

2008 Special Issue

Variational Bayesian least squares: An application to brain–machine interface data[☆]

Jo-Anne Ting^{a,*}, Aaron D'Souza^b, Kenji Yamamoto^c, Toshinori Yoshioka^d, Donna Hoffman^e, Shinji Kakei^f, Lauren Sergio^g, John Kalaska^h, Mitsuo Kawato^d, Peter Strick^e, Stefan Schaal^{a,d}

^a University of Southern California, Los Angeles, CA, 90089, USA

^b Google, Inc., Mountain View, CA 94043, USA

^c National Institute of Radiological Sciences, Chiba 263-8555, Japan

^d ATR Computational Neuroscience Laboratories, Kyoto 619-0288, Japan

^e University of Pittsburgh, Pittsburgh, PA 15261, USA

^f Tokyo Metropolitan Institute for Neuroscience, Tokyo 183-8526, Japan

^g York University, Toronto, Ontario, Canada M3J 1P3

^h Université de Montréal, Montréal, Canada H3C 3J7

ARTICLE INFO

Article history:

Received 15 December 2007

Received in revised form

10 June 2008

Accepted 17 June 2008

Keywords:

High-dimensional regression

Variational Bayesian methods

Linear models

Dimensionality reduction

Feature selection

Brain–machine interfaces

EMG prediction

Statistical learning

ABSTRACT

An increasing number of projects in neuroscience require statistical analysis of high-dimensional data, as, for instance, in the prediction of behavior from neural firing or in the operation of artificial devices from brain recordings in brain–machine interfaces. Although prevalent, classical linear analysis techniques are often numerically fragile in high dimensions due to irrelevant, redundant, and noisy information. We developed a robust Bayesian linear regression algorithm that automatically detects relevant features and excludes irrelevant ones, all in a computationally efficient manner. In comparison with standard linear methods, the new Bayesian method regularizes against overfitting, is computationally efficient (unlike previously proposed variational linear regression methods, is suitable for data sets with large numbers of samples and a very high number of input dimensions) and is easy to use, thus demonstrating its potential as a drop-in replacement for other linear regression techniques. We evaluate our technique on synthetic data sets and on several neurophysiological data sets. For these neurophysiological data sets we address the question of whether EMG data collected from arm movements of monkeys can be faithfully reconstructed from neural activity in motor cortices. Results demonstrate the success of our newly developed method, in comparison with other approaches in the literature, and, from the neurophysiological point of view, confirms recent findings on the organization of the motor cortex. Finally, an incremental, real-time version of our algorithm demonstrates the suitability of our approach for real-time interfaces between brains and machines.

© 2008 Elsevier Ltd. All rights reserved.

1. Introduction

In recent years, there has been growing interest in large scale

[☆] This research was supported in part by National Science Foundation grants ECS-0325383, IIS-0312802, IIS-0082995, ECS-0326095, ANI-0224419, a NASA grant AC#98-516, an AFOSR grant on Intelligent Control, the ERATO Kawato Dynamic Brain Project funded by the Japanese Science and Technology Agency, the ATR Computational Neuroscience Laboratories, funds from the Department of Veterans Affairs, Medical Research Service, NIH grant P01 NS044393 (PLS), and funds from CIHR and FRSQ. KY was partially supported by a Japan Society for the Promotion of Sciences Postdoctoral Fellowship for research abroad.

* Corresponding address: University of Southern California, Hedco Neurosciences Building, 3641 Watt Way, Los Angeles, CA 90089, USA. Tel.: +1 213 740 6717; fax: +1 213 740 5687.

E-mail address: joanneti@usc.edu (J.-A. Ting).

analyses of brain activity, with respect to associated behavioral variables. For instance, projects can be found in the area of brain–machine interfaces, where neural firing is directly used to control an artificial system like a robot (Chapin, Moxon, Markowitz, & Nicolelis, 1999; Hochberg et al., 2006; Lebedev & Nicolelis, 2006; Nicolelis, 2001; Nicolelis & Ribeiro, 2006; Taylor, Tillery, & Schwartz, 2002), or where non-invasive brain signals serve to either control a cursor on computer screen (Wolpaw & McFarland, 2004), or to classify visual stimuli presented to a subject (Haynes & Rees, 2005; Kamitani & Tong, 2004). In such scenarios, the brain signals to be processed are typically high dimensional, in the order of hundreds or thousands of inputs, with large numbers of redundant and irrelevant signals. Linear modeling techniques like linear regression are among

the primary analysis tools for such data (Lebedev & Nicolelis, 2006; Musallam, Corneil, Greger, Scherberger, & Andersen, 2004; Wessberg & Nicolelis, 2004). However, the computational problem of data analysis not only involves data fitting, but also requires that the model extracted from the data has good generalization properties. This issue is crucial for predicting behavior from future neural recordings, e.g., for continual on-line interpretation of brain activity to control prosthetic devices, or for longitudinal scientific studies of information processing in the brain. Surprisingly, robust linear modeling of high-dimensional data is non-trivial, as the danger of fitting noise and of encountering numerical problems is high. Classical techniques like ridge regression, stepwise regression, subset selection techniques, or Partial Least Squares regression (Wold, 1975) are known to be prone to overfitting, and may often require careful human supervision to ensure useful results. Other methods such as Least Absolute Shrinkage and Selection Operator (LASSO) regression (Tibshirani, 1996) attempt to shrink certain regression coefficients to zero, resulting in interpretable models that are sparse. However, LASSO regression has an open parameter that needs to be set, using either n -fold cross-validation or manual hand-tuning.

