LETTER Communicated by Christoph Guger

A Graphical Model Framework for Decoding in the Visual ERP-Based BCI Speller

S. M. M. Martens
smm.martens@gmail.com
J. M. Mooij
joris.mooij@tuebingen.mpg.de
N. J. Hill
jeremy.hill@tuebingen.mpg.de
Empirical Inference Department, Max Planck Institute for Biological Cybernetics, Tübingen 72076, Germany

J. Farquhar
j.Farquhar@donders.ru.nl
Radboud University Nijmegen Medical Centre, Donders Institute for Brain, Cognition and Behaviour, Nijmegen 6525, Netherlands

B. Schölkopf
bernhard.schoelkopf@tuebingen.mpg.de
Empirical Inference Department, Max Planck Institute for Biological Cybernetics, Tübingen 72076, Germany

We present a graphical model framework for decoding in the visual ERP-based speller system. The proposed framework allows researchers to build generative models from which the decoding rules are obtained in a straightforward manner. We suggest two models for generating brain signals conditioned on the stimulus events. Both models incorporate letter frequency information but assume different dependencies between brain signals and stimulus events. For both models, we derive decoding rules and perform a discriminative training. We show on real visual speller data how decoding performance improves by incorporating letter frequency information and using a more realistic graphical model for the dependencies between the brain signals and the stimulus events. Furthermore, we discuss how the standard approach to decoding can be seen as a special case of the graphical model framework. The letter also gives more insight into the discriminative approach for decoding in the visual speller system.

1 Introduction

The Farwell and Donchin speller (Farwell & Donchin, 1988) is a brain-computer interface that enables users to spell words by focusing their attention on letters in a letter grid displayed on a computer screen. The user's is electroencephalogram (EEG) is recorded while a sequence of controlled stimulus events over time takes place on the letters. A stimulus event in the standard visual speller is a short increase of the brightness ("flash") of a specific group of letters on the screen. The pattern of flashes of the letter that the user is focusing on evokes a characteristic EEG signal that is correlated with the sequence of flashes of that letter over time. A computer program analyzes the recorded EEG signal, inferring the letter on which the user is focusing. This decoding is not a trivial task since the signal-to-noise ratio of the relevant EEG signals is poor.

Increasing the communication rate of the speller can be achieved in two ways: by decreasing the time interval between stimulus events or reducing the number of stimulus events necessary for inferring the user-selected letter. The latter can be achieved by optimizing the design of the sequence of stimulus events. In the standard design, each stimulus event involves the flashing of letters in a particular row or column of the letter grid. Other designs exist that in theory would need fewer stimulus events per communicated letter for a given decoding accuracy than the standard design. We say that these designs have good error-correction capabilities. A new design for the visual speller with good error correction capabilities was studied by Hill, Farquhar, Martens, Biessman, and Schölkopf (2009). Surprisingly, the study revealed that these error-correcting designs in practice perform worse than the standard design in the visual speller. This finding was explained by the fact that the new design increases the number of flashes per communicated letter, leading to a reduction of the signal-to-noise ratio of the EEG due to refractory effects (Martens, Hill, Farquhar, & Schölkopf, 2009).

It seems that the stimulus design in the visual speller involves a trade-off between error-correcting capabilities and the amount of refractory effect. One possible solution to this is to reduce the refractory effects, for example, by using a more salient stimulus type (Hill et al., 2009; Martens et al., 2009). However, it is not clear whether this is an effective solution for all subjects, including patient users with a reduced attention span. Also, if the time interval between subsequent stimulus events were decreased, the refractory effects might become more pronounced again.

In this letter, we evaluate an alternative solution by explicitly taking into account the possibility of refractory effects in the decoding of visual speller data. For this purpose, we introduce the use of graphical models as a framework for the decoding process in the speller system.

We begin in section 2.1 by introducing some terminology in the speller system and discuss standard decoding. We then propose in section 2.1 two graphical models, which represent the generation of brain signals in

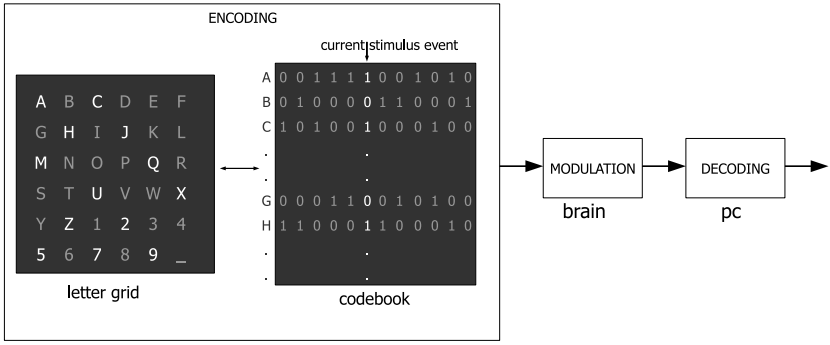

Figure 1: Schematic of the visual speller system showing the encoding, modulation, and decoding units. For the encoding unit, the letter grid is shown during a stimulus event on {A,C,H,J,M,Q,U,X,Z,2,5,7,9}. The corresponding column in the codebook is indicated by the arrow. Since the letters {A,C,G} take part in this stimulus event, their entries for the given stimulus event are 1.

response to the stimulus events, and in section 2.3, we derive the decoding rules based on these graphical models. The first graphical model (which is related to the standard way of decoding visual speller data) does not take into account refractory effects, whereas the second model does. We show how prior knowledge, like letter frequency information, can be easily incorporated in the decoding. We discuss in sections 2.4 and 2.5 subtleties in the training that can be understood more easily in the graphical model framework. We demonstrate in section 2.5 that the commonly used decoding approach may give a maximum a posteriori solution under a number of conditions. Finally, in section 3, we test if an error-correction design outperforms the standard design on real speller data using the proposed decoding.

2 Methods

2.1 Encoding and Decoding in the Visual Speller System. The letter grid in the visual speller may contain numbers, punctuation characters, and other symbols. For simplicity, we will refer to the items in the grid as letters. We distinguish three units in the speller system: encoding, modulation, and decoding (see Figure 1). The encoding describes how each letter is encoded as a sequence of stimulus events over time. In the modulation process, the stimulus events on the user-selected letter are translated into attention-modulated brain signals. The decoding consists of inferring which letter was selected based on the measured brain signals.

While designing a good encoding for the speller, it is helpful to write down these stimulus events per letter as code words. These are bit strings

of length N for which each entry corresponds to a stimulus event. An entry has the value 1 if the letter participates in the stimulus event and value 0 otherwise. The codebook weight refers to the number of 1's in the code word. The Hamming distance between two binary code words of equal length is the number of positions for which the two code words have different values (Hamming, 1950). The collection of code words for all the letters in the letter grid will be referred to as the codebook. Each column in this codebook represents a stimulus event at a given point in time (see Figure 1).

The standard encoding is one in which the stimulus events take place on rows and columns of letters (Farwell & Donchin, 1988). We will refer to this codebook as the row-column codebook (*RC*). The minimum Hamming distance d is the smallest Hamming distance between any two code words in the codebook and is related to how many misclassified code word entries can be corrected. An RC code of length 24 has $d = 4$. In contrast, a Hadamard code *HAD* of length 24 (Levenshtein, 1964) has $d = 12$ and is therefore expected to exhibit superior error-correction properties.

