amended to represent a probabilistic predictive model, which can be used to classify the brain signal, provide feedback and adapt the model in real time.

The devised learning scheme is implemented as a Matlab application, which communicates with an EEG device in real time. Series of experiments were conducted both on artificial and real data. The results of the experiments on artificial data show the number of advantages of the interactive approach over the traditional one. Experiments on real data confirm the superiority of the interactive approach as well.

On artificial data the simulator of the adaptive learning method always converges to the upper F1 score threshold of 0.9. The simulator based on the traditional method rarely achieved the 0.9 threshold, in most cases it gets stuck at a certain level of F1 score. The F1 score level at which stagnation begins depends on the

complexity of the simulation and varies from 0.5 to 0.8. The adaptive simulator responds to more complex problems by increasing the time it requires in order to reach the 0.9 threshold.

We have conducted two types of experiments on real data:

1. Classification of facial expressions. During this series of experiments the test subject was allowed to respond to stimuli with movement of facial muscles, which are well detected by the EEG device.

2. Classification of mental states. The test subject is not allowed to engage any muscle activity in association with certain mental states and has to communicate with the system using mental activity only.

Classification of facial expressions using the traditional approach yielded F1 score = 0.69 on average. Using the interactive approach the test subject was able to bring the F1 score up to 0.9. Experiments on mental state classification were not as successful. The best result is the increase of F1 score from 0.4 (traditional approach) to 0.47 (interactive approach) for one of the test subjects.

We conclude that adaptive learning process has the multiple major advantages over the traditional one and deserves a future study.

Interaktiivne kohandamisvõimeline õppimine: uus lähenemine aju-arvuti liidese süsteemi õpetamiseks

Magistritöö (30 EAP)

Ilja Kuzovkin

Kokkuvõte

Aju-arvuti liides (AAL) on süsteem, mis võimaldab infovahetust inimese aju ja arvuti vahel. Kasutades erinevaid neuropildistuste tehnikaid aju aktiivsust salvestatakse ja saadetakse arvutisse, kus signaal töödeldakse masinõpe meetoditega. AALi põhieesmärk on anda inimesele võimalust juhtida välisseadet kasutades mõttejõudu.

Inimese mõtteseisundite eristame on raske ülesanne, mis ei ole lahendatav ainult masinõpe kasutamisega. Vastuvõetav klassifitseerimise täpsuse tase on saavutatav pärast pikajalist õpetamise protsessi, mille jooksul inimene õpib kuidas ta peab tekitama sobivad mõtteseisundid, ning arvuti loob mudeli, mis oskab neid eristada.

Käesolevas töös me esitame uut lähenemist AAL süsteemi õpetamise protsessi jaoks. See põhineb inimese ja arvuti koostoimimise ideel, mille jooksul mõlemad osapooled adapteerivad oma käitumist vastavalt sellele, millist tagasisided nad saavad suhtlemise ajal. Pakutud viisi vastandiks on võetud traditsiooniline lähenemine, kus katseisik ei saa tagasisidet õppeprotsessi edukusest selle käigus.

Teine uudsus traditsioonilise meetodiga võrreldes on juhendamata õppealgoritmi kasutamine (iseorganiseeriv kaart, SOM) meie süsteemi tuumana. Algne iseorganiseeruva kaardi algoritm on täiendatud niimoodi, et ta esindab tõenäosusliku ennustamise mudelit, mis oskab klassifitseerida aju signaali, anda tagasisidet kat-

seisikule ning vajadusel kohandada mudelit reaalajas.

Väljpakutud õppimise skeem on realiseeritud Matlab rakendusena, mis suhtleb EEG seadmega reaalajas. Kasutades seda rakendust said läbiviidud eksperimendid kunst-, ning reaalaandmete põhjal. Eksperimentide tulemused näitavad, et interaktiivsel meetodil on eelised traditsioonilise meetodiga võrreldes. Eksperimendid reaalandmetel ka kinnitavad interaktiivse meetodi paremust.

Kunstlike andmete põhjal interaktiivse meetodi simulaatori tulemus alati jõuab eelmääratud F1 skoori 0.9 piirile. Simulaator, mis baseerub traditsioonilise meetodi põhjal, jõuab aga 0.9 piirini väga harva. Tavaliselt mudel jääb kinni ebaoptimaalse täpsuse tasemel. Sõltuvalt ülesande keerukusest algab kinnijäämine F1 skoori tasemetel 0.5 kuni 0.8. Adaptiivse käitumise simulaator oli võimeline ka raskemate ülesannete puhul saavutada skoori 0.9, kugi vajas selleks rohkem aega kui lihtsamate ülesannete puhul.

