A comparison of regression techniques for a two-dimensional sensorimotor rhythm-based brain–computer interface

2010 J. Neural Eng. 7 016003

(http://iopscience.iop.org/1741-2552/7/1/016003)

J. Neural Eng. **7** (2010) 016003 (9pp) doi:10.1088/1741-2560/7/1/016003

A comparison of regression techniques for a two-dimensional sensorimotor rhythm-based brain–computer interface

Joan Fruitet[1], **Dennis J McFarland**[2] **and Jonathan R Wolpaw**[2]

[1] Ecole Normale Supérieure, 45 rue d'Ulm, 75230 Paris cedex 05, France
[2] Laboratory of Neural Injury and Repair, Wadsworth Center, New York State Department of Health and State University of New York, Albany, NY 12201, USA

E-mail: joan.fruitet@gmail.com

Received 19 May 2009
Accepted for publication 16 December 2009
Published 14 January 2010
Online at stacks.iop.org/JNE/7/016003

Abstract
People can learn to control electroencephalogram (EEG) features consisting of sensorimotor-rhythm amplitudes and use this control to move a cursor in one, two or three dimensions to a target on a video screen. This study evaluated several possible alternative models for translating these EEG features into two-dimensional cursor movement by building an offline simulation using data collected during online performance. In offline comparisons, support-vector regression (SVM) with a radial basis kernel produced somewhat better performance than simple multiple regression, the LASSO or a linear SVM. These results indicate that proper choice of a translation algorithm is an important factor in optimizing brain–computer interface (BCI) performance, and provide new insight into algorithm choice for multidimensional movement control.

(Some figures in this article are in colour only in the electronic version)

1. Introduction

Many people with severe motor disabilities require alternative methods for communication and control. Numerous studies over the past two decades have shown that scalp-recorded EEG activity can be the basis for non-muscular communication and control systems, commonly called brain–computer interfaces (BCIs) (Birbaumer *et al* 1999, Farwell and Donchin 1988, Pfurtscheller *et al* 1993, Wolpaw *et al* 1991). EEG-based communication systems measure specific features of EEG activity and use the results as control signals. Some BCI systems use features that are potentials evoked by stereotyped stimuli (Farwell and Donchin 1988). Others use EEG components in the frequency domain that are spontaneous in the sense that they are not dependent on specific sensory events (e.g. Wolpaw and McFarland (2004)).

Effective BCI operations depend on use of appropriate methods for recording brain signals, extracting features from these signals and translating these features into device commands (Wolpaw *et al* 2002). The feature translation algorithm is extremely important, and has been the subject of numerous studies and data competitions (e.g. Blankertz *et al* (2006), see McFarland *et al* (2006a) for a review). An effective translation algorithm weights relevant features in proportion to the information they contain about the user's desired outcome. Many kinds of algorithms, linear as well as nonlinear, are possible (e.g. Muller *et al* (2003)).

With the Wadsworth sensorimotor rhythm (SMR)-based BCI system, users learn over a series of training sessions to use SMR amplitudes in mu (8–13 Hz) and/or beta (18–27 Hz) frequency bands recorded over left and/or right sensorimotor cortex to move a cursor in one, two or three dimensions (see McFarland *et al* (2006) for full system description). At present, the translation algorithm used for online control is a simple linear multiple regression.

This study compares the present translation algorithm to several promising alternatives in terms of performance in two-dimensional SMR-based cursor control. The algorithms were compared through offline analyses of data recorded while subjects controlled the cursor with the standard

algorithm. This is the first formal comparison of alternative translation algorithms in regard to two-dimensional control. Multidimensional control presents unique challenges, such as the need for independence among the signals that control the separate dimensions (e.g. Wolpaw and McFarland (2004)).

2. Methods

2.1. Data collection and preprocessing

2.1.1. BCI users. The BCI users were eight adults (five men and three women, ages 20–39). Six had no disabilities, while two had spinal cord injuries (at C6 and T7) and were confined to wheelchairs. All gave informed consent for the study, which had been reviewed and approved by the New York State Department of Health Institutional Review Board. After an initial evaluation defined the frequencies and scalp locations of each person's spontaneous mu and beta rhythm (i.e. sensorimotor-rhythm (SMR)) activity, he or she learned EEG-based cursor control over several months (two to three 24 min sessions/week). The standard online protocol, which has been described in the previous publications (McFarland *et al* 1997a, Schalk *et al* 2004, Wolpaw and McFarland 2004), is summarized here.

2.1.2. Standard online protocol. The user sat in a reclining chair facing a 51 cm video screen 3 m away, and was asked to remain motionless during performance. Scalp electrodes recorded 64 channels of EEG (Sharbrough *et al* 1991), each referenced to an electrode on the right ear (amplification 20 000, bandpass 0.1–60 Hz and sampling rate 160 Hz).

A daily session consisted of eight 3 min runs separated by 1 min breaks, and each run had 20–30 trials. Each trial consisted of a 1 s period from target appearance to the beginning of cursor movement, a period of cursor movement of 15 s maximum, a 1.5 s post-movement reward period and a 1 s inter-trial interval. Users participated in two to three sessions/week at a rate of one session every 2–3 days. The last seven sessions for each user were used for this offline analysis. The total number of available trials ranged between 1022 and 1579.

To control each dimension of cursor movement (vertical or horizontal), one EEG channel over the left sensorimotor cortex (i.e. electrode locations C3 or CP3) and/or one channel over the right sensorimotor cortex (i.e. C4 or CP4) were derived from the digitized data according to a Laplacian transform (McFarland *et al* 1997b). Every 50 ms, the most recent 400 ms segment from each channel was analyzed by a 16th-order (McFarland and Wolpaw 2008) autoregressive model using the Berg algorithm (Marple 1987) to determine the amplitude (i.e. square root of power) in a 3 Hz wide mu or beta frequency band, and the amplitudes of the one or two channels were used in a linear equation that specified a cursor movement in that dimension as described by Wolpaw and McFarland (2004). Thus, the cursor moved 20 times/s in the vertical and horizontal dimensions simultaneously. Complete EEG and cursor movement data were stored for later offline analyses.