The commonly used decoding in the visual P300 speller consists of feeding a segment of EEG after each stimulus event to a classifier (Kaper, Meinicke, Grossekathoefer, Lingner, & Ritter, 2004; Krusienski et al., 2006). A target event is a stimulus event occurring on the letter that the user selected, and the evoked brain signal is a target response. Similarly, a nontarget event is a stimulus event occurring on other letters, and the evoked brain signal is a nontarget response. This classifier is trained on target and nontarget responses from all bits in a training set and assigns a classifier output value larger than some threshold for a target response and smaller than the threshold for a nontarget response. It is common practice to infer the letter corresponding to the row and column with the largest sum of classifier outputs for all stimulus events (Kaper et al., 2004; Krusienski et al., 2006; Rakotomamonjy & Guigue, 2008; Guger et al., 2009), or, equivalently, the letter for which the inner product of its code word c with the vector of classifier outputs $k = [k_1\, k_2\, \ldots\, k_N]$ is largest—that is, the letter for which the code word satisfies

$$\widehat{c} = \underset{c \in C}{\operatorname{argmax}} \langle c, k \rangle, \tag{2.1}$$

where C denotes the codebook.

2.2 Graphical Models. Graphical models are useful tools to model the (conditional) independencies between random variables. In this letter, we focus on a subset of graphical models called directed acyclic graphs (DAGs). A DAG consists of a set of nodes $\mathcal{V}$ and a set of directed edges $\mathcal{E}$ between the nodes. Each node $i \in \mathcal{V}$ in the graph represents a random variable X_i, and missing edges between the nodes represent conditional independencies. The graph is assumed to be acyclic, that is, there are no directed paths

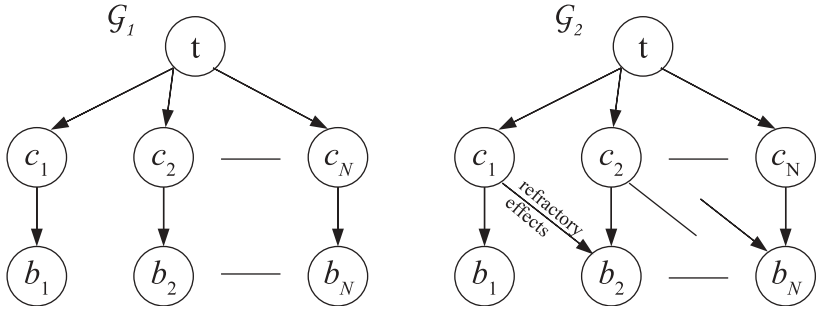

Figure 2: Candidate graphical models $\mathcal{G}_1$ and $\mathcal{G}_2$ for the modulation process in the visual speller system.

$i_1 \to i_2 \to \cdots \to i_k$, where $i_1 = i_k$. Each node i has a set of parent nodes π_i, which can be the empty set.

We denote the joint probability of a set of n random variables $\{X_1, X_2, \ldots, X_n\}$ by $p(x) = p(x_1, x_2, \ldots, x_n) \stackrel{\triangle}{=} P(X_1 = x_1, X_2 = x_2, \ldots, X_n = x_n)$. For DAGs, the joint probability factors into the probabilities $p(x_i \mid \pi_i)$ of the variables conditioned on its parents (Koller & Friedman, 2009):

$$p(x) = \prod_{i \in \mathcal{V}} p(x_i \mid \pi_i). \tag{2.2}$$

For instance, consider graph $\mathcal{G}_1$ depicted in Figure 2. The graph is a DAG with variables t, c_j, and b_j, $j \in \{1, 2, \ldots, N\}$, which are spread over three layers. We can think of this graph as a generative model describing the modulation process in the visual speller system. The variable $t \in \mathcal{T}$ represents a letter from the alphabet $\mathcal{T}$ that consists of all the items in the letter grid. Variable c_j represents an entry from the code word that corresponds to the letter t, which is selected by the user and can take a value from the set $\{0, 1\}$, whereas b_j represents an EEG segment in which we expect to capture the response to a stimulus event c_j. This observed brain signal is a multidimensional and continuous-valued variable.

The directed edges between the letter t and the code word entries c_j represent direct influences. In fact, each letter is associated with a unique bit string such that all code word entries are determined if the letter is fixed, and vice versa. The directed edges between the code word entries c_j and the brain signals b_j also denote direct influences, although their relationship is not deterministic. For example, when a code word entry is set to 0, the corresponding brain response may be of small duration and small amplitude. When a code word entry is 1, the corresponding brain response may be of longer duration and larger amplitude such as a P300 event-related potential response. In practice, b_j also consist of non-task-related

signals such as background EEG signals, measurement noise, and artifacts. The amplitude of the signal that represents the task-related brain response is small compared to the total observed brain signal, which makes the decoding a nontrivial task.

Figure 2 shows an additional graph, $\mathcal{G}_2$, which models the modulation process in a slightly more complex way. Although the variables in $\mathcal{G}_1$ and $\mathcal{G}_2$ are the same, there is a different dependency among the variables. In $\mathcal{G}_2$ there are edges between brain signals b_j at time point j and code word entries at a previous stimulus event c_{j-1}, which are absent in $\mathcal{G}_1$. These slanted edges aim at modeling refractory and overlap effects: Martens et al. (2009) have shown that the shape of the brain response depends on the target- to target interval (TTI) value and therefore on preceding codebook entries. In particular, if a preceding target code word entry c_{j-1} was a 1 (target stimulus event), the brain may not be able to produce the same response if the current code word entry c_j is again a 1. In addition, the brain response to a target event overlaps with the brain response to the subsequent stimulus event. We may extend this dependency further to higher orders by adding more edges, such that b_j depends not only on c_j and c_{j-1} but also on c_{j-2}, and so on.

The graphs convey the following conditional independencies:

$$\mathcal{G}_1 : \; b_j \perp\!\!\!\perp t, c_{j-1}, c_{j-2}, \ldots, b_{j-1}, b_{j-2}, \ldots \mid c_j, \tag{2.3}$$

$$\mathcal{G}_2 : \; b_j \perp\!\!\!\perp t, c_{j-2}, c_{j-3}, \ldots, b_{j-1}, b_{j-2}, \ldots \mid c_{j-1}, c_j. \tag{2.4}$$

In words, this means the following.

- In $\mathcal{G}_1$, if the value of the code word entry c_j at time point j is given, the probability distribution of the observed brain signals b_j at time point j is determined. Moreover, if c_j is given, the probability distribution of b_j does not depend on the letter t, previous code word entries $(c_{j-1}, \ldots, c_1)$, or previous brain signals $(b_{j-1}, \ldots, b_j)$.
- In $\mathcal{G}_2$, if the value of the code word entry c_j at time point j is given, the probability of the observed brain signals b_j at time point j is still uncertain since the probability distribution of b_j also depends on c_{j-1}. However, if c_{j-1} and c_j are given, the probability distribution of b_j does not depend on t, earlier code word entries $(c_{j-2}, \ldots, c_1)$, or earlier brain signals $(b_{j-1}, \ldots, b_1)$.

Consequently, we can express the joint probability as a factorization of the variables conditioned on its parents:

$$\mathcal{G}_1 : \; p(t, c, b) = p(t) \prod_{j=1}^{N} p(c_j \mid t) p(b_j \mid c_j), \tag{2.5}$$

$$\mathcal{G}_2 : \; p(t, c, b) = p(t) \prod_{j=1}^{N} p(c_j \mid t) p(b_j \mid c_{j-1}, c_j), \tag{2.6}$$

where we set the fictional variable c_0 in equation 2.6 to 0 with probability 1. Later we will be interested in the joint probability $p(c, b)$ of the code word and the brain signals given by

$$\mathcal{G}_1 : \; p(c, b) = p(c) \prod_{j=1}^{N} p(b_j \mid c_j), \tag{2.7}$$

$$\mathcal{G}_2 : \; p(c, b) = p(c) \prod_{j=1}^{N} p(b_j \mid c_{j-1}, c_j), \tag{2.8}$$

with $p(c) = \sum_t p(t) \prod_j p(c_j \mid t)$ expressing how the letter prior $p(t)$ induces a code word prior $p(c)$. Note that the prior $p(c)$ of a code word c is equal to the letter prior $p(t)$ of the letter corresponding to that code word (and vanishes for nonvalid code words $c \notin C$).