Me viisime läbi kaks tüüpi eksperimentidest:

1. Näoilmete klassifitseerimine. Eksperimendi jooksul pidi katseisik vastama stiimulitele kasutadades näo lihaseid. Need liigutused on kergesti tuvastatavad EEG seadme poolt.
2. Mõtteseisundi klassifitseerimine. Katseisik pidi vastama stiimulitele ainult kasutades mõtteseisundeid, talle ei olnud lubatud seostada lihaste liigutused ekraanil olevate stiimulitega.

Näoilmete klassifitseerimine kasutades traditsioonilist lähenemist andis keskmiselt F1 skoori 0.69. Kasutades aga interaktiivset meetodit suutsid katseisikud tõsta süsteemi täpsust 0.9-ni. Eksperimendid mõtteseisundite klassifitseerimisest ei olnud nii edukad, kuid ka siin F1 skoori parenemine oli saavutatud: parima tulemuse saavutas katseisik, kes suutis interaktiivse meetodi abil tõsta F1 skoori 0.4-st (saavutatud traditsioonilise meetodiga) 0.47-ni.

Me järeldame, et interaktiivne lähenemine süsteemi õpetamiseks omab järgmisi eelisi traditsioonilise meetodiga võrreldes:

- Katseisik saab tagasisidet õppeprotsessi ajal, mis võimaldab tal kohandada oma käitumist parema tulemuse saavutamiseks.
- Katseisik ei saa tekitada “valet” signaali näidist, mis oluliselt takistaks adekvaatse mudeli loomist.
- Meie algoritm omab ainult üht parameetri, samas kui traditsiooniline meetod, sõltudes algoritmi valikust, võib nõuda ulatusliku otsingu algoritmi parameetrite ruumis, selleks ei jõuda adekvaatse mudelini.

Appendix A: Software download links and instructions

Components of the system

The most current implementation of the system can be downloaded from the URL `http://www.ikuz.eu/adaptive/code-masters.zip`. The data we have collected can be downloaded from URL `http://www.ikuz.eu/adaptive/data-masters.zip`.

Adaptive experimenter

In order to run this application open `Adaptive Experimenter` directory in your Matlab environment and run `experimenter_feedback_gui.m`.

Traditional experimenter

To run this application open `Traditional Experimenter` directory in your Matlab environment and run `experimenter.m`.

Generator

In order to run this application open `Generator` directory in your Matlab environment and run `generator.m`. If you want to run Generator and Experimenter simultaneously run them as separate Matlab instances.

Adaptive simulator

In order to run this application open `Adaptive Simulator` directory in your Matlab environment and run `adaptive_offline.m`.

Traditional simulator

In order to run this application open `Traditional Simulator` directory in your Matlab environment and run `traditional_offline.m`.

Toolboxes

FieldTrip

The toolbox can be downloaded from `http://fieldtrip.fcdonders.nl/download`. After the download add it to the Matlab path and run command `ft_defaults` to initialize the toolbox components. To read the data stream from the Emotiv EPOC device run the `emotiv2ft.exe` application located in the `FieldTrip\realtime\bin\win32` directory.

SVM toolbox

The toolbox is required to run `Traditional Experimenter` and `Traditional Simulator` applications and can be downloaded from `http://www.cis.hut.fi/projects/somtoolbox/download`. Add the unpacked toolbox directory to the Matlab path.

EEG-like data generator

The Matlab code to generate EEG-like data can be downloaded from `http://www.cs.bris.ac.uk/home/rafal/phasereset`. It is, however, also included in the distribution of our system.

Bibliography

[aud] Physiological and histological investigation of structure and function of the normal auditory system
`http://www.ihr.mrc.ac.uk/index.php/research/current/1`.

[BKM+91] JW Belliveau, DN Kennedy, RC McKinstry, BR Buchbinder, RM Weisskoff, MS Cohen, JM Vevea, TJ Brady, and BR Rosen. Functional mapping of the human visual cortex by magnetic resonance imaging. *Science*, 254(5032):716–719, 1991.

[CL11] Chih-Chung Chang and Chih-Jen Lin. Libsvm: a library for support vector machines. *ACM Transactions on Intelligent Systems and Technology (TIST)*, 2(3):27, 2011.

[CS03] Eleanor A Curran and Maria J Stokes. Learning to control brain activity: A review of the production and control of eeg components for driving brain–computer interface (bci) systems. *Brain and cognition*, 51(3):326–336, 2003.

[CV95] Corinna Cortes and Vladimir Vapnik. Support-vector networks. *Machine learning*, 20(3):273–297, 1995.

[dB05] Roberto Coelho de Berrêdo. *A review of spiking neuron models and applications*. PhD thesis, Universidade Federal de Minas Gerais, 2005.