2.1.3. Offline preprocessing and feature extraction. First, because the 64 electrodes are digitized sequentially (the value number n of the electrode number e is not recorded at the time $t_0 + \frac{n}{\text{Sampling\_rate}}$ but $t_0 + \frac{n}{\text{Sampling\_rate}} + \frac{e-1}{64\times\text{Sampling\_rate}}$ where t_0 is the time when the recording starts) and further processing compares the values at different electrodes that are assumed to occur at the same time, the data were temporally aligned using a cubic spline interpolation. A cubic spline is a piecewise cubic polynomial which is twice continuously differentiable. The interpolation of a data set $\{x_i, y_i\}_{1\leqslant i\leqslant n}$ of n points such as $x_i < x_{i+1}$ with a cubic spline consists in finding a function F that satisfies

$$F(x) = \begin{cases} P_1(x), & x \in [x_1, x_2] \\ \cdots & \text{where} \quad P_1 \ldots P_{n-1} \text{ are cubic polynomial} \\ P_{n-1}(x), & x \in [x_{n-1}, x_n]. \end{cases}$$

$$\forall\ 1 \leqslant i \leqslant n, \quad F(x_i) = y_i$$

$$\forall\ 1 \leqslant i < n, \qquad P_i'(x_{i+1}) = P_{i+1}'(x_{i+1}),$$

$$P_i''(x_{i+1}) = P_{i+1}''(x_{i+1})$$

$$P_1''(x_1) = P_{n-1}''(x_n) = 0$$

Second, in accord with McFarland *et al* (1997), a large Laplacian spatial filter was applied to enhance the signal to noise ratio.

Third, a spectral analysis using a 16th-order autoregressive model determined the power in the seventeen 3 Hz wide spectral bins centered at frequencies from 8 to 24 Hz. The logarithms of these power values served as the features that were translated into cursor movements. As in online cursor control, the offline analysis used a 400 ms window that shifted in 50 ms steps. Thus, the cursor movement was computed 20 times/s. To explore new control possibilities more fully, these features were determined for ten electrodes located over the sensorimotor cortex (FC3, C5, C3, C1, CP3, FC2, C2, C4, C6 and CP4), not simply for the two electrodes (C3 and C4 for most users) that had actually been used for online control.

Thus, each 400 ms window of data was described by a vector of features, the dimension of which was the number of electrodes (10) times the number of frequency bins (17), for a total of 170 features. This differs from the protocol that had been used online, in which cursor movement in each dimension was a linear function of only 1–4 features.

2.2. Two approaches to comparing the different translation algorithms

2.2.1. Using the whole trial. The simplest approach to offline analysis is to determine from the full data of each trial the position of the target to which the user had to move the cursor. For this approach, the features used by the translation algorithm are the means of the feature values for all the 400 ms windows of each trial. The number of features used to describe each trial is then equal to the number of electrodes times the number of frequencies used (i.e. 170).

Although this determination of target position might be addressed by a classification algorithm, we used instead a set of regression algorithms for reasons detailed in McFarland and Wolpaw (2005). (Briefly, regression is preferable to classification because it is better suited to controlling

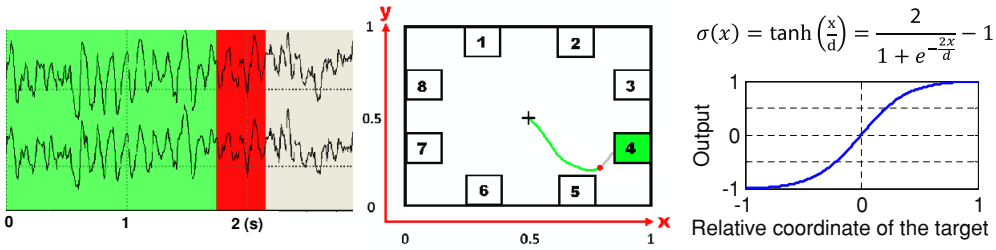

Figure 1. Left: EEG recorded from two electrodes during one trial. At zero time the target appears. One second later the cursor appears and begins to move. One of the sliding windows is highlighted (in red). Center: hypothetical example showing the eight possible target locations, and the target for the current trial (in green). The green (gray) curve represents the cursor movement generated by the offline analysis up to (after) the time of the red window. The red sliding window must be used to move the red dot. Right: the sigmoid function. If the target is far from the cursor, the exact relative position of the target does not mater, just the direction is important. This is why, for example, the output for the relative coordinate of (0.5; 0.5) is very close to the one of (1; 1), meaning that in both cases the user should move fast to the right and to the top.

continuous cursor movements in real time and it more readily generalizes to novel target locations.)

The primary result of offline analysis, and the measure used to compare alternative regression algorithms, was the proportion of the variance in target position that was predicted by the algorithm (i.e. r^2).

2.2.2. Using the individual 400 ms windows. The whole-trial approach to offline analysis is not readily transferable to actual online use. During online cursor control, cursor movements are made every 50 ms based on the most recent 400 ms window. Thus, in order to obtain offline results from methods that would be compatible with online use, we also compared the alternative translation algorithms using the features extracted from the individual 400 ms windows.

Another difference between online and whole-trial offline analysis is the user's ability to use visual feedback (i.e. cursor position) to make corrective movements. For example, as seen in figure 1, prior cursor movement positioned the cursor so that to reach the target the user needed to move the cursor up, even though the target was lower than the initial starting point. To take account of such effects, the desired output for the regression should not be the absolute coordinates of the target, but rather the coordinates of the target relative to the cursor position.