2.3 Decoding Based on Graphical Models. One of the uses of graphical models is to do inference. For this purpose, we want to compute the posterior probability of one or more variables conditioned on some other variables (Koller & Friedman, 2009). Maximum a posteriori decoding in the context of the visual speller means that we infer the communicated letter by selecting the letter with the largest posterior probability given the measured brain signals:

$$\widehat{t} = \operatorname*{argmax}_{t \in \mathcal{T}} p(t \mid b). \tag{2.9}$$

By identifying letters in $\mathcal{T}$ with their corresponding code words in C, we may equivalently select the code word with the largest probability given the measured brain signals:

$$\widehat{c} = \operatorname*{argmax}_{c \in C} p(c \mid b). \tag{2.10}$$

This is equivalent to selecting the code word with the largest joint probability since $p(c \mid b) = p(c, b)/p(b)$ and $p(b)$ is independent of the code word:

$$\widehat{c} = \operatorname*{argmax}_{c \in C} p(c, b). \tag{2.11}$$

The joint probability $p(c, b)$ of the code word and the brain signals was defined previously for $\mathcal{G}_1$ and $\mathcal{G}_2$ in equations 2.7 and 2.8, respectively. Therefore, to perform the decoding according to equation 2.11, we need to find the distribution of brain signals given the current, and possibly preceding, code word entries. This generative approach has been successfully adopted for $\mathcal{G}_1$ in Martens and Leiva (2010). Another approach is to turn around the conditional probabilities in the joint in equations 2.7 and 2.8

by applying Bayes' rule: $p(x \mid y) = p(y \mid x)p(x)/p(y)$. The resulting expressions for the joint probability can be inserted in equation 2.11 to obtain the decoding rules for $\mathcal{G}_1$ and $\mathcal{G}_2$:

$$\mathcal{G}_1 : \widehat{c} = \underset{c \in C}{\operatorname{argmax}}\, p(c) \prod_{j=1}^{N} \frac{p(c_j \mid b_j)}{p(c_j)}, \tag{2.12}$$

$$\mathcal{G}_2 : \widehat{c} = \underset{c \in C}{\operatorname{argmax}}\, p(c) \prod_{j=1}^{N} \frac{p(c_{j-1}, c_j \mid b_j)}{p(c_{j-1}, c_j)}, \tag{2.13}$$

where the factor $p(b_j)$ is independent of the code word and is therefore discarded. Notice that by learning the conditional probabilities $p(c_j \mid b_j)$ and $p(c_{j-1}, c_j \mid b_j)$ in equations 2.12 and 2.13 from data, we perform a discriminative training for a generative model.

2.4 Homogeneity Assumptions. We will assume homogeneity such that in $\mathcal{G}_1$, the conditional distribution $p(b_j = \beta \mid c_j = \gamma)$ in equation 2.7 does not depend on the bit j (for fixed β and γ). This means that the brain signal b_j generated by a stimulus event defined by c_j for bit j cannot be distinguished from the brain signal generated by another stimulus event defined by c_i at bit i, if the two stimulus events have the same value $c_j = c_i = \gamma$. Similarly, given $\mathcal{G}_2$, we assume that $p(b_j = \beta \mid c_j = \gamma, c_{j-1} = \gamma')$ in equation 2.8 does not depend on the bit j for a fixed β, γ, and γ'.

It is important to note that the homogeneity assumption in $\mathcal{G}_1$ implies a bit independence for the probability distribution $p(b_j \mid c_j)$ but not necessarily for the conditional probability $p(c_j \mid b_j)$. Indeed, by using Bayes' rule on the homogeneity assumption $p(b_j = \beta \mid c_j = \gamma) = p(b_i = \beta \mid c_i = \gamma)$, it follows that the equation $p(c_j = \gamma \mid b_j = \beta) = p(c_i = \gamma \mid b_i = \beta)$ holds only if $p(c_j = \gamma) = p(c_i = \gamma)$. These homogeneity assumptions are relevant for the training phase, as will be explained in the next section.

2.5 Bit Dependencies. The per bit conditional probability factor $p(c_j = \gamma \mid b_j = \beta)$ in equation 2.12 may be estimated for each bit j individually using the training examples corresponding to that code word entry c_j. However, we may want to accumulate the training examples of all bits $j \in \{1, \ldots, N\}$ and estimate a conditional probability $f(\gamma \mid \beta)$ on the complete training set aggregated over all bits j in favor of a more accurate estimation. Unfortunately, from section 2.4 we know that $p(c_j = \gamma \mid b_j = \beta)$ is bit dependent, and therefore $f(\gamma \mid \beta)$ is in general not equal to $p(c_j = \gamma \mid b_j = \beta)$. Consequently, we may not simply substitute the per bit conditional probability $p(c_j \mid b_j)$ by the global conditional probability $f(c_j \mid b_j)$ in the decoding rule of equation 2.12.

Fortunately, the homogeneity assumption offers a solution such that we can use the learned global conditional probability $f(\gamma \mid \beta)$ for decoding. The global bit probability $f(\gamma)$ and the global joint probability $f(\beta, \gamma)$ are by definition:

$$f(\gamma) = \frac{1}{N} \sum_{j=1}^{N} p(c_j = \gamma), \tag{2.14}$$

$$f(\beta, \gamma) = \frac{1}{N} \sum_{j=1}^{N} p(b_j = \beta, c_j = \gamma). \tag{2.15}$$

From this, it follows that $f(\beta \mid \gamma) = f(\beta, \gamma)/f(\gamma)$ can be expressed as

$$f(\beta \mid \gamma) = \frac{\frac{1}{N} \sum_{j=1}^{N} p(b_j = \beta, c_j = \gamma)}{\frac{1}{N} \sum_{j=1}^{N} p(c_j = \gamma)}, \tag{2.16}$$

$$= \frac{\frac{1}{N} \sum_{j=1}^{N} p(b_j = \beta \mid c_j = \gamma) p(c_j = \gamma)}{\frac{1}{N} \sum_{j=1}^{N} p(c_j = \gamma)}, \tag{2.17}$$

$$= \frac{p(b_j = \beta \mid c_j = \gamma) \frac{1}{N} \sum_{j=1}^{N} p(c_j = \gamma)}{\frac{1}{N} \sum_{j=1}^{N} p(c_j = \gamma)}, \tag{2.18}$$

$$= p(b_j = \beta \mid c_j = \gamma). \tag{2.19}$$

Due to the homogeneity assumption, the global probability distribution $f(\beta \mid \gamma)$ is equal to the per bit probability distributions $p(b_j = \beta \mid c_j = \gamma)$. Similarly, equality holds between $f(\beta \mid \gamma, \gamma')$ and $p(b_j = \beta \mid c_j = \gamma, c_{j-1} = \gamma')$ under $\mathcal{G}_2$. We may therefore substitute $p(b_j \mid c_j)$ and $p(b_j \mid c_{j-1}, c_j)$ in equations 2.7 and 2.8 by $f(b_j \mid c_j)$ and $f(b_j \mid c_{j-1}, c_j)$, respectively, and apply Bayes' rule to find the following decoding rules:

$$\mathcal{G}_1 : \widehat{c} = \underset{c \in C}{\operatorname{argmax}}\, p(c) \prod_{j=1}^{N} \frac{f(c_j \mid b_j)}{f(c_j)}, \tag{2.20}$$

$$\mathcal{G}_2 : \widehat{c} = \underset{c \in C}{\operatorname{argmax}}\, p(c) \prod_{j=1}^{N} \frac{f(c_{j-1}, c_j \mid b_j)}{f(c_{j-1}, c_j)}, \tag{2.21}$$

where $f(b_j) = 1/N \sum_{j=1}^{N} p(b_j)$ is independent of the code word and has been discarded.