[EE04] Abbas Erfanian and Ali Erfani. Ica-based classification scheme for eeg-based brain-computer interface: the role of mental practice and concentration skills. In *Engineering in Medicine and Biology Society, 2004. IEMBS'04. 26th Annual International Conference of the IEEE*, volume 1, pages 235–238. IEEE, 2004.

[emo] Emotiv epoc specifications
`http://emotiv.com/upload/manual/EPOCSpecifications.pdf`.

[Emo12] EPOC Emotiv. Software development kit, 2012.

[FGG97] Nir Friedman, Dan Geiger, and Moises Goldszmidt. Bayesian network classifiers. *Machine learning*, 29(2-3):131–163, 1997.

[Fis91] Bruce J. Fisch. *Spehlmann's EEG Primer.* ELSEVIER, 1991.

[GEV03] Gary N Garcia, Touradj Ebrahimi, and J-M Vesin. Support vector eeg classification in the fourier and time-frequency correlation domains. In *Neural Engineering, 2003. Conference Proceedings. First International IEEE EMBS Conference on*, pages 591–594. IEEE, 2003.

[GFF$^+$93] PM Grasby, CD Frith, KJ Friston, CRSF Bench, RSJ Frackowiak, and RJ Dolan. Functional mapping of brain areas implicated in auditory—verbal memory function. *Brain*, 116(1):1–20, 1993.

[GPD85] George L Gerstein, Donald H Perkel, and Judith E Dayhoff. Cooperative firing activity in simultaneously recorded populations of neurons: detection and measurement. *The Journal of neuroscience*, 5(4):881–889, 1985.

[HFL10] Christoph S Herrmann, Ingo Fründ, and Daniel Lenz. Human gamma-band activity: a review on cognitive and behavioral correlates and network models. *Neuroscience & Biobehavioral Reviews*, 34(7):981–992, 2010.

[HIT$^+$03] Takashi Hanakawa, Ilka Immisch, Keiichiro Toma, Michael A Dimyan, Peter Van Gelderen, and Mark Hallett. Functional properties of brain areas associated with motor execution and imagery. *Journal of Neurophysiology*, 89(2):989–1002, 2003.

[HKI09] H.J. Hwang, K. Kwon, and C.H. Im. Neurofeedback-based motor imagery training for brain? computer interface (bci). *Journal of neuroscience methods*, 179(1):150–156, 2009.

[HP00] Ernst Haselsteiner and Gert Pfurtscheller. Using time-dependent neural networks for eeg classification. *Rehabilitation Engineering, IEEE Transactions on*, 8(4):457–463, 2000.

[Izh03] Eugene M Izhikevich. Simple model of spiking neurons. *Neural Networks, IEEE Transactions on*, 14(6):1569–1572, 2003.

[KABG$^+$06] Elif Kirmizi-Alsan, Zubeyir Bayraktaroglu, Hakan Gurvit, Yasemin H Keskin, Murat Emre, Tamer Demiralp, et al. Comparative analysis of event-related potentials during go/nogo and cpt: decomposition of electrophysiological markers of response inhibition and sustained attention. *Brain research*, 1104(1):114–128, 2006.

[KC06] Michael A Kisley and Zoe M Cornwell. Gamma and beta neural activity evoked during a sensory gating paradigm: effects of auditory, somatosensory and cross-modal stimulation. *Clinical neurophysiology*, 117(11):2549–2563, 2006.

[Koh82] Teuvo Kohonen. Self-organized formation of topologically correct feature maps. *Biological cybernetics*, 43(1):59–69, 1982.

[KSO07] Noriaki Kanayama, Atsushi Sato, and Hideki Ohira. Crossmodal effect with rubber hand illusion and gamma-band activity. *Psychophysiology*, 44(3):392–402, 2007.

[LC02] Hyekyoung Lee and Seungjin Choi. Pca-based linear dynamical systems for multichannel eeg classification. In *Neural Information Processing, 2002. ICONIP'02. Proceedings of the 9th International Conference on*, volume 2, pages 745–749. IEEE, 2002.

[LCL$^+$07] Fabien Lotte, Marco Congedo, Anatole Lécuyer, Fabrice Lamarche, Bruno Arnaldi, et al. A review of classification algorithms for eeg-based brain–computer interfaces. *Journal of neural engineering*, 4, 2007.

[LdS91] Fernando Lopes da Silva. Neural mechanisms underlying brain waves: from neural membranes to networks. *Electroencephalography and clinical neurophysiology*, 79(2):81–93, 1991.

[Luc05] Steven J Luck. An introduction to the event-related potential technique (cognitive neuroscience). 2005.