To make our model realistic, we introduced a delay to take into account the time the user needs to perceive the feedback and correct his trajectory. This is why the relative coordinate of the target was computed as the coordinates of the target minus the coordinate of the cursor a short time (i.e. the delay) before the start of the sliding window.

Unfortunately, it is not possible to directly extract the position of the target from a single window of data since the quality of the EEG signal and the precision of the extracted features are not sufficient to locate the target with only 400 ms of signal. On the other hand, a single window should have more information than just the direction of the target. That is, when the cursor is far from the target the user may try to move faster, so that each window may also contain information about the distance between the cursor and the target. This is why a sigmoid function (figure 1(c)) is used to take into account the distance of the target while limiting extreme cursor movements.

This individual windows approach consists in deducting from each 400 ms window of the signal the relative direction/position of the target. The performances of the different regression methods are then evaluated in terms of the person's r^2.

2.2.3. Evaluating the performance of the translation algorithms. Estimating the performance of a method is an important issue in machine learning (Kohavi 1995). The simplest technique, sometimes called holdout or test sample estimation (Lachenbruch and Mickey 1968), consists in partitioning the data (i.e. the vectors of features of all the trials when using the whole-trial method and of all the windows when using the single-window method) into two subsets: a training set that is used to train the method and a testing set that is used to evaluate its performances. To obtain unbiased results, the data used for testing must be different from the data used for training; otherwise, we would evaluate the capacity of the method to 'remember' the data and not how it will be able to generalize what it learnt to a new sample. This technique reaches its limits when the total amount of data is limited, which is the case here. Indeed, if there are not enough training data, the algorithm will not be able to learn what best predicts the criterion variable and therefore will not generalize well to new examples. On the other hand, if there are not enough testing data, the results may not be reliable.

More complex techniques, like the bootstrap (Efron and Tibshirani 1993) or the cross validation (Mosier 1951, Stone 1974, Kohavi 1995), were developed to obtain more precise evaluations of the performances. The n-fold cross validation technique consists in splitting the data into n parts (instead of two for the holdout), each of which will be used alternatively to train or to test the algorithm (figure 2). The final performance is the mean of the performances on the n different testing sets.

In our case the data are already split into seven parts (the seven last sessions recorded during seven different days). Using a sevenfold cross validation enables us to obtain results compatible with the online experiment. The methods are trained on six sessions and are tested on a different session

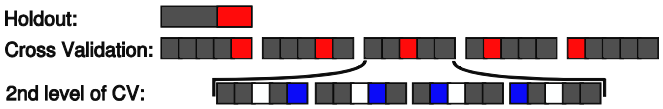

Figure 2. Holdout: one data set is used for training (gray) and the other for testing (red). Cross validation: the data are split into n parts. A total of n analyses are performed. Each part is used as the testing set (red) for one analysis while the $n-1$ other parts are used as the training set (gray). To adjust the parameters, a second level of cross validation is made; before each of the n analyses, $n-1$ analyses are made (training sets are in gray and testing sets in blue). To obtain an unbiased result, the testing data set of the first level of cross validation (red) is not used during the second level of cross validation (white).

that was recorded during a different day. In this way we can test the robustness of the methods, meaning that the signal characteristics may change from the training and testing data sets due to slight changes of the electrodes position or of the different moods of the user.

Some of the methods have parameters that need to be adjusted (like the C of SVR or the width σ of the RBF kernel). To obtain unbiased results, the parameters are set by doing a sixfold cross validation on each of the training sets of the sevenfold cross validation (figure 2). To optimize the results, the features are normalized (for every trial, each feature is normalized according to the mean and standard deviation of its values over the training trials only; in this way it is compatible with online analysis).

In order to evaluate the use of doing cross validation a three-way split analysis was also performed, using the first four sessions for training, session five for selecting the parameters and the last two sessions to evaluate the performances. The three-way split analysis resulted in a lower performance on the test set of about 15% of r^2.

2.2.4. Quantifying the differences between methods. During online experiments the performances of the users can be measured by the percentage of hit targets. Although this could be done for the offline analysis, it would be biased for two reasons. First during online analysis the users correct their trajectories according to the feedback and those same corrections are used during offline analysis even if they are not appropriate. Secondly, during the online experiment there is only one target present on the screen for each trial, though the only way to fail a trial is if the user cannot reach this target within 20 s. (The online experiment was designed to teach the users how to move the cursor and not to evaluate their performances. That is why only one target was present at the same time.)

Because we are evaluating regression techniques, a natural measurement of the performances is the correlation of the output of the regression with the objective (i.e. the position of the target for the global approach or the relative direction of the target for each sliding window).

Unfortunately, it is not possible to directly translate r^2 into performance accuracy. The only thing that can be said is that the closer it is to 1, the more the user would have been moving toward the target if he had used this method.

Another issue is that the results vary a lot across users and vary depending on which series are used for training and testing, though the benefit from using a particular method for the regression can be hidden by the natural variation of the results. A solution is to measure the performance of each method as the ratio of its results (given as the person's r^2) over the results of a method of reference.

The objective is then to compute the increase (in percent) of the results by using a particular method compared to a method of reference, namely multiple linear regression, as well as a confidence interval of this increase.

