IDIAP RESEARCH REPORT

Online Classifier Adaptation in Brain-Computer Interfaces

Anna Buttfield [a] José del R. Millán [a b]

IDIAP–RR 06-16

March 2006

[a] IDIAP Research Institute, Martigny, Switzerland
[b] Ecole Polytechnique Fédérale de Lausanne (EPFL), Switzerland

IDIAP Research Institute **www.idiap.ch**
Rue du Simplon 4 P.O. Box 592 1920 Martigny – Switzerland
Tel: +41 27 721 77 11 Fax: +41 27 721 77 12 Email: info@idiap.ch

IDIAP Research Report 06-16

Online Classifier Adaptation in Brain-Computer Interfaces

Anna Buttfield   José del R. Millán

March 2006

Abstract. Brain-computer interfaces (BCIs) aim to provide a new channel of communication by enabling the subject to control an external systems by using purely mental commands. One method of doing this without invasive surgical procedures is by measuring the electrical activity of the brain on the scalp through electroencephalography (EEG). A major obstacle to developing complex EEG-based BCI systems that provide a number of intuitive mental commands is the high variability of EEG signals. EEG signals from the same subject vary considerably within a single session and between sessions on the same or different days. To deal with this we are investigating methods of adapting the classifier while it is being used by the subject. By keeping the classifier constantly tuned to the EEG signals of the current session we hope to improve the performance of the classifier and allow the subject to learn to use the BCI more effectively. This paper discusses preliminary offline and online experiments towards this goal, focusing on the initial training period when the task that the subject is trying to achieve is known and thus supervised adaptation methods can be used. In these experiments the subjects were asked to perform three mental commands (imagination of left and right hand movements, and a language task) and the EEG signals were classified with a Gaussian classifier.

1 Introduction

The goal of a Brain-computer interface (BCI) is to provide a channel of communication that does not rely on the subject's peripheral muscles and nerves. This would obviously be very useful for paralysed people or anyone else who cannot use normal methods to communicate with other people and interact with his or her environment.

The concept of a brain-computer interface (BCI) was first developed by Vidal in the 1970s [13]. However, advances in BCI research were limited until developments in our neurological understanding of the structure and behaviour of the brain combined with modern advances in signal acquisition and processing techniques to make the current approach to BCIs possible. There are a number of different approaches to develop a BCI, which are often difficult to compare as they are trying to achieve different tasks in different ways. Invasive approaches involve surgically implanting electrodes into the cortex of animals or humans, and directly measuring the activity of the neurons. This approach has achieved some good results, but it does carry ethical and practical concerns. Alternatively, non-invasive BCIs measure brain activity by techniques such as electroencephalogram (EEG) recordings, which use electrodes on the scalp to detect the electrical activity of the brain. Despite the drawbacks of EEG, which include noisy signals and poor spatial resolution, this approach has shown promising results. For a thorough overview of the BCI field see reviews by Millán [4] and Wolpaw et al. [14].

In an EEG-based BCI, the EEG signals are recorded by electrodes on the scalp, with the number of electrodes varying from one or two electrodes positioned by hand to 128 or more electrodes built into an electrode cap. Features are then extracted from these EEG signals and sent to the classifier, which translates these features into the control signals. These control signals can then be used to drive different applications, such as controlling a robot in a model indoor environment [7]. Communication applications such as operating a virtual keyboard have also been developed [1, 6, 9, 15].

One major challenge of creating a BCI is the variability in the EEG signals. Their distribution change both between BCI sessions and within individual sessions, due to a number of factors including changes in background brain activity, fatigue and concentration levels, and intentional change of mental strategy by the subject. This means that a classifier trained on past data from subject will probably not be optimal for following sessions. Even for a subject who has developed a high degree of control of the EEG there are variations in the EEG signals over a session. In a subject who is first starting to learn to use the BCI these variations are going to be more pronounced, as the subject has not yet learned to generate stable EEG signals.

Because of this inherent non-stationarity of EEG signals, we are investigating ways in which we can improve performance of the classifier by adapting the classifier as it is being used. This means that we train a classifier offline on previous data, then as we receive EEG signals in the new session we adapt the classifier with this new data. Through this online classifier adaptation we aim to keep the classifier constantly tuned to the EEG signals it is receiving in the current session. In performing online adaptation we are limited in both time and computing resources. The BCI system is classifying the incoming signals in real time, and we do not want to reduce the rate at which we can sample data and make decisions by using an adaptation strategy that takes too much time. For computing resources, since the classifier adaptation is only a part of the online operation of the BCI, which will probably also include at least signal acquisition and a graphical user interface, we would like to keep the adaptation algorithm as lightweight as possible. So in most cases with online learning we will use each data sample only once and in chronological order, since we adapt the classifier based on each new sample as it is presented, then discard the sample. This is in contrast to stochastic gradient descent, which also takes samples individually, but is not limited to taking samples in order and can reuse samples as many times as necessary for convergence. A range of different techniques have been developed to address the problem of online learning [8].