Notice the similarity of these to equations 2.12 and 2.13. It turns out that we can use the conditional probability $f(c_j \mid b_j)$ estimated from all the bits in the training set if we divide by the global bias $f(c_j)$ instead of $p(c_j)$ for

each bit j. This $f(c_j)$ depends on only the value of c_j and is therefore bit independent in contrast to $p(c_j)$. From now on, we refer to the factors $f(c_j)$ in equation 2.20 and $f(c_{j-1}, c_j)$ in 2.21 as global bias correction factors. They correct for the presence of bit dependencies of $p(c_j \mid b_j)$ and $p(c_{j-1}, c_j \mid b_j)$. A more rigorous derivation of the global bias correction factors can be found in appendix C.

There are two special cases for which the product of the global bias correction factors $\prod_{j=1}^{N} f(c_j)$ in equation 2.20 is constant for all code words and consequently negligible in the decoding: (1) all code words have the same weight, and (2) each value of c_j is equally probable in the training set. However, this is not necessarily true for the product of the global bias correction factors $\prod_{j=1}^{N} f(c_{j-1}, c_j)$ in equation 2.21.

Equation 2.20 also shows that the practice of balancing the number of training examples for the two classes (as, e.g., in Kaper et al., 2004; Martens et al., 2009) yields $f(c_j) = 0.5$. In that case, the global bias correction factor becomes code word independent and can be neglected. But if the balancing is done by throwing away examples of the abundant class, the resulting reduction of the training set will lead to a less accurate estimation of the conditional probability.

The standard decoding method for visual speller data as defined in equation 2.1 arises as a special case of $\mathcal{G}_1$ under the following three additional assumptions: the classifier outputs can be transformed into probabilistic quantities according to a logistic function, all letters in the letter grid are equally likely, and all code words have the same weight (see appendix A). If one uses a classifier that gives nonprobabilistic outputs, it is unclear how to incorporate factors such as $f(c_j)$, $f(c_{j-1}, c_j)$, and letter priors $p(c)$ in the decoding.

2.6 Training by Regularized Logistic Regression. We may learn the conditional probabilities $f(c_j \mid b_j)$ in equation 2.20 and $f(c_{j-1}, c_j \mid b_j)$ in equation 2.21 by a logistic regression. A logistic regression models the posterior probabilities of the classes by a generalized linear model while at the same time ensuring that the probabilities sum to 1 and remain in $[0, 1]$ (Hastie, Tibshirani, & Friedman, 2001). The models are as follows:

$$f(c_j \mid b_j) = \frac{\exp\left(w_{1,c_j}^T b_j + \eta_{1,c_j}\right)}{\sum_{c_j} \exp\left(w_{1,c_j}^T b_j + \eta_{1,c_j}\right)}, \tag{2.22}$$

$$f(c_{j-1}, c_j \mid b_j) = \frac{\exp\left(w_{2,c_{j-1},c_j}^T b_j + \eta_{2,c_{j-1},c_j}\right)}{\sum_{c_{j-1},c_j} \exp\left(w_{2,c_{j-1},c_j}^T b_j + \eta_{2,c_{j-1},c_j}\right)}. \tag{2.23}$$

The parameters w_{1,c_j} and η_{1,c_j} in the binary classification problem 2.22 and w_{2,c_{j-1},c_j} and η_{2,c_{j-1},c_j} in the multiclass classification problem 2.23 can be learned by maximum likelihood. A regularization term is added to the log

likelihood to reduce the chance of overfitting (see appendix B for a more detailed description).

2.7 Letter Prior. Suppose we have trained on a data set with a given letter prior $p(t)$, whereas the letters in the test set come from a different distribution $p'(t)$. This may happen if we let the subject do a copy-spelling task for training with randomly drawn letters and then let the subject communicate proper sentences. Since we want to optimize the letter decoding performance in the test set, we should simply replace $p(c)$ in equations 2.20 and 2.21 by the code word prior $p'(c)$ induced by the letter prior of the test set $p'(t)$.

3 Real Visual Speller Data

3.1 Setup. Eleven subjects performed a copy-spelling task with the visual speller system implemented in the BCPy2000 platform (http://www.bci2000.org/wiki/index.php/Contributions:BCPy2000). The subject observed a PC screen on which a 6×6 letter grid was displayed (as in Figure 1). The task was to focus attention on a specified letter of the grid and passively count the number of times a stimulus event occurred on that letter. All subjects used the system with a standard letter intensification type of stimulus. The time interval between the start of one stimulus event and the start of the next event, the stimulus onset asynchrony (SOA), was set to 183 ms. Each intensification lasted 100 ms and was followed by a no-intensification period of 83 ms. We recorded a 16-channel common-average-reference EEG sampled at 500 Hz using a QuickAmp system (BrainProducts GmbH). Each subject spelled sentences from the book *The Diving Bell and the Butterfly* by Bauby (1998) until the subject indicated that he or she was tired, resulting in 73 to 113 trials (letters spelled) per subject. Feedback was given to the subjects after their spelling session. Two different codebooks of length $N = 72$ were used: a standard row-column codebook (RC) and a Hadamard codebook (HAD; see Figure 3). For the RC codebook, each stimulus event occurred on 6 letters in the same row or column, whereas for the HAD codebook, each stimulus event involved 13 to 20 letters spread over the grid.

The codebooks alternated per communicated letter. The HAD codebook was created by selecting 36 code words of a Hadamard code of length 24, permuting the columns to increase randomness of target events, concatenating code words three times, and assigning each resulting code word of length 72 to a letter in the grid.[1] The RC has a small minimum Hamming distance of 12, and the HAD has a large minimum Hamming distance of 36. The weight of the code words is 12 for the RC code and between 33 and 39

[1]For Hadamard codes, $d = N/2$, such that a Hadamard code of length $N = 72$ bits would not have yielded a larger d than the proposed concatenated code.

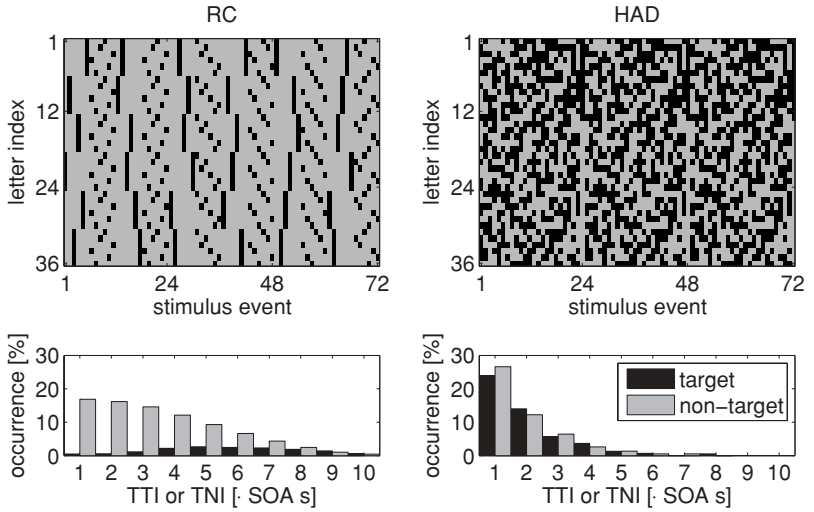

Figure 3: The upper plots show examples of code books depicted as gray (0 in code word) and black (1 in code word) pixels for the RC and the HAD codebooks. In this RC codebook, the stimulus events on the rows and columns are not mixed up; first all the rows take turns, then all the columns, and so on. Both code books have 36 code words at length $N = 72$. The lower plots show the percentage of stimulus events at a particular target-to-target (TTI) or target-to-nontarget interval (TNI) value for both codebooks.

for the HAD codebook. The large percentage of 1's in the HAD codebook leads to small TTI values, whereas the small percentage of 1's in the RC codebook results in a widespread distribution of TTI values (see Figure 3). We expect that the error-correcting capabilities of the HAD code book are diminished by strong refractory effects due to the large number of small TTI targets. Nevertheless, by applying the decoding method based on $\mathcal{G}_2$, which models these refractory effects, the HAD codebook may outperform the RC code.