First, for each of the n parts of the n-fold cross validation and each user, we compute the logarithm of the ratio of the result (in terms of r^2) of the two methods. The mean and standard deviation are then calculated and used to compute the confidence interval of the logarithm of the ratio according to

$$\left[\bar{r} - \frac{t_{1-\alpha}^{m-1}}{2}\frac{S}{\sqrt{m}}, \bar{r} + \frac{t_{1-\alpha}^{m-1}}{2}\frac{S}{\sqrt{m}}\right],$$

where $\bar{r}$ and S are the mean and standard deviation of the logarithm of the ratio of the results, m equals n times the number of users, $1-\alpha$ is the confidence level and t_γ^k is the quantile of order γ of the Student law with k degrees of freedom.

The increase can be computed by the following equation:

$$\text{incr} = 100 \times (e^{\bar{r}} - 1),$$

which is also applied to the boundaries of the previous interval to compute the final confidence interval. The use of the logarithm and then the exponential is necessary because we are using ratios and not differences—for the mean it is equivalent to computing the geometric mean.

2.3. The different regression techniques

Three different regression techniques are compared. The first technique is 'multiple linear regression'. It is closest to the technique that was actually used online, and thus is the one to which the other techniques are compared.

The second technique is the 'LASSO regression' (Tibshirani 1996), which adds a penalty to the multiple linear regression to make it generalize better. More precisely, LASSO regression consists in minimizing the square of the $l2$ norm of the difference between the goal and output, and an $l1$ penalty. It can be written as

$$\min_\beta \|Y - X\beta\|_2^2 + \lambda \sum_{i=1}^{p} |\beta_i|,$$

where $Y \in \mathbb{R}^n$ is the desired output, $X \in \mathbb{R}^{n\times p}$ is the features matrix and $\beta \in \mathbb{R}^p$ is a parameter vector.

The third technique is the application of support vector machines (Vapnik 1995, 1998a, 1998b) to regression, also called SVR (Smola 1996). It is used with two different kernels: a linear kernel and a radial basis function kernel:

$$k(x, y) = e^{\frac{-\|x-y\|^2}{2\sigma^2}}.$$

2.4. Feature selection

2.4.1. Critical issues. In one sense the higher the dimension of the feature space (i.e. the more electrodes and frequencies used) the more information on each trial or each sliding window is available, and so the more accurate the cursor control can be. Unfortunately, not all features contain useful information and instead may add mostly noise. This is why using more features can appear to increase the information, but actually decreases the information to noise ratio. When too many features are used, every sample (trial or sliding window) of the training data set can be predicted from noise alone. The performances will then be close to perfect on the training set and catastrophic on the testing one. The solution to this problem is to try to select the features that will add useful information without causing a decrease in global performance.

Two different approaches to features selection can be found in the machine learning literature: the so-called filter and wrapper methods (Blum and Langley 1997). The filter methods consist in eliminating the irrelevant features (or selecting the useful ones) before the use of the learning algorithm. In contrast, the wrapper methods test the learning algorithm with different subsets of features to try to select the optimal subset. The wrapper approach can provide better feature selection (Kohavi and John 1995), but generally implies a higher computational cost than filter methods due to the multiple trainings of the learning algorithm with different feature subsets.

The time needed to train the SVR (especially with the RBF kernel when using large amount of features) being too long to use the wrapper method, we will focus on the filter approach and propose a fast and efficient way to select the features.

The filter methods can be subdivided into two categories. First, the methods that evaluate the use of each feature separately. The usefulness of a feature can be measured by its correlation with the desired output (here the optimal cursor movement) or by its mutual information, to take into account nonlinearity (McFarland *et al* 2006a). Secondly, the methods that evaluate in one comprehensive analysis the usefulness of all the features. For example, if the features are normalized, the coefficients of a regression that incorporates all the features as independent variables indicate the usefulness of each feature (e.g. the LASSO method is often used this way to select features (Tibshirani 1996)).

The major drawback of most of the methods of the first category is that they do not take into account the redundancy of the information. For example, if two features contain very similar information, they will certainly be selected together, although only one may be useful, the other could just add more noise. This is particularly true in our case, where features extracted from neighbor electrodes and/or at close frequencies, will contain similar information and highly correlated noise. To deal with this issue it is possible to recursively select the features that add the most information to the group of features already selected. We can then order the features in a way that takes into account the redundancies of the information, and uses only those features that add useful information.

Table 1. Feature ordering algorithm. *best_feature*(A, B) is a function that gives the index of the column of A that contains the most information about B. $A_{\_,j}$ represents the jth column of A seen as a vector of $\mathbb{R}^m$, $\langle X|Y\rangle = X^T \cdot Y$ is the Euclidian scalar product and $\|X\| = \sqrt{\langle X|X\rangle}$ is its associated norm.

Feature ordering algorithm	
(1)	*for* $i = 1..n$
(2)	$ind = best\_feature(A, B)$
(3)	$features\_order(i) \leftarrow ind$
(4)	*for* $j = 1..n$
(5)	$A_{\_,j} \leftarrow A_{\_,j} - \frac{\langle A_{\_,j}\|A_{\_,ind}\rangle}{\|A_{\_,ind}\|^2} A_{\_,ind}$
	end
	end

This is a greedy algorithm (i.e. it makes locally optimal choice at each stage with the hope of finding the global optimum) that is not necessarily optimal, since it is not certain that the subset of the features selected is the one that contains the most information. Nevertheless, determining the best subset would be extremely time consuming, and this algorithm should provide a good approximation to the optimum solution.

2.4.2. The feature selection algorithm. Let $A \in \mathbb{R}^{m\times l}$ be the matrix representing the training data, where $l = n_{\text{Elec}} \times n_{\text{frequencies}}$ is the dimension of the features space (or the number of features) and m is the number of data points (in our case the total number of sliding windows for all the trials). $A_{i,j}$ is then the value of the jth feature for the ith sliding window.

Let $B \in \mathbb{R}^{m\times 2}$ be the matrix of the desired output. The two dimensions of the matrix represent the horizontal and vertical control command for every sliding window.