The need for adaptation in BCIs has been recognised for some time [5, 14], however little research has been published concerning this area. Online adaptation has been proposed to address two issues: compensation for the inherent non-stationarity of EEG signals, and allowing mutual adaptation between the subject and the BCI. In this way there are two different situations where we would like to use online learning to adapt the classifier — in initial training, to facilitate mutual adaptation between the subject and the BCI, and during ongoing use, where we would like to be able to make relatively small adjustments to the classifier to correct for drifts in the signal. The amount of adaptation necessary, and the amount of available information on classifier performance, will determine what approach to take to classifier adaptation.

1.1 Supervised adaptation for subject training

To help the subject learn to use the BCI we would like to be able to give them fast and accurate feedback on how well the different tasks can be differentiated from each other. If we are limited to training the classifier offline, we are effectively giving them feedback on how well his or her EEG signals match the pattern of the previously collected data. This approach has the advantage of encouraging the subject to learn to produce stable EEG signals that match those already collected. However, it makes it harder for the subject to refine his or her mental task generation strategies — they need keep the same strategy for the entirety of a session and remember exactly what they were doing to test it in the next session. In the best case this is slow training, since the true feedback is always lagged by a session — in effect we're giving them feedback on how similar his or her signals are to the previous sessions, not how easily we can differentiate the signals. In the worst case this training is misleading — EEG signals always change between sessions, but we are giving them feedback on how well his or her signals match the signals from the previous session, when reproducing these signals may be impossible. This is particularly an issue in early training where there is less data on which to build a model, and that data is likely to be less stable.

During initial training we know what class the subject is trying to generate at all times, so we can use supervised methods to adapt the classifier at this stage. The same techniques could be applied during ongoing use as a periodic recalibration step. In either case, the goal is to adapt the classifier to compensate for the changes in the signal between sessions, and then track the signals as they vary throughout the session. This variation through the session will probably be more pronounced in the early sessions, as in the beginning the subject will be trying different mental strategies and generally becoming accustomed to using the BCI.

1.2 Adaptation in ongoing use

Once the subject has reached a reasonably stable level of performance in the initial training they will start to use the BCI in a real situation. Even with a well trained subject, however, there is still drift in the signal between sessions and during usage — particularly in long sessions— that will degrade the performance of the BCI. In this situation we don't know the exact intention of the subject, so without complete information about the performance of the BCI we would need to develop other methods for online classifier adaptation that are not supervised.

Reinforcement learning [12] is a framework that could be useful for this situation. The problem that reinforcement learning attempts to address is that of learning when we only receive occasional feedback on how well or badly we are performing, rather than explicitly being told what the correct response should have been for each sample. That is, we could use whatever partial information we can glean about BCI performance during ongoing use to improve the classifier. In particular, we can gain some partial information by examining the EEG signals or examining how the BCI task is being performed.

One approach that might be able to give us some information about the performance of the classifier is the recognition of cognitive error potentials [2]. Error potentials are the reaction in the subject's brain to a mistake made by the interface. Thus, if an error potential is detected when the classifier makes a wrong classification, we know which class the subject was not attempting to produce, even though we don't know what was the actual target. If we can reliably recognise these error potentials we know when the classifier has made a mistake in the recognition of the subject's intent, and we can update the classifier based on this information. In this case, we have rapid feedback on whether a classification was erroneous, and we can use this negative feedback to update the classifier. An alternative source of information is contextual information about how well the interface is being used, for example, evaluating the quality of the robot's path in a robot navigation problem, or noting when the subject deletes letters in a keyboard application. In both cases we have only have occasional feedback on how well or badly the classifier is performing, which is the situation addressed by reinforcement learning.

2 Online Classifier Adaptation

2.1 Gaussian classifier

We use a Gaussian classifier to separate the signal into the different classes of mental task. Each class is represented by a number of Gaussian prototypes, typically less than four. That is, we assume that the class-conditional probability function of class C_i is a superposition of N_j Gaussian prototypes. We also assume that all classes have equal prior probability. All classes have the same number of prototypes N_p, and for each class each prototype has equal weight $1/N_p$.