3.2 Signal Analysis. The signal analysis was performed offline in Matlab. The EEG was bandpass-filtered between 0.5 and 10 Hz with steep FIR Bartlett-Hanning filters, and cut up in 600 ms EEG epochs synchronized by the stimulus cues. These epochs were downsampled to 25 Hz. We trained and decoded on the complete 72 bits code words. We performed the training on $L = \{5, 10, 20, 40\}$ letters and tested the decoding performance on the remaining letters. We applied an l-fold cross-validation on the training set with the percentage of correctly inferred letters as a criterion to select the optimal regularization parameter, using $l = 10$ folds if the number of

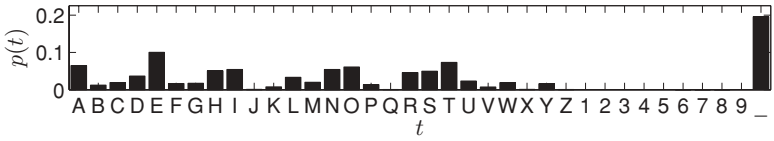

Figure 4: Probability of characters in English. The SPACE character (represented by '_') has the largest probability, about 20%.

training letters L was larger than 10, and $l = L$ folds otherwise. After the cross-validation, a logistic regression was trained on the complete training set using the selected regularization parameter.

Decoding was done on the test set according to equations 2.20 and 2.21, where the learned logistic regression parameters were applied to the data in the test set according to equations 2.22 and 2.23. We set the letter priors based on English character frequencies in about 1415 works of fiction (http://millikeys.sourceforge.net/freqanalysis.html; see Figure 4). We calculated the decoding performance as the percentage of correctly inferred letters in the test set.

4 Results

4.1 Effect of Global Bias Correction. We investigated the impact of the global bias correction on the decoding performance (see section 2.5 for theoretical background). For this purpose, one subject used a different Hadamard code, which we will refer to as HADspecial. This codebook consists of just two code words with weights 39 and 3, respectively. The resulting global bias corrections take on completely different values for the two code words (see Figure 5). This particular data set contained 27 trials in which one of these two code words was communicated by the subject. To increase the test set size, we split each communicated code word up into three code words of length $N = 24$, as if each code word had been communicated three times.

We recalculated the prior probabilities of the two code words after setting the probabilities of the other letters in Figure 4 to 0 giving $p(E) = 0.34$ and $p(\_) = 0.66$. We performed a decoding according to $\mathcal{G}_1$ as in equation 2.20. In addition, we performed a naive decoding $\mathcal{G}_{1,\text{no correction}}$, which uses a logistic regression trained on all bits but ignores the global bias correction factor in the joint, that is, according to $p(c) \prod_j f(c_j \mid b_j)$, and another naive decoding $\mathcal{G}_{1,\text{wrong correction}}$, which uses a logistic regression trained on all bits but uses the wrong global bias correction factor, according to $p(c) \prod_j f(c_j \mid b_j)/p(c_j)$.

The decoding performance of $\mathcal{G}_{1,\text{no correction}}$ and $\mathcal{G}_{1,\text{wrong correction}}$ was lower than the performance of $\mathcal{G}_1$, which used the correct global bias correction factor (see Figure 5). The difference in performance between $\mathcal{G}_1$ and $\mathcal{G}_{1,\text{no correction}}$ was significant at the 5% level (40 training letters, one-tailed

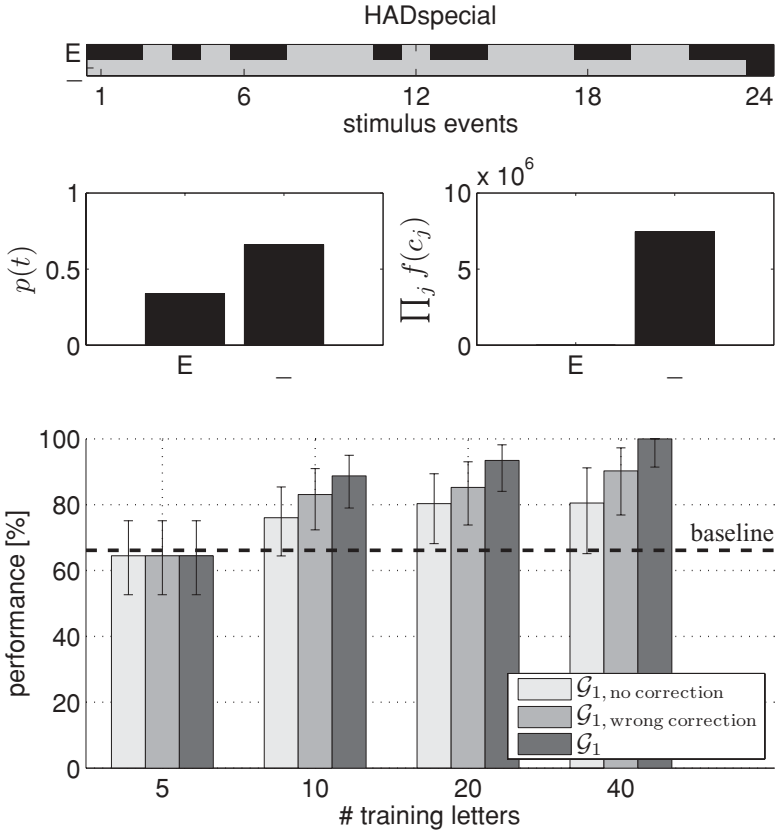

Figure 5: Effect of the global bias correction on the decoding performance in $\mathcal{G}_1$ for the HADspecial codebook (upper plot). Priors of the letters $p(t)$ as well as the product of the global bias correction factors $\prod_{j=1}^{N} f(c_j)$ (normalized by the correction value for E) are shown in the middle plots (global bias correction factors $f(0) = 0.8$ and $f(1) = 0.2$). The lower plot shows the performance of the naive decodings $\mathcal{G}_{1,\text{no correction}}$ and $\mathcal{G}_{1,\text{wrong correction}}$, and the correct decoding according to $\mathcal{G}_1$. Error bars denote 95% confidence intervals. The baseline shows the expected accuracy by always selecting the letter with the largest prior probability. (For a color version of this figure see the supplemental material, available online at http://www.mitpressjournals.org/doi/suppl/10.1162/NECO_a_00066.)

Fisher's exact test, $p = 0.003$), whereas the decoding difference between $\mathcal{G}_1$ and $\mathcal{G}_{1,\text{wrong correction}}$ was marginally significant ($p = 0.06$).

4.2 Effect of Letter Frequency Information. We investigated the increase in decoding performance if letter frequency information is used. For

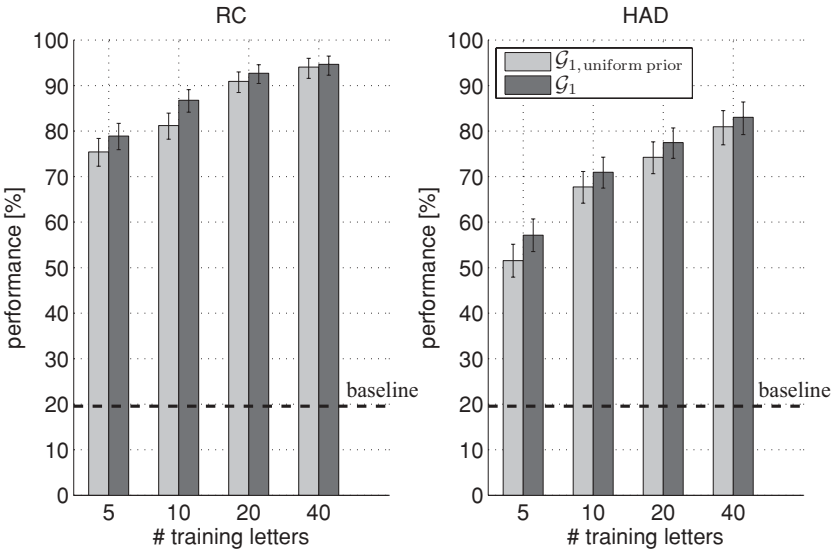

Figure 6: Decoding performance of $\mathcal{G}_1$ on RC and HAD data using a uniform prior (light bars) and a realistic letter prior (dark bars). The bars are the average decoding performances over nine subjects; error bars denote 95% confidence intervals. The baseline shows the expected performance by always selecting the letter with the largest prior probability in Figure 4. (For a color version of this figure see the supplemental material, available online at http://www.mitpressjournals.org/doi/suppl/10.1162/NECO_a_00066.)

this purpose, we analyzed the visual speller data from the 10 subjects who used the RC and HAD codebooks. One subject did not reach above-chance performance and was left out of the analysis. We used equation 2.20 with a realistic letter prior as in Figure 4 and also with a uniform letter prior, referred to as $\mathcal{G}_{1,\text{uniform prior}}$.