The algorithm is described in table 1. The first step is to select the feature that contains the most information about the desired output. In our case, it is evaluated by the correlation of the feature with the output. The second step consists in removing from the other features the information that is contained in the feature that has just been selected. If we consider the features as vectors in the Euclidian space $\mathbb{R}^m$, this step is exactly the orthogonal projection of the remaining features in the direction of the previously selected feature. The algorithm will continue by selecting the next feature according to the first step and so on. More precisely, the algorithm saves the result of the first evaluation of the usefulness of the features and selects the new features according to the new evaluation (meaning after the multiple projections) and the original evaluation. This guarantees that it selects only features that originally contained unique information.

Once the features have been ordered, the p first ones will be used for the regression (where p is a parameter that has been adjusted by cross validation).

2.4.3. Improving the RBF kernel. The feature selection algorithm does not take into account the fact that the first selected feature is generally more important than the feature

number p. In the case of the SVR with the RBF kernel, we can define a norm that takes into account this difference in importance between the features by adding a weight to each component. If x is the jth data point ($x = A_{-,j}$), x_i being the value of the ith feature for the jth data point ($x_i = A_{i,j}$), we define the norm of x by

$$\|x\|_w^2 = \frac{1}{\sum_{i=1}^{n} w_i} \sum_{i=1}^{n} w_i x_i^2$$

with $\quad w \in]0, 1]^n \quad$ and $\quad \forall\, i < j, \quad w_i \geqslant w_j.$

The modified kernel is now

$$k(x, y) = \mathrm{e}^{\frac{-\|x-y\|_w^2}{2\sigma^2}}.$$

The vector of weights is defined by

$$w_i = \frac{\beta}{\beta + i^\alpha - 1},$$

where α and β are the parameters that have to be adjusted. To decrease the time necessary for the training of the SVR, it is possible to set the weights w_i to 0 when $i > p$. This leads to the use of only the p first features, but is a good approximation when w_p is small.

3. Results

3.1. Comparison of regression techniques when using the whole trial

Figure 3 shows that the linear techniques (multiple regression, LASSO regression and SVR with a linear kernel) give very similar results on the test data set. The test data set results are improved by the SVR technique with a radial basic function kernel, particularly when more data are used for training. The results on the test set for the SVR with the RBF kernel are much lower than the ones on the training set, reflecting most likely an over-fitting issue. This can be solved by a better features selection, which is done by modifying the RBF kernel (see section 2.4.3 and results in figure 6).

However, those results have to be interpreted with caution. First, in order to obtain results from the training data sets of different size (in percent of the total data), trials from the same session were split between the training and testing data sets, and so, the result of figure 3 may be biased. In the remainder of the paper all the results will be obtained with a training data set of 85% of the total data corresponding to a sevenfold cross validation (this way session will not be split between training and testing data sets).

Secondly, figure 3 was obtained with the global trial approach. One could make the hypothesis that even if the translation of each sliding window to the cursor movement is a linear process, the translation of the global trial into the position of the target could be nonlinear, which would explain the better result of the SVR with the RBF kernel.

Therefore, the next section will be dedicated to adjust and validate the model used for the sliding window approach and section 3.3 will focus on a deeper comparison of the SVR with a RBF kernel to multiple linear regression.

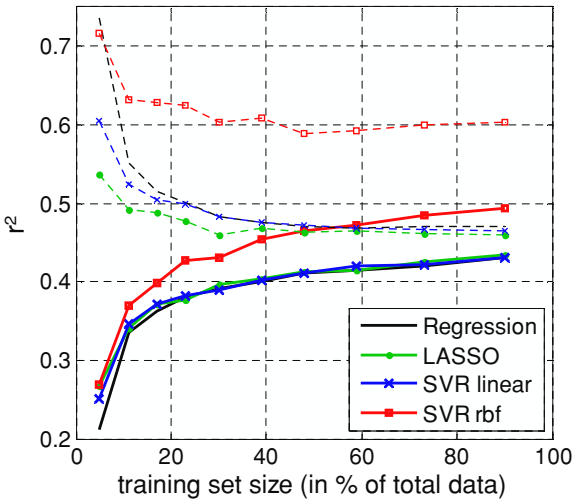

Figure 3. Results on the training (dashed lines) and testing (full lines) data sets in terms of the square of Pearson's correlation coefficient as a function of the size of the training data set for the four different methods.

3.2. Using the multiple linear regression technique to compare the influence of different parameters when working with sliding windows

To do a realistic offline analysis using sliding windows, a model of the interaction between the user and the BCI had to be built. Before using this model to compare feature translation algorithms it has to be validated. This model is mostly based on the assumption that the users take into account the position of the cursor and aim toward the target rather than just focusing on the target. One way of validating this assumption is to calculate the influence of using the cursor position (which is done in the next section) as well as the benefit of using the sigmoid function.

To be as close as possible to online analysis, the multiple linear regression technique was used, with only two electrodes and no features selection.

3.2.1. Influence of the delay used for the sliding windows method. A very long delay (e.g. Inf in figure 4) represents the case in which the cursor's prior movement is not taken into account (i.e. the infinite delay means that the position stays at the starting point). For all the users, except two, taking into account the cursor position increases the correlation between the output and the target direction (figure 4 and table 2). This increase is maximal for a specific delay, which varies across the users. These results confirm our hypothesis that the users take into account the position of the cursor.

3.2.2. Using the sigmoid function to compute the relative position of the target. The effects on performance from using the sigmoid function are shown in table 3.

3.2.3. Comparison of interpolation techniques. Figure 5 shows the influence of the interpolation method on the results of the multiple linear regression when working with sliding

Table 2. Increase of the results on the testing data set (in percent of r^2) by taking into account the cursor position compared to using only the target position. The best delay is the delay for which the increase is maximum. The values in bold are significant ($p < 0.05$).