Dropping the constant terms, we can define the posterior probability y_c of the class c in terms of the total activation of the classifier (A) and the activation of class c (a_c):

$$A = \sum_{i=1}^{N_c} \sum_{j=1}^{N_p} a_{ij} \tag{1}$$

$$a_c = \sum_{j=1}^{N_p} a_{cj} \tag{2}$$

$$y_c = \frac{a_c}{A} \tag{3}$$

where N_c is the number of classes and a_{ij} is the activation level of the j^{th} prototype of class C_i, with centre μ_{ij} and diagonal covariance matrix $\mathbf{\Sigma}_i$, for a given sample $\mathbf{x}$

$$a_{ij} = \frac{1}{\prod_k \Sigma_{ik}} \exp\left(-\frac{1}{2} \sum_k \frac{(\mathbf{x}_k - \mu_{ijk})^2}{\Sigma_{ik}} \right) \tag{4}$$

In this equation, μ_{ijk} is the k^{th} element of the vector μ_{ij}, and Σ_{ik} is the element (k,k) of the diagonal matrix Σ_i. Usually each prototype of each class would have an individual covariance matrix Σ_{ij}, but to reduce the number of parameters the model uses a single diagonal covariance matrix common to all the prototypes of the same class.

The decision of the classifier for input vector $\mathbf{x}$ is now the class with the highest probability, provided that the probability is above a given threshold, otherwise the result is “unknown”.

2.2 Adaptation with stochastic gradient descent

Training of the classifier starts from an initial model that can be either a previously trained classifier or a new classifier created by estimating the prototype centres with a clustering algorithm. This initial estimate is then improved by stochastic gradient descent to minimise the mean square error, given by:

$$E = \frac{1}{2} \sum_{i=1}^{N_c} (y_i - t_i)^2 \tag{5}$$

where $\mathbf{t}$ is the target vector in the form 1-of-C; that is, if the second of three classes was the desired output, the target vector is (0,1,0).

This optimisation is performed on the mean and covariance of each prototype. We calculate the derivative of the error with respect to element l of the mean and the covariances respectively, for prototype p of class c:

$$\frac{\partial E(x)}{\partial \mu_{cpl}} = \frac{a_{cp}}{A} \frac{[x_l - \mu_{cpl}]}{\Sigma_{cl}} \left[(y_c - t_c) - \sum_{i=1}^{N_c} (y_i(y_i - t_i)) \right] \tag{6}$$

$$\frac{\partial E(x)}{\partial \Sigma_{cpl}} = \frac{1}{2} \frac{a_{cp}}{A} \frac{[x_l - \mu_{cpl}]^2 - \Sigma_{cpl}}{(\Sigma_{cl})^2} \left[(y_c - t_c) - \sum_{i=1}^{N_c} (y_i(y_i - t_i)) \right] \tag{7}$$

The gradient descent update equations are now defined as follows, with learning rates for the centres and covariances α and β respectively:

$$(\mu_{cpl})_{t+1} = (\mu_{cpl})_t - \alpha \cdot \frac{\partial E(\mathbf{x}_t)}{\partial \mu_{cpl}} \tag{8}$$

$$(\Sigma_{cpl})_{t+1} = (\Sigma_{cpl})_t - \beta \cdot \frac{\partial E(\mathbf{x}_t)}{\partial \Sigma_{cpl}} \tag{9}$$

At each step the updates to the covariance matrices are computed individually then averaged over the prototypes of each class to give Σ_c.

When updating the covariance matrices it is important to ensure that they never become negative. One way to do this is simply to impose a small positive lower limit on $(\Sigma_{cpl})_{t+1}$. An alternative method is to use exponentiated gradient descent to update the covariances [3], which ensures that the covariances are always positive:

$$(\Sigma_{cpl})_{t+1} = (\Sigma_{cpl})_t \cdot \exp\left(-\beta \cdot \frac{\partial E(\mathbf{x}_t)}{\partial \Sigma_{cpl}}\right) \tag{10}$$

2.3 Stochastic Meta Descent

Stochastic Meta Descent (SMD) [10] is an extension of gradient descent that uses adaptive learning rates to accelerate learning. The SMD algorithm is a non-linear extension of earlier work [11].