[M$^+$67] James MacQueen et al. Some methods for classification and analysis of multivariate observations. In *Proceedings of the fifth Berkeley symposium on mathematical statistics and probability*, volume 1, page 14. California, USA, 1967.

[MGPF99] Johannes Müller-Gerking, Gert Pfurtscheller, and Henrik Flyvbjerg. Designing optimal spatial filters for single-trial eeg classification in a movement task. *Clinical neurophysiology*, 110(5):787–798, 1999.

[ner] Nerve cell `http://en.wikipedia.org/wiki/Nerve_cell`.

[NYC04] Sander Nieuwenhuis, Nick Yeung, and Jonathan D Cohen. Stimulus modality, perceptual overlap, and the go/no-go n2. *Psychophysiology*, 41(1):157–160, 2004.

[OFMS11] Robert Oostenveld, Pascal Fries, Eric Maris, and Jan-Mathijs Schoffelen. Fieldtrip: open source software for advanced analysis of meg, eeg, and invasive electrophysiological data. *Computational intelligence and neuroscience*, 2011:1, 2011.

[Pea01] Karl Pearson. Liii. on lines and planes of closest fit to systems of points in space. *The London, Edinburgh, and Dublin Philosophical Magazine and Journal of Science*, 2(11):559–572, 1901.

[PKN$^+$96] G Pfurtscheller, J Kalcher, Ch Neuper, D Flotzinger, and M Pregenzer. On-line eeg classification during externally-paced hand movements using a neural network-based classifier. *Electroencephalography and clinical Neurophysiology*, 99(5):416–425, 1996.

[PNG$^+$00] Gert Pfurtscheller, C Neuper, C Guger, WAHW Harkam, Herbert Ramoser, Alois Schlogl, BAOB Obermaier, and MAPM Pregenzer. Current trends in graz brain-computer interface (bci) research. *Rehabilitation Engineering, IEEE Transactions on*, 8(2):216–219, 2000.

[PP07] Satu Palva and J Matias Palva. New vistas for α-frequency band oscillations. *Trends in neurosciences*, 30(4):150–158, 2007.

[Qui93] John Ross Quinlan. *C4. 5: programs for machine learning*, volume 1. Morgan kaufmann, 1993.

[SE05] Abdulhamit Subasi and Ergun Erçelebi. Classification of eeg signals using neural network and logistic regression. *Computer Methods and Programs in Biomedicine*, 78(2):87–99, 2005.

[som] Kohonen's self-organizing map
`http://newplans.net/Robotics/ArtificialNetwork/kohonen's.htm`.

[ŠSS03] Jakub Št'astnỳ, Pavel Sovka, and Andrej Stančák. Eeg signal classification: introduction to the problem. *Radioengineering*, 12(3):51–55, 2003.

[TAM07] Ryota Tomioka, Kazuyuki Aihara, and Klaus-Robert Müller. Logistic regression for single trial eeg classification. *Advances in neural information processing systems*, 19:1377–1384, 2007.

[TBS$^+$93] Vernon L Towle, José Bolaños, Diane Suarez, Kim Tan, Robert Grzeszczuk, David N Levin, Raif Cakmur, Samuel A Frank, and Jean-Paul Spire. The spatial location of eeg electrodes: locating the best-fitting sphere relative to cortical anatomy. *Electroencephalography and clinical neurophysiology*, 86(1):1–6, 1993.

[TGP04] George Townsend, Bernhard Graimann, and Gert Pfurtscheller. Continuous eeg classification during motor imagery-simulation of an asynchronous bci. *Neural Systems and Rehabilitation Engineering, IEEE Transactions on*, 12(2):258–265, 2004.

[tre] Decision trees
`https://www.projectrhea.org/oldkiwi/index.php/Lecture_21_-_Decision_Trees(Continued)`.

[VHAP00] Juha Vesanto, Johan Himberg, Esa Alhoniemi, and Juha Parhankangas. *SOM toolbox for Matlab 5*. Citeseer, 2000.

[VRPG90] AC Metting Van Rijn, A Peper, and CA Grimbergen. High-quality recording of bioelectric events. *Medical and Biological Engineering and Computing*, 28(5):389–397, 1990.

[WF05] Ian H Witten and Eibe Frank. *Data Mining: Practical machine learning tools and techniques*. Morgan Kaufmann, 2005.

[Won10] Cheuk-Wah Wong. The brain, thinking, and memory: A theory based on pathophysiology of amnesia. *Neurology*, 100, March 2010.

[YBHC04] Nick Yeung, Rafal Bogacz, Clay B Holroyd, and Jonathan D Cohen. Detection of synchronized oscillations in the electroencephalogram: an evaluation of methods. *Psychophysiology*, 41(6):822–832, 2004.

Internet URLs were valid on May 20, 2013.

License