Users	A	B	C	D	E	F	G	H	Mean
Increase (%)	**29**	**22**	**13**	−5	**18**	6	**17**	6	**12.8**
Confidence interval at 95%	15; 46	7; 39	3; 24	−14; 6	8; 30	−18; 38	8; 26	−7; 21	7.7; 18.1
Best delay (s)	0.75	0.5	0.75	2	0.3	0.2	0.5	0.2	0.65

Table 3. Increase of the results on the testing data set (in percent of r^2) by using the sigmoid function to compute the relative position of the target. The values in bold are significant ($p < 0.05$). (The multiple linear regression technique was used with the sliding windows approach, two electrodes and features selection.)

Users	A	B	C	D	E	F	G	H	Mean
Increase (%)	−1.53	8.22	1.5	−3.49	**3.12**	23.9	**5.77**	**7.7**	**5.4**
Confidence interval at 95%	−3.8; 0.8	−0.4; 17	−3.3; 7.7	−2.2; 9.1	2.3; 4.6	−1.9; 56	2.7; 9.7	3.6; 11.9	4.1; 6.7

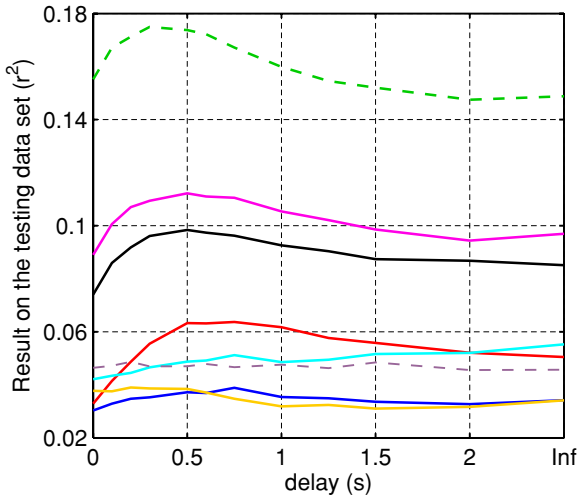

Figure 4. Results on the testing data set (as r^2) as a function of the delay. Each user is represented by a different color. The two users with spinal cord injuries are represented by dashed lines. The right end of the curves is with an infinite delay (i.e. the previous cursor movement is not taken into account, so that the desired output of the regression is the absolute coordinate of the target and does not depend on the cursor position). (The multiple linear regression technique was used with the sliding windows approach, two electrodes and features selection.)

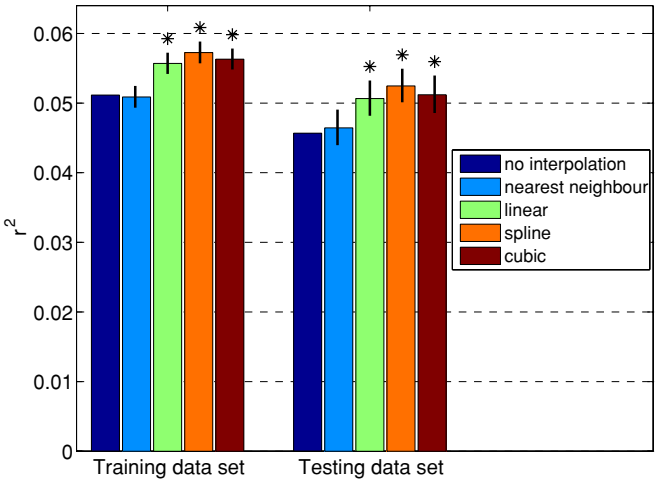

Figure 5. Average results in terms of r^2 for eight users (with the multiple linear regression technique, the sliding windows approach, two electrodes and features selection), when no interpolation is used (dark blue) and for four different interpolation methods. The error bars represent the interval of confidence at 95% of the increase of the results compared to no interpolation. Except for the nearest neighbor, all the interpolation techniques significantly increase the results.

windows. A cubic spline interpolation improves the results by 15% ([9.7% 20.3%] $p < 0.05$) compared to no interpolation. This result underscores the importance of temporally aligning the signal from the different electrodes. The cubic spline interpolation also enhances the result by 3.6% ([0.9% 6.3%] $p < 0.05$) compared to a linear interpolation that was used during online analysis and by 2.5% ([0.1% 4.9%] $p < 0.05$) compared to a cubic interpolation.

These results may be explained by the fact that the cubic splines are twice continuously differentiable, which makes them more adequate to model continuous phenomenon, such as oscillatory rhythms, than piecewise affine functions (used in linear interpolation) or piecewise cubic Hermite polynomials (which are just once continuously differentiable).

3.3. Comparing multiple linear regression and SVR with a RBF kernel, with and without features selection, when using sliding windows

For each user, the seven sessions were used for a sevenfold cross validation. The feature normalization and the adjustment of the parameters (number of features to use for the multiple regression; weights of the features, σ and C for the SVR) were done on the training data sets only, by another level of cross validation (figure 2).

Early results, like the global trial approach, have shown that SVR with a linear kernel was not able to enhance the performances compared to multiple linear regression. We then focus on the comparison of SVR with a RBF kernel and multiple linear regression as well as the impact of selecting features.