The SMD algorithm is applied to each parameter in the classifier separately (the centre and covariance of each Gaussian prototype), and each parameter maintains and adapts an individual learning rate. This is in contrast to basic gradient descent, which uses a single learning rate for all parameters. Thus the parameters $\boldsymbol{\mu}_{ij}$ and $\boldsymbol{\Sigma}_{ij}$ of prototype j of class i have learning rates $\mathbf{p}_{ij}$ and $\mathbf{q}_{ij}$ respectively, gradient traces $\mathbf{v}_{ij}$ and $\mathbf{w}_{ij}$ respectively, and gradients $(\boldsymbol{\delta\mu}_{ij})_t$ and $(\boldsymbol{\delta\Sigma}_{ij})_t$ respectively. For simplicity the indices i and j have been dropped from the following equations.

The equation for adapting the Gaussian prototype centre $\boldsymbol{\mu}$ with respect to the error function E and input $\mathbf{x}_t$ is:

$$\boldsymbol{\mu}_{t+1} = \boldsymbol{\mu}_t + \mathbf{p}_t \cdot (\boldsymbol{\delta}_{\boldsymbol{\mu}})_t, \text{ where } (\boldsymbol{\delta}_{\boldsymbol{\mu}})_t \equiv -\frac{\partial E(\mathbf{x}_t)}{\partial \boldsymbol{\mu}} \tag{11}$$

This equation is an extension of the gradient descent update rule, since if we replaced the vector of learning rates $\mathbf{p}_t$ with a scalar we have the basic gradient descent update rule. We update the learning rates by exponentiated gradient descent, which allows the learning rates to cover a large range of positive values:

$$\mathbf{p}_t = \mathbf{p}_{t-1} \cdot \exp(\alpha\, (\boldsymbol{\delta}_{\boldsymbol{\mu}})_t\, \mathbf{v}_t) \tag{12}$$

In this equation the term α is the meta-learning rate for the centres. The term $\mathbf{v}_t$ is the gradient trace, which projects forward into time the effect of a change in learning parameter on the variables, and is defined as $\mathbf{v}_t \equiv -\frac{\partial \boldsymbol{\mu}_t}{\partial \ln(\mathbf{p})}$. From this we can derive an iterative update rule:

$$\mathbf{v}_{t+1} = \mathbf{v}_t + \mathbf{p}_t \left((\boldsymbol{\delta}_{\boldsymbol{\mu}})_t - (\mathbf{H}_{\mu})_t \mathbf{v}_t \right) \tag{13}$$

where $(\mathbf{H}_\mu)_t$ is the Hessian matrix of E with respect to $\boldsymbol{\mu}$.

A similar system of equations is derived for the covariance updates. Using linear gradient descent would give us a parallel system of equations to those for the centres. If we choose to use exponentiated gradient descent, we need to derive a new set of equations:

$$\mathbf{\Sigma}_{t+1} = \mathbf{\Sigma}_t \cdot \exp(\mathbf{q}_t(\boldsymbol{\delta}_{\mathbf{\Sigma}})_t), \text{ where } (\boldsymbol{\delta}_{\mathbf{\Sigma}})_t \equiv -\frac{\partial E\left(\mathbf{x_t}\right)}{\partial \mathbf{\Sigma}} \tag{14}$$

The learning rate update for the covariance is then

$$\mathbf{q}_t = \mathbf{q}_{t-1} \cdot \exp(\beta \left(\boldsymbol{\delta}_{\mathbf{\Sigma}}\right)_t \mathbf{w}_t) \tag{15}$$

with β being the meta-learning rate for the covariances. The gradient trace $\mathbf{w}$ for the covariance is derived as:

$$\mathbf{w_{t+1}} = \exp(\mathbf{q}_t(\boldsymbol{\delta}_{\mathbf{\Sigma}})_t) \cdot \left[\mathbf{w_t} + \mathbf{\Sigma}_t\mathbf{q_t}\left((\boldsymbol{\delta}_{\mathbf{\Sigma}})_t - (\mathbf{H}_{\mathbf{\Sigma}})_t\mathbf{w_t}\right)\right] \tag{16}$$

where $(\mathbf{H}_{\mathbf{\Sigma}})_t$ is the Hessian matrix of E with respect to $\mathbf{\Sigma}$.

The complicating factor when implementing SMD is the calculation of the Hessians in Equations (13) and (16). While there is a method of efficiently calculating the product of a Hessian and a vector, this method is extremely cumbersome for a Gaussian classifier. An alternative to using the exact Hessian is to use an approximation, such as the Levenberg-Marquardt or outer product approximation. This approximation is based on the properties of the error function, Equation 5. The elements of the Hessian with respect to the vector $\boldsymbol{\mu}$, where μ_m and μ_n are the m^{th} and n^{th} elements of $\boldsymbol{\mu}$, can be approximated by:

$$\mathbf{H}_{\mu(m,n)} = \frac{\partial^2 E}{\partial \mu_m \partial \mu_n} \approx \sum_k \frac{\partial y_k}{\partial \mu_m}\frac{\partial y_k}{\partial \mu_n} \tag{17}$$

This approximation is only valid for a well-trained network, since the elements that it ignores are only negligible on a trained network but not on an untrained network. We further simplify this approximation by neglecting the off-diagonal elements. A similar approximation is obtained for $\mathbf{H}_\Sigma$.