Using realistic prior knowledge about the probability of the letters increased the decoding performance (see Figure 6) up to 5%. The difference in performance between $\mathcal{G}_1$ and $\mathcal{G}_{1,\text{uniform prior}}$ was significant for the HAD data (5 training letters, Pearson's chi square test, $p = 0.03$) but not for the RC data ($p = 0.14$).

4.3 $\mathcal{G}_1$ Versus $\mathcal{G}_2$. The two decoding methods from equations 2.20 and 2.21 were tested on visual speller data from the 10 subjects who used the RC and HAD codebooks. One subject did not reach above-chance performance and was left out of the analysis. For large training set sizes, graph $\mathcal{G}_2$ showed on average the same decoding performance as graph $\mathcal{G}_1$ on the RC data (see Figure 7), whereas $\mathcal{G}_2$ performed significantly better than $\mathcal{G}_1$ on the

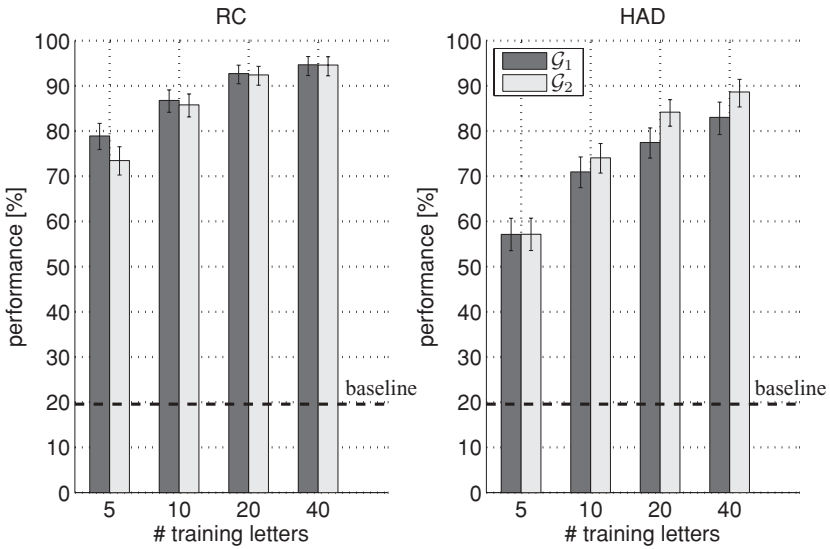

Figure 7: Decoding performance of graph $\mathcal{G}_1$ (dark bars) versus $\mathcal{G}_2$ (light bars) on RC and HAD data. The bars are the average letter accuracy performance over nine subjects; error bars denote 95% confidence intervals. The baseline shows the expected accuracy by always selecting the letter with the largest realistic letter prior probability in Figure 4. (For a color version of this figure see the supplimental material, available online at http://www.mitpressjournals.org/doi/suppl/10.1162/NECO_a_00066.)

HAD data (40 training letters, Pearson's chi square test, $p = 0.01$). For small training set sizes, $\mathcal{G}_1$ performed better than $\mathcal{G}_2$ on the RC data ($p = 0.03$) and equally well on the HAD data.

4.4 Effect of Codebook. The decoding performance of the RC codebook was superior over the HAD codebook, independent of the number of training trials (see Figure 7). Using $\mathcal{G}_2$ for decoding instead of $\mathcal{G}_1$ improved the performance of the HAD code, but not so much that it outperformed the RC code.

5 Conclusion

The aim of this letter is to promote a flexible framework using graphical models for maximum a posteriori decoding in the speller. The framework can be seen as an upper level in the decoding process in which the researcher picks or designs a realistic graphical model for the generation of brain signals in response to stimulus events. We proposed two graphical models,

$\mathcal{G}_1$ and $\mathcal{G}_2$, each with different dependencies between the variables. We have shown that the commonly used decoding approach can be seen as a special case of the simple graphical model $\mathcal{G}_1$.

The lower level involves the training or learning on the selected graphical model. We showed how to do this training discriminatively, and this principle has been successfully applied in speech recognition (Kapadia, 1998) and natural language processing (Collins, 2002). Although we applied a regularized logistic regression classifier to perform the learning, one has the freedom to use his or her favorite classifier as long as it gives quantities that can be interpreted probabilistically. For example, a support vector machine classifier, whose outputs were squeezed through a logistic function, resulted in a similar decoding performance as the logistic regression we used.

The homogeneity assumption for the generation of the brain signals allows us to perform the learning on the complete training data set instead of bit-wise. This is common practice in the literature. One should, however, be cautious if the conditional probability of a code word entry being 0 or 1 is bit dependent. Therefore, during decoding, a global bias correction should be included that corrects for the global bias in the training data set. The necessity of the global bias correction directly follows from the homogeneity assumption. We showed that the global bias correction is crucial if a codebook is used in which the code words have different weights and the global probability of a bit being 1 is far from 0.5.

In both graphical models, letter frequency information is incorporated by code word priors. The results demonstrate that adding letter frequency information improves the decoding performance. A next step would be to use letter priors conditioned on previously communicated letters.

Graph $\mathcal{G}_2$ models dependencies between brain signals and previous stimulus events and therefore recognizes the presence of refractory effects of the brain signals. The training and decoding involves the classification of pairs of bits, a four–class problem. The results show that this graphical model yields a better decoding performance on data sets in which the code words are characterized by a large weight. For small training set sizes, however, $\mathcal{G}_2$ suffers from a slower learning curve and performs worse than $\mathcal{G}_1$. This is to be expected since $\mathcal{G}_2$ encodes a more general class of models than $\mathcal{G}_1$. Therefore, there is a trade-off between model generality and learning speed of the training procedure with respect to the number of data points.

We tested two codebooks: the standard row-column (RC) code and a Hadamard (HAD) code. If the per bit classification accuracy were independent of the codebook, the HAD codebook would be superior to the RC codebook. However, refractory effects lead to lower per bit classification accuracies in codebooks with a large weight such as the HAD codebook. In our experiment, we used the HAD codebook and tried to make the decoding suffer less from refractory effects by using the more sophisticated graphical model $\mathcal{G}_2$. The effort we made in modeling the refractory effects

by $\mathcal{G}_2$ improved the performance of the HAD data significantly, but not so much that the HAD outperformed the RC codebook. Our explanation for this finding is that $\mathcal{G}_2$ cannot simply make up for the reduction in binary classification performance: it merely expresses uncertainty for the bits with refractory effects, whereas $\mathcal{G}_1$ misclassifies these bits. The more realistic prior knowledge in the form of the English letter prior in Figure 4 is apparently not strong enough to exploit this difference.