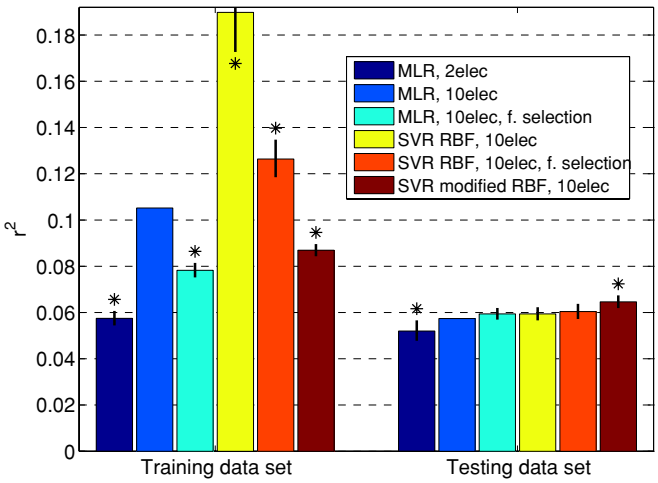

Figure 6. Average results (as r^2) for eight users, for multiple linear regression (MLR) and SVR with a RBF kernel, when working on sliding windows. Feature-unique weights for the RBF were calculated by the feature selection algorithm. The error bars represent the interval of confidence at 95% of the increase of the results compared to MLR with ten electrodes and no features selection. The stars show methods significantly different from ten electrodes and no feature selection.

Figure 6 shows the average increase, for all users, of SVR with RBF compared to using multiple linear regression with ten electrodes. The relative increase of r^2 resulting from using SVR with the modified RBF kernel over the use of multiple regression when using ten electrodes is 12.6% and is significant ([7.9%; 17.4%] $p < 0.05$). The relative increase of r^2 reaches 24.2% ([15.8%; 33.7%] $p < 0.05$) when comparing SVR with ten electrodes to multiple regression with the two electrodes that are used during online analysis. The results for SVR with the standard RBF kernel are not significantly different from MLR with ten electrodes, which underlines the importance of appropriate feature selection.

The important differences between the results on the training sets and the testing sets mean generalization to new examples is difficult and suggest, especially for the SVR, that the use of additional training data could improve the result.

4. Discussion

4.1. Importance of the user correction

The user's task was simply to move the cursor from the middle of the screen to one of the targets, and so could in theory be done without any feedback. Nevertheless, taking into account the cursor position during offline analysis improves the performance by about 13%. This fact suggests that the users (or at least some of them) are using the feedback to control the cursor trajectory in real time, meaning that they are able to correct the cursor movement and not just aim at the target during the whole trial. Moreover, the delay that represents the user reaction time, about 600 ms, is very similar to the reaction time when subjects perform a conventional motor task. These results are in general agreement with studies of manual reaction times (e.g. Albert *et al* (2007)) and our previous study of a different mu-rhythm task (Friedrich *et al* 2009).

It is possible to look at this BCI as a way to select among eight possible targets and so to compute the bit rate transmission, which is often used as a measure of BCI performance. But this view does not take into account the main purpose of this BCI, namely, allowing the user to control the cursor in real time. The results suggest that using this BCI is similar to using a standard pointing device such as a computer mouse.

One of the goals of this study was to develop a model of online experiments to enable realistic offline analysis using online recorded data. The benefits of taking into account the cursor movement support our model. The next step to validate our model will be to confirm with an online analysis the prediction of our model (namely the advantage of SVR with the modified RBF kernel over multiple linear regression).

4.2. Feature selection

As figure 6 shows, when the number of electrodes used increased from 2 to 10, the results on the training data set were greatly improved (+83% for MLR), whereas the results on the testing data set just slightly improved (+10% for MLR). The use of the feature selection algorithm prevents the over-fitting issue. This results in a decrease in the training set but also in an increase in the testing set. It is particularly true for the SVR with RBF kernel: the modified RBF kernel, which uses feature selection, decreases the results in the training data set by 54% and improves the results in the testing data set by 9% compared to SVR with RBF kernel and no feature selection.

It is also important to note that using the modified kernel produced better results than just using a standard RBF kernel on selected features. This justified ordering the features and giving them different weights in contrast to using uniform weighting of the subset of the features.

Another advantage of feature selection is that when fewer features are used, the training and testing of the SVR is faster. For example, computing the results in figure 6 took about twice as much time when using all the features as when selecting the features and using the modified RBF kernel with some weights w_i set to zero.

4.3. SVR with the modified RBF kernel versus multiple regression

One important difference is that the training of the SVR, especially when using a nonlinear kernel, is 10 to 100 times slower than the training of the multiple linear regression. Fortunately this is not a real issue for the online analysis. Evaluation of the output for each sliding window can be done in real time, and training the SVR takes about a minute and so can be done between sessions.

It is important to note that this study consisted of offline analysis of previously recorded data. The users were trained on and used an online method (McFarland *et al* 2006b) that was very close to multiple regression used here. For this reason, the comparison of the multiple regression and the SVR with RBF kernel might be biased. Moreover, to save some computation

time, some parameters, like the time delay for the use of the cursor positions, were optimized using the multiple regression only. As a result, the benefit of using a SVR with a RBF kernel during an online study may be even more than +12.6% compared to multiple linear regression reported here.

Another way to increase the accuracy of this BCI was explored in this study, namely increasing the number of features by using the electrodes surrounding the two electrodes used in the online study. This augmentation of the number of features potentially created a problem of over-fitting, in particular for the SVR with a RBF kernel, but it was successfully solved by defining a new kernel using a RBF kernel and a feature selection algorithm. The use of this modified RBF kernel with ten electrodes resulted in an average improvement of +24.2% compared to the multiple regression using only two electrodes.

All the results of this study have been obtained with an offline analysis and so will have to be confirmed by online studies. It will also be interesting to investigate the impact of these new techniques on the evolution of the users' performance. It is as yet unknown whether the SVR will decrease the training time necessary to achieve multidimensional cursor control, or whether it will improve the accuracy of the control the user ultimately achieves.