3 Experimental results

3.1 The IDIAP BCI

The IDIAP BCI is based around a portable BioSemi acquisition system. The electrode caps contain either 32 or 64 electrodes, arranged in the standard 10/20 International System.

The IDIAP BCI uses EEG rhythm modulation as a control signal. BCI experiments at IDIAP are generally performed in an asynchronous or self-paced paradigm — that is, the subject is not tied to a cue from the system but performs the tasks at his or her own pace, and the command signals are extracted from spontaneous brain activity. The subject is trained to perform three mental tasks while being given feedback on his or her performance. The system analyses the EEG signals to distinguish between the tasks, which may include imagination of left and right hand movement, arithmetic operations, rotation of geometrical objects, and language tasks. The most common combination is imagination of left and right hand movement and a language task, specifically vocabulary search. Classification is performed by calculating the frequency components on sliding 1/2-second windows of a selection of electrodes (in between 8 and 12) over a relevant feature band (typically between 8Hz and 30Hz, with a resolution of 2 Hz), and passing these frequency features to a statistical classifier. The classifier models the different classes as a number of Gaussians in this high-dimensional feature space, and the output of the classifier is the posterior probability of each class. This system has been used in the past to operate simple computer games, a virtual keyboard, and navigate a robot through a model house-like environment [6, 7].

3.2 Offline analysis

We tested these algorithms on a three class problem (imagination of left and right hand movements, and a third task such as word association). Each task was performed for one second before switching to a different task. Data is from three subjects, each with four sessions of almost four minutes collected with a break of ten minutes between sessions. No feedback on classifier performance was given to the subjects during data collection. Samples are passed to the classifier 16 times per second and the output from eight samples is averaged to give a decision every 0.5 seconds.

In this experiment we measured how well the classifier tracked the changing signals by applying online learning through all the sessions and measuring the classification performance. The classifier was first trained offline on the data from the first session. The resulting classifier was then applied to sessions two through four, with the final adapted classifier from the end of the previous session used as the initial classifier for the next session. We tested basic gradient descent and the SMD algorithm against the static classifier with no adaptation. Testing basic gradient descent over a range of learning parameters shows that the optimal parameters vary between data sets. From this we chose a good learning parameter for gradient descent, which also serves as the initialisation value for SMD. In these experiments both gradient descent and SMD outperform the static classifier. Figure 1 shows the performance over time of the different classifiers for the three sessions of the second subject. The classification rates of the adaptive algorithms are better than the static classifier with SMD better than gradient descent. A t-test shows that these differences are statistically significant. Similar trends are observed with the other subjects.

In addition to applying the online learning algorithms throughout the session, we wanted to see whether there is a performance gain in applying the online learning algorithms for the first half of the session only, then applying the resultant classifier to the second half of the session with no further learning. This is similar to a recalibration scenario, where we want to use supervised learning for only part of the session. Results from this experiment show that there is a small but statistically significant improvement when using the classifier trained on the first half of the data over the classifier with no further training, but the t-test shows no statistically significant difference between gradient descent and SMD.

Table 1 shows the classification results for the three subjects averaged over their three sessions, and the overall average. The results are given as the percentage improvement of the online learning algorithms over the static classifier. There are three parts: the improvement in the first half of the data, the improvement in the second half of the data when the online learning is continued, and the improvement on the second half of the data when the classifier is only trained on the first half of the data. Although there is variation between the subjects, the figures hold to the general trend as discussed above.

3.3 Initial online experiments

As an initial experiment to test feasibility of supervised online adaptation in the IDIAP BCI, we have implemented basic gradient descent to adapt the classifier during initial training. The experimental setup that we tested the system on was a computer simulation of driving a wheelchair through a corridor while avoiding obstacles. The subject was guided by an operator, who told the subject which task to attempt to produce as the wheelchair moved through the corridor. In this way the data was labelled with the target classes and the subject was learning to generate the BCI tasks while becoming used to the simulator interface. This means that the data sets differ from the previous sets analysed in that they are not necessarily balanced between classes, and the length of time each class is generated for varies. The more complicated, "real-world" setup also makes it more difficult for the subject to concentrate on the mental tasks, as he or she can be distracted by watching the wheelchair and anticipating its movements.