Future work consists of testing the HAD codebook with a more salient stimulus type as in Martens et al. (2009) for which the refractory effects are reduced. In this setting, the HAD codebook is expected to outperform the RC codebook. A further increase in bit rate could then be achieved by speeding up the presentation of stimulus events. At faster stimulus rates, refractory effects are likely to occur even if salient stimulus types are used, and the system would profit from a decoding that models these effects in combination with stronger prior knowledge in the form of a letter prior conditioned on previously communicated letters.

Appendix A: Relationship Between Standard Decoding and MAP Decoding

Selecting the code word that maximizes $\langle c, k \rangle$ as in equation 2.1 is equivalent to a MAP solution as in equation 2.20 under the following three conditions:

1. The classifier outputs k_j can be transformed into probabilistic quantities according to a logistic function. If $f(c_j = 1 \mid b_j) \propto \frac{\exp(k_j)}{\exp(k_j)+\exp(-k_j)}$, then $c_j k_j \propto c_j \log(\frac{f(c_j=1 \mid b_j)}{1-f(c_j=1 \mid b_j)})$.
   We rewrite equation 2.1 as

$$\widehat{c} = \operatorname*{argmax}_{c \in C} \sum_j \log(1 - f(c_j = 1 \mid b_j)) + \sum_j c_j \log\left(\frac{f(c_j = 1 \mid b_j)}{1 - f(c_j = 1 \mid b_j)}\right), \tag{A.1}$$

   where the first term $\log(1 - f(c_j = 1 \mid b_j))$ may be added since it is independent of c_j. Separating the cases $c_j = 0$ and $c_j = 1$ gives

$$\widehat{c} = \operatorname*{argmax}_{c \in C} \sum_j \begin{cases} \log(1 - f(c_j = 1 \mid b_j)) & \text{if } c_j = 0 \\ \log(f(c_j = 1 \mid b_j)) & \text{if } c_j = 1 \end{cases}. \tag{A.2}$$

   From this we see that

$$\widehat{c} = \operatorname*{argmax}_{c \in C} \sum_j \log f(c_j \mid b_j), \tag{A.3}$$

$$= \operatorname*{argmax}_{c \in C} \prod_j f(c_j \mid b_j). \tag{A.4}$$

2. All letters in the letter grid are equally likely (i.e., the code word prior is uniform). If the marginal probability of the code words $p(c)$ is constant, this factor can be ignored in equation 2.20.
3. All code words have the same weight. In that case, the global bias factor $f(c_j)$ in equation 2.20 can be ignored, and equation 2.27 is equivalent to equation 2.20.

Appendix B: Fitting the Logistic Regression Parameters

Suppose we are given a training set of i.i.d. samples $\mathcal{D} = \{(c^{(m)}, b^{(m)})\}_{m=1}^{M}$ drawn from a training distribution. The likelihood of the parameters in $\mathcal{G}_1$ is given by

$$\mathcal{G}_1 : \; f\left(\mathcal{D} \mid w_{1,c_j}, \eta_{1,c_j}\right) = \prod_{m=1}^{M} f\left(c^{(m)}, b^{(m)} | w_{1,c_j}, \eta_{1,c_j}\right), \tag{B.1}$$

$$= \prod_{m=1}^{M} \frac{\exp\left(w_{1,c_j^{(m)}}^T b^{(m)} + \eta_{1,c_j^{(m)}}\right)}{\sum_{c_j} \exp\left(w_{1,c_j}^T b^{(m)} + \eta_{1,c_j}\right)} \prod_{m=1}^{M} p(b^{(m)}). \tag{B.2}$$

Instead of maximizing the likelihood, we minimize the following loss function $L(w_{1,c_j}, \eta_{1,c_j})$, which includes a regularization term,

$$\mathcal{G}_1 : \; L\left(w_{1,c_j}, \eta_{1,c_j}\right) = -\sum_{m=1}^{M} \log\left(\frac{\exp\left(w_{1,c_j^{(m)}}^T b^{(m)} + \eta_{1,c_j^{(m)}}\right)}{\sum_{c_j} \exp\left(w_{1,c_j}^T b^{(m)} + \eta_{1,c_j}\right)}\right)$$

$$+ R \sum_{c_j} \| w_{1,c_j} \|^2, \tag{B.3}$$

with R the regularization parameter. Notice that the factor $\prod_{m=1}^{M} p(b^{(m)})$ in the likelihood in equation B.2 may be neglected since the factor does not depend on the parameters w_{1,c_j} and η_{1,c_j}. Alternatively, one can derive equation B.3 as the MAP estimate of the parameters w_{1,c_j} and η_{1,c_j} if we assume a gaussian prior over the weights w_{1,c_j}. To minimize the loss function, we set its derivative with respect to w_{1,c_j} and η_{1,c_j} to zero. The resulting equations can be solved using iteratively reweighted least squares (Hastie et al., 2001). The derivations for $\mathcal{G}_2$ are calculated likewise.

Appendix C: Alternative Way of Training the Logistic Regression Classifier

In this appendix, we show that the bit dependency of $p(c_j \mid b_j)$ can be dealt with in two different ways and that these two approaches are asymptotically equivalent. The derivation is presented only for $\mathcal{G}_1$, since for $\mathcal{G}_2$, it is similar.

First, let $\phi : \mathcal{B} \to \mathbb{R}^d$ be some feature function, mapping brain signals into d-dimensional real vectors. We assume that for each bit $j = 1, \ldots, N$, the probability of the class c_j given the brain signal b_j is of the logistic regression form

$$p(c_j = \gamma \mid b_j = \beta) = \frac{\exp\left(w_{\gamma,j}^T \phi(\beta) + \eta_{\gamma,j}\right)}{\sum_\Gamma \exp\left(w_{\Gamma,j}^T \phi(\beta) + \eta_{\Gamma,j}\right)}, \tag{C.1}$$

where $w_{\gamma,j}$ and $\eta_{\gamma,j}$ are a d-dimensional weight vector and a bias parameter that both depend on γ and on j. The conditional probability of the brain signal given the class is therefore

$$\begin{aligned} p(b_j = \beta \mid c_j = \gamma) &= \frac{\exp\left(w_{\gamma,j}^T \phi(\beta) + \eta_{\gamma,j}\right)}{\sum_\Gamma \exp\left(w_{\Gamma,j}^T \phi(\beta) + \eta_{\Gamma,j}\right)} \frac{p(b_j = \beta)}{p(c_j = \gamma)}, \\ &= \psi_j(\beta) \exp\left(w_{\gamma,j}^T \phi(\beta) + \eta_{\gamma,j} - \rho_{\gamma,j}\right), \end{aligned} \tag{C.2}$$

where we defined $\rho_{\gamma,j} := \log p(c_j = \gamma)$ and separated off the following factor,

$$\psi_j(\beta) := \frac{p(b_j = \beta)}{\sum_\Gamma \exp\left(w_{\Gamma,j}^T \phi(\beta) + \eta_{\Gamma,j}\right)}, \tag{C.3}$$

which does not depend on γ. This will turn out to be convenient later.