Acknowledgments

This work was supported in part by grants from NIH (HD30146 (NCMRR, NICHD) and EB00856 (NIBIB & NINDS)) and the James S McDonnell Foundation.

References

Albert N B, Weigelt M, Hazeltine E and Ivry R B 2007 Target selection during bimanual reaching to direct cues is unaffected by the perceptual similarity of targets *J. Exp. Psychol.* **33** 1107–16

Birbaumer N, Ghanayim N, Hinterberger T, Iversen I, Kotchoubey B, Kubler A, Perlmouter J, Taub E and Flor H 1999 A spelling device for the paralyzed *Nature* **398** 297–8

Blankertz B, Muller K-R, Krusienski D J, Schalk G, Wolpaw J R, Schlogl A, Pfurtscheller G, Millan J R, Schroder M and Birbaumer N 2006 The BCI competition III: validating alternative approaches to actual BCI problems *IEEE Trans. Neural Syst. Rehabil. Eng.* **14** 153–9

Blum A and Langley P 1997 Selection of relevant features and examples in machine learning *Artif. Intell.* **97** 245–71

Efron B and Tibshirani R 1993 *An Introduction to the Bootstrap* (London: Chapman and Hall)

Farwell L A and Donchin E 1988 Talking off the top of your head: toward a mental prosthesis utilizing event-related brain potentials *Electroencephalogr. Clin. Neurophysiol.* **70** 510–23

Friedrich E V C, McFarland D J, Neuper C, Vaughan T M, Brunner P and Wolpaw J R 2009 A scanning protocol for a sensorimotor rhythm-bases brain–computer interface *Biol. Psychol.* **80** 169–75

Kohavi R 1995 A study of cross-validation and bootstrap for accuracy estimation and model selection *Proc. 14th Int. Joint Conf. on Artificial Intelligence* vol 2 pp 1137–43

Kohavi R and John G H 1995 Wrappers for feature subset selection *Artif. Intell.* **97** 273–324

Lachenbruch P A and Mickey M R 1968 Estimation of error rates in discriminant analysis *Technometrics* **10** 1–11

Marple S L 1987 *Digital Spectral Analysis with Applications* (Englewood Cliffs, NJ: Prentice-Hall)

McFarland D J, Anderson C W, Muller K R, Schlogl A and Krusienski D J 2006a BCI meeting 2005—workshop on BCI signal processing: feature extraction and translation *IEEE Trans. Neural Syst. Rehabil. Eng.* **14** 135–8

McFarland D J, Krusienski D J and Wolpaw J R 2006b Brain–computer interface signal processing at the Wadsworth Center: mu and sensorimotor beta rhythms *Prog. Brain Res* **159** 411–9

McFarland D J, Lefkowicz T and Wolpaw J R 1997a Design and operation of an EEG-based brain–computer interface (BCI) with digital signal processing technology *Behav. Res. Meth. Instrum. Comput.* **29** 337–45

McFarland D J, McCane L M, David S V and Wolpaw J R 1997b Spatial filter selection for EEG-based communication *Electroencephalogr. Clin. Neurophysiol.* **103** 386–94

McFarland D J and Wolpaw J R 2005 Sensorimotor rhythm-based brain–computer interface (BCI): feature selection by regression improves performance *IEEE Trans. Neural Syst. Rehabil. Eng.* **14** 372–9

McFarland D J and Wolpaw J R 2008 Sensorimotor rhythm-based brain–computer interface (BCI): model order selection for autoregressive spectral analysis *J. Neural Eng.* **5** 155–62

Mosier Cl 1951 Symposium: the need and means of cross-validation: I. Problems and designs of cross-validation *Educ. Psychol. Meas.* **11** 5–11

Muller K R, Anderson C W and Birch G E 2003 Linear and non-linear methods for brain–computer interfaces *IEEE Trans. Neural Syst. Rehabil. Eng.* **11** 165–9

Pfurtscheller G, Flotzinger D and Kalcher J 1993 Brain–computer interface—a new communication device for handicapped persons *J. Microcomput. Appl.* **16** 293–9

Schalk G, McFarland D J, Hinterberger T, Birbaumer N and Wolpaw J R 2004 BCI2000: a general-purpose brain–computer interface (BCI) system *IEEE Trans. Biomed. Eng.* **51** 1034–43

Sharbrough F, Chatrian C E, Lesser R P, Luders H, Nuwer M and Picton T W 1991 American electroencephalographic society guidelines for standard electrode position nomenclature *J. Clin. Neurophysiol.* **8** 200–2

Smola A J 1996 Regression estimation with support vector learning machines *Master's thesis* Technische Universität München

Stone M 1974 Cross-validatory choice and assessment of statistical predictions *J. R. Stat. Soc.* B **36** 111–47

Tibshirani R 1996 Regression shrinkage and selection via the Lasso *J. R. Stat. Soc.* B **58** 267–88

Vapnik V 1995 *The Nature of Statistical Learning Theory* (New York: Springer)

Vapnik V 1998a *Statistical Learning Theory* (New York: Wiley)

Vapnik V 1998b The support vector method of function estimation *Nonlinear Modeling: Advanced Black-Box Techniques* ed J A K Suykens and J Vandewalle (Boston, MA: Kluwer)

Wolpaw J R, Birbaumer N, McFarland D J, Pfurtscheller G and Vaughan T M 2002 Brain–computer interfaces for communication and control *Clin. Neurophysiol.* **113** 767–91

Wolpaw J R and McFarland D J 2004 Control of a two-dimensional movement signal by a noninvasive brain–computer interface in humans *Proc. Natl Acad. Sci.* **101** 17849–54

Wolpaw J R, McFarland D J, Neat G W and Forneris C A 1991 An EEG-based brain–computer interface for cursor control *Electroencephalogr. Clin. Neurophysiol.* **78** 252–9