One performance measure used in this task was the time the subject took to steer the wheelchair to the end of the corridor and back again. Times over a number of days are shown in Figure 2.

We want to compare the online classification results against the offline performance of static classifiers. For these experiments we take an initial classifier, adapt it online throughout the session, and produce the final classifier (which then becomes the initial classifier for the next session). We measure the classification rates of the initial classifier and the final classifier on this session, and compare with the online classification rate. Tables 2 and 3 show the online classification rates of the classifier, compared to the static initial and final adapted classifiers, in terms of bit rate[1] and correct-error-rejection rates, respectively. The online classification rates are much higher than the static classifiers. Also, in each session the online adaptation produces a final classifier that outperforms the initial classifier. A t-test on the bit rates shows that differences are statistically significant.

Figures 3 and 4 show the probabilities of each sample for Session #4, where Figure 3 is the online classification rate and Figure 4 is the offline performance of the final classifier. The online classification rates track the EEG signals well, with no clear bias between classes. The final classifier can be seen to perform well on the last part of the session but less well on the early part of the session. This is consistent with drift in the signal, which means that the final classifier is tuned to the later part of the session but does not classify well on the different signals from the early part of the session.

In all the online adaptation seems to be providing consistent feedback to the subject, allowing for predictable responses from the classifier. This can be observed in the consistent online classification rates and the stable time taken to complete the task.

4 Discussion

Work until now has focused on modifying our original Gaussian classifier for online adaptation during the user training phase, where the true target class is always known and we can use a supervised learning method. In particular we have derived a SMD implementation for our Gaussian classifier, which involved finding suitable approximations to make online use feasible. In both online and offline experiments, the algorithms were performed in an online paradigm — adapting on each sample only once, in the order the samples were generated.

Although the experimental results reported here show the benefits of online adaptation, some questions need still to be addressed. The first of them is to validate the online findings during actual mental control of a robot with more subjects. We will also explore the online performance of SMD during those experiments to see whether this advanced algorithm is better at tracking the drifting signals while keeping a good model of the data. A second key question is a better understanding of EEG variation. We believe online classifier adaptation would improve the performance of a BCI because of the high variability in EEG signals, but no systematic study has been done to formally analyse the extent of signal variation through different stages in a subject's usage of a BCI. Such a study would be helpful in justifying the use of online adaptation and determining whether it is necessary in all cases. Finally, the main research issue is adaptation throughout ongoing use, where we don't have explicit information about which is the user's intent. As discussed above, reinforcement learning is an appropriate framework for this situation. In particular, we can use occasional feedback on the classifier performance coming from the recognition of cognitive error potentials or from how well the brain controlled device is operating.

Acknowledgements

This work is supported by the Swiss National Science Foundation through the National Centre of Competence in Research on "Interactive Multimodal Information Management (IM2)" and by the European IST Programme FET Project FP6-003758. This paper only reflects the authors' views and funding agencies are not liable for any use that may be made of the information contained herein.

[1] The expression that estimates the bit rate takes into account the rejection rates of the classifier as described in [7].

References

[1] N. Birbaumer, N. Ghanayim, T. Hinterberger, I. Iversen, B. Kotchoubey, A. Kubler, J. Perelmouter, E. Taub, and H. Flor. A spelling device for the paralysed. *Nature*, 398:297–298, 1999.

[2] P. Ferrez and J. Millán. You are wrong!—Automatic detection of interaction errors from brain waves. In *Proc. 19th Int. Joint Conf. Artificial Intelligence*, 2005.

[3] J. Kivinen and M. K. Warmuth. Additive versus exponentiated gradient updates for linear prediction. In *STOC '95: Proceedings of the twenty-seventh annual ACM symposium on Theory of computing*, pages 209–218, New York, NY, USA, 1995. ACM Press.

[4] J. Millán. Brain-computer interfaces. In M. A. Arbib, editor, *Handbook of Brain Theory and Neural Networks*. The MIT Press, second edition, 2002.

[5] J. Millán. On the need for on-line learning in brain-computer interfaces. In *Proc. Int. Joint Conf. Neural Networks*, 2004.

[6] J. Millán, F. Renkens, J. Mouriño, and W. Gerstner. Brain-actuated interaction. *Artificial Intelligence*, 159:241–259, 2004.

[7] J. Millán, F. Renkens, J. Mouriño, and W. Gerstner. Non-invasive brain-actuated control of a mobile robot by human EEG. *IEEE Transactions on Biomedical Engineering*, 51:1026–1033, 2004.