Now we employ the homogeneity assumption, which states that the conditional probability in equation C.2 is independent of j. In other words, for all $j, i = 1, \ldots, N$, we have

$$\begin{aligned} \forall_\beta \forall_\gamma : \psi_j(\beta) &\exp\left(w_{\gamma,j}^T \phi(\beta) + \eta_{\gamma,j} - \rho_{\gamma,j}\right) \\ &= \psi_i(\beta) \exp\left(w_{\gamma,i}^T \phi(\beta) + \eta_{\gamma,i} - \rho_{\gamma,i}\right). \end{aligned} \tag{C.4}$$

Since this holds for any γ, we can sum the equations over γ, which gives for $j, i = 1, \ldots, N$:

$$\begin{aligned} \forall_\beta : \quad \psi_j(\beta) &\sum_\Gamma \exp\left(w_{\Gamma,j}^T \phi(\beta) + \eta_{\Gamma,j} - \rho_{\Gamma,j}\right) \\ &= \psi_i(\beta) \sum_\Gamma \exp\left(w_{\Gamma,i}^T \phi(\beta) + \eta_{\Gamma,i} - \rho_{\Gamma,i}\right). \end{aligned} \tag{C.5}$$

Now, forming the quotient of equations C.4 and C.5, the factors $\psi_j(\beta)$ and $\psi_i(\beta)$ drop out and we obtain

$$\forall_\beta \forall_\gamma : \quad \frac{\exp\left(w_{\gamma,j}^T \phi(\beta) + \eta_{\gamma,j} - \rho_{\gamma,j}\right)}{\sum_\Gamma \exp\left(w_{\Gamma,j}^T \phi(\beta) + \eta_{\Gamma,j} - \rho_{\Gamma,j}\right)} = \frac{\exp\left(w_{\gamma,i}^T \phi(\beta) + \eta_{\gamma,i} - \rho_{\gamma,i}\right)}{\sum_\Gamma \exp\left(w_{\Gamma,i}^T \phi(\beta) + \eta_{\Gamma,i} - \rho_{\Gamma,i}\right)}, \tag{C.6}$$

for all $j, i = 1, \ldots, N$.

A solution to these equations is obtained by taking all N weight vectors to be identical

$$w_{\gamma,j} = w_\gamma \quad j = 1, \ldots, N,$$

for some global weight vector w_γ, and all N bias parameters to be related by

$$\eta_{\gamma,j} = \eta_\gamma + \rho_{\gamma,j} \quad j = 1, \ldots, N$$

for some global bias parameter η_γ. Thus, using the homogeneity assumption, we can rewrite equation C.1 as

$$p(c_j = \gamma \mid b_j = \beta) = \frac{\exp\left(w_\gamma^T \phi(\beta) + \eta_\gamma + \rho_{\gamma,j}\right)}{\sum_\Gamma \exp(w_\Gamma^T \phi(\beta) + \eta_\Gamma + \rho_{\Gamma,j})}. \tag{C.7}$$

The different conditional probabilities are now expressed in terms of the global weight vector w_γ and the global bias parameter η_γ, and the logarithms of the prior class probabilities $\rho_{\gamma,j}$. Note that the only dependence on j is now via the offsets $\rho_{\gamma,j}$. In other words, the homogeneity assumption boils down to sharing parameters in combination with j-specific offsets for the bias parameter.

The global weight vector and global bias parameters in equation C.7 can be trained by using (regularized) maximum likelihood in a way very similar to ordinary (regularized) logistic regression training, with the only difference that the bit-dependent offsets $\rho_{\gamma,j}$ (which are given at training time) have to be taken into account. Although this is a possible approach, a disadvantage is that this particular training method has to be implemented—most off-the-shelf methods do not support offsets in the bias parameter that can differ for each training point.

Alternatively, we may consider the mixture distribution obtained by mixing the different bits $j = 1, \ldots, N$ together:

$$\begin{aligned} f(\gamma, \beta) &:= \frac{1}{N} \sum_{j=1}^{N} p(c_j = \gamma, b_j = \beta) \\ &= \frac{1}{N} \sum_{j=1}^{N} p(c_j = \gamma) p(b_j = \beta \mid c_j = \gamma) \\ &= p(\beta \mid \gamma) \frac{1}{N} \sum_{j=1}^{N} p(c_j = \gamma) \\ &= p(\beta \mid \gamma) f(\gamma). \end{aligned} \tag{C.8}$$

By rewriting equation C.2 in terms of the global parameters w_γ and η_γ introduced above, we can express the conditional distribution $p(\beta \mid \gamma)$ in equation C.8 as follows:

$$p(\beta \mid \gamma) = \exp\left(w_\gamma^T \phi(\beta) + \eta_\gamma\right) \psi_j(\beta),$$

with j arbitrary. Again, a logistic regression form appears for the conditional distribution $f(\gamma \mid \beta)$:

$$f(\gamma \mid \beta) = \frac{f(\gamma, \beta)}{\sum_\Gamma f(\Gamma, \beta)} = \frac{\exp\left(w_\gamma^T \phi(\beta) + \eta_\gamma + \log f(\gamma)\right)}{\sum_\Gamma \exp\left(w_\Gamma^T \phi(\beta) + \eta_\Gamma + \log f(\Gamma)\right)}.$$

This suggests an alternative way of learning the global weight vector w_γ and the global bias parameter η_γ: simply aggregate the training examples corresponding to different bits $j = 1, \ldots, N$ together into one large pool, and train a logistic regression classifier from the aggregated data. Asymptotically this will yield the same global weight vector as before, and the resulting global bias parameter differs by only the logarithm of the global bias correction factor $f(\gamma)$. Although the two different training methods are asymptotically equivalent (where some care needs to be taken with the regularization), the latter method is much easier to implement in practice, as one can use an off-the-shelve logistic regression classifier.

Acknowledgments

We thank Karin Bierig for the experimental help.

References

Bauby, J.-D. (1998). *The diving bell and the butterfly: A memoir of life in death*. New York: Vintage International.

Collins, M. (2002). Discriminative training methods for hidden Markov models: Theory and experiments with perceptron algorithms. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, *10* (pp. 1–8). Norwell, MA: Kluwer.

Farwell, L. A., & Donchin, E. (1988). Talking off the top of your head: Toward a mental prosthesis utilizing event-related brain potentials. *Electroencephalogr. Clin. Neurophysiol.*, *70*, 510–523.

Guger, C., Daban, S., Sellers, E., Holzner, C., Krausz, G., Carabalona, R., et al. (2009). How many people are able to control a P300–based brain-computer interface (BCI)? *Neuroscience Letters*, *462*, 94–98.

Hamming, R. W. (1950). Error detecting and error correcting codes. *Bell System Technical Journal*, *29*, 147–160.

Hastie, T., Tibshirani, R., & Friedman, J. (2001). *The elements of statistical learning*. Berlin: Springer.

Hill, N. J., Farquhar, J., Martens, S. M. M., Biessman, F., & Schölkopf, B. (2009). Effects of stimulus type and of error-correcting code design on BCI speller performance. In D. Schuurmans & Y. Bengio (Eds.), *Advances in neural information processing systems, 21* (pp. 665–672). Cambridge, MA: MIT Press.

Kapadia, S. (1998). *Discriminative training of hidden Markov models*. Unpublished doctoral dissertation, Cambridge University.

Kaper, M., Meinicke, P., Grossekathoefer, U., Lingner, T., & Ritter, H. (2004). BCI Competition 2003—Data set IIb: Support vector machines for the P300 speller paradigm. *IEEE Trans. Biomed. Eng.*, *51*, 1073–1076.

Koller, D., & Friedman, N. (2009). *Probabilistic graphical models: Principles and techniques*. Cambridge, MA: MIT Press.

Krusienski, D. J., Sellers, E. W., Cabestaing, F., Bayoudh, S., McFarland, D. J., Vaughan, T. M., et al. (2006). A comparison of classification techniques for the P300 speller. *J. Neural. Eng.*, *3*, 299–305.

Levenshtein, V. I. (1964). Application of the Hadamard matrices to a problem in coding. *Problems of Cybernetics*, *5*, 166–184.

Martens, S. M. M., Hill, N. J., Farquhar, J., & Schölkopf, B. (2009). Overlap and refractory effects in a brain-computer interface speller based on the visual P300 Event-Related Potential. *Journal of Neural Engineering*, *6*, 026003.

Martens, S. M. M., & Leiva, J. M. (2010). A generative model approach for decoding in the visual ERP-based BCI speller. *Journal of Neural Engineering*, *7*, 026003.

Rakotomamonjy, A., & Guigue, V. (2008). BCI competition III: Dataset II—Ensemble of SVMs for BCI P300 speller. *IEEE Trans. Biomed. Eng.*, *55*, 1147–1154.

Received March 19, 2010; accepted June 23, 2010.