[8] D. Saad, editor. *On-Line Learning in Neural Networks*. Cambridge University Press, 1998.

[9] R. Scherer, G. Müller, C. Neuper, B. Graimann, and G. Pfurtscheller. An asynchronously controlled EEG-based virtual keyboard: Improvement of the spelling rate. *IEEE Transactions on Biomedical Engineering*, 51(6):979–984, 2004.

[10] N. Schraudolph. Local gain adaptation in stochastic gradient descent. In *Proc. 9th Int. Conf. Artificial Neural Networks*, 1999.

[11] R. S. Sutton. Adapting bias by gradient descent: An incremental version of delta-bar-delta. In *Proceedings of the Tenth National Conference on Artifical Intelligence*, pages 171 – 176. MIT Press, 1992.

[12] R. S. Sutton and A. G. Barto. *Reinforcement Learning: An Introduction*. MIT Press, 1998.

[13] J. Vidal. Real-time detection of brain events in EEG. *Proceedings IEEE*, 65:633–641, 1977.

[14] J. Wolpaw, N. Birbaumer, D. McFarland, G. Pfurtscheller, and T. Vaughan. Brain-computer interfaces for communication and control. *Clinical Neurophysiology*, 113:767–791, 2002.

[15] J. Wolpaw, D. McFarland, T. Vaughan, and G. Schalk. The Wadsworth Center brain-computer interface (BCI) research and development program. *IEEE Transactions on neural Systems and Rehabilitation Engineering*, 11(2):204–207, 2003.

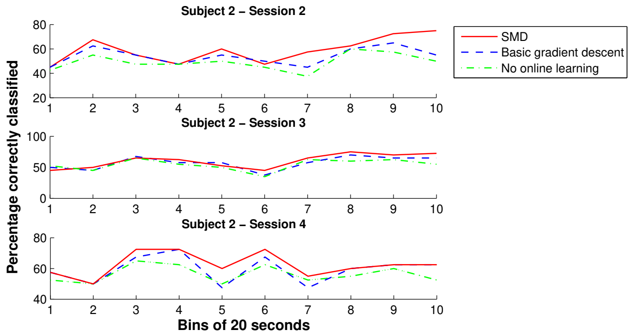
Subject 2 – Session 2
Subject 2 – Session 3
Subject 2 – Session 4
Percentage correctly classified
Bins of 20 seconds
SMD
Basic gradient descent
No online learning
80
60
40
20
100
50
0
1
2
3
4
5
6
7
8
9
10

Figure 1:

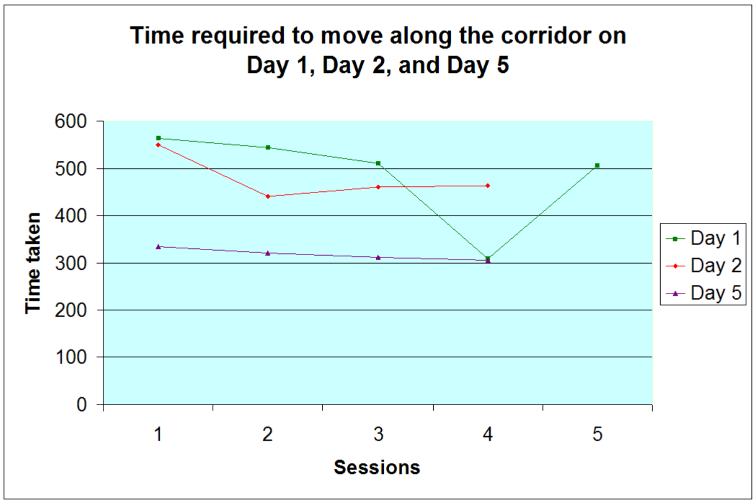
Time required to move along the corridor on
Day 1, Day 2, and Day 5
Time taken
600
500
400
300
200
100
0
1
2
3
4
5
Sessions
Day 1
Day 2
Day 5

Figure 2:

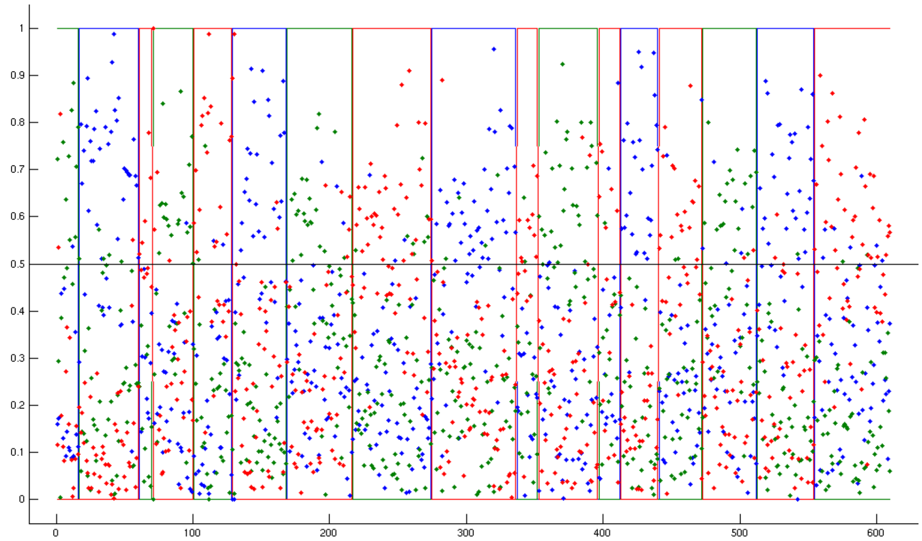

Figure 3:

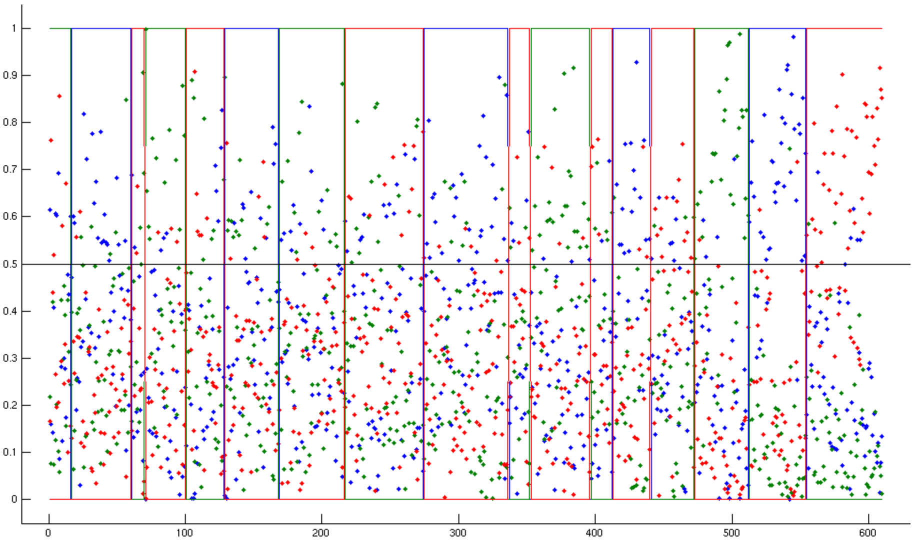

Figure 4:

Table 1:

		First half	Second half	
		Adapting	Adapting	Not adapting
Subject 1	GD	3.5	7.9	3.9
	SMD	4.9	9.3	5.9
Subject 2	GD	6.1	7.8	3.1
	SMD	9.3	18.6	5.2
Subject 3	GD	18.3	18.7	7.1
	SMD	28.0	33.0	2.5
Average	GD	9.3	11.5	4.7
	SMD	14.0	20.3	4.5

Table 2:

Session #	Initial classifier	Online classification	Final classifier
#1	0.29	1.44	0.65
#2	0.20	1.41	0.67
#3	0.14	1.34	0.71
#4	0.18	1.34	0.67
Average	0.20 ± 0.06	1.38 ± 0.05	0.67 ± 0.02

Table 3:

Session #	Initial classifier Cor - Err - Rej	Online classification Cor - Err - Rej	Final classifier Cor - Err - Rej
#1	20.1 - 37.5 - 42.3	64.3 - 11.7 - 24.0	40.3 - 26.4 - 33.3
#2	26.9 - 45.0 - 28.1	63.9 - 12.2 - 23.8	43.3 - 26.9 - 29.8
#3	23.6 - 48.7 - 27.7	62.2 - 13.4 - 24.4	41.0 - 24.2 - 34.8
#4	23.9 - 46.3 - 29.8	61.1 - 12.8 - 26.1	41.3 - 26.1 - 32.6
Av. Cor	23.6 ± 3.0	62.9 ± 1.5	41.4 ± 1.3
Av. Err	44.4 ± 5.3	12.5 ± 0.7	25.9 ± 1.2
Av. Rej	32.0 ± 8.0	24.6 ± 1.0	32.6 ± 2.1

List of Figures

List of Tables