In this paper, we will focus on how to improve linear data analysis for the high-dimensional scenarios described above, with a view towards developing a “black box” approach that automatically detects the most relevant input dimensions for generalization and excludes other dimensions in a statistically sound way. We are particularly interested in situations where the data contains a very large quantity of samples and the number of input dimensions is very high, as in brain–machine interfaces. For this purpose, we investigate a full Bayesian treatment of linear regression with automatic relevance detection (Neal, 1994) that is computationally efficient and suitable for large amounts of very high-dimensional data. This algorithm can be formulated in closed form with the help of a variational Bayesian approximation, and it introduces “probabilistic backfitting” for linear regression, a key component which contributes greatly towards the algorithm’s computational efficiency. Besides several synthetic data evaluations, we apply the algorithm, named Variational Bayesian Least Squares (VBLS) (Ting et al., 2005), to the reconstruction of EMG data from motor cortical firing from data sets collected by Sergio and Kalaska (1998) and Kakei, Hoffman, and Strick (1999, 2001). This data analysis addresses important neurophysiological questions in terms of whether motor cortical neurons can directly predict EMG traces (Bennett & Lemon, 1996; McKiernan, Marcario, Karrer, & Cheney, 1998; Morrow & Miller, 2003; Todorov, 2000; Townsend, Paninski, & Lemon, 2006), whether motor cortices have a muscle-based topological organization, and whether information in motor cortices should be used to predict behavior in future brain–machine interfaces. Our main focus in this paper is on the statistical analysis of these kinds of data. Comparisons with classical linear analysis techniques and a brute force combinatorial model search (which was executed on a cluster computer) demonstrate that our VBLS algorithm indeed achieves the “black box” quality of a statistical analysis technique that requires no tuning of parameters by the user.

This paper describes in detail the VBLS algorithm and its application to the EMG reconstruction problem by building and extending our prior work in D’Souza, Vijayakumar, and Schaal (2004) and Ting et al. (2005). We discuss the neurophysiological implications of our analyses and present a real-time version of VBLS in order to simulate an application in real-time brain machine interfaces.

2. High dimensional regression

Before developing our VBLS algorithm, it is useful to briefly revisit classical linear regression techniques. Assuming there are N observed data samples in the data set $D = \{\mathbf{x}_i, y_i\}_{i=1}^N$ (where $\mathbf{x}_i \in \mathfrak{R}^{d \times 1}$ are inputs and y_i are scalar outputs), the standard model for linear regression is:

$$y_i = \sum_{m=1}^d b_m x_{im} + \epsilon \quad (1)$$

where \mathbf{b} is the regression vector made up of b_m components, d is the number of input dimensions, and ϵ is additive mean-zero noise. The Ordinary Least Squares (OLS) estimate of the regression vector is $\mathbf{b} = (\mathbf{X}^T \mathbf{X})^{-1} \mathbf{X}^T \mathbf{y}$, where $\mathbf{X} \in \mathfrak{R}^{N \times d}$ consists of vectors \mathbf{x}_i arranged in its rows and $\mathbf{y} \in \mathfrak{R}^{N \times 1}$ has coefficients y_i . The main problem with OLS regression in high-dimensional input spaces is that the full rank assumption of $(\mathbf{X}^T \mathbf{X})^{-1}$ is often violated due to underconstrained data sets. Ridge regression (Hoerl & Kennard, 1970) can “fix” such problems numerically by stabilizing the matrix inversion with a diagonal term $(\mathbf{X}^T \mathbf{X} + \alpha \mathbf{I})^{-1}$, but usually introduces uncontrolled bias. Additionally, if the input dimensionality exceeds around 1000 dimensions, the matrix inversion can become prohibitively computationally expensive.

Several ideas exist how to improve over OLS. First, stepwise regression (Draper & Smith, 1981) can be employed. However, stepwise regression has been strongly criticized for its potential for overfitting and its inconsistency in the presence of collinearity in the input data (Derksen & Keselman, 1992). To deal with such collinearity directly, dimensionality reduction techniques like Principal Components Regression (PCR) (Massey, 1965) are useful. These methods retain directions in an input space with large variance, regardless of whether the directions influence the prediction (Schaal, Vijayakumar, & Atkeson, 1998), and can even eliminate low variance inputs that may have high predictive power for the outputs (Frank & Friedman, 1993). Another class of linear regression methods is projection regression techniques, most notably Partial Least Squares (PLS) regression (Wold, 1975). PLS regression performs computationally inexpensive $O(d)$ univariate regressions along projection directions, chosen according to the correlation between inputs and outputs. While slightly heuristic in nature, PLS regression is a surprisingly successful algorithm for ill-conditioned and high-dimensional regression problems, although it also has a tendency towards overfitting (Schaal et al., 1998). There are also more efficient methods for matrix inversion (Hastie & Tibshirani, 1990; Strassen, 1969), but these methods assume a well-condition regression problem *a priori* and degrade in the presence of collinearities in inputs. Finally, there is a class of sparsity inducing methods such as LASSO regression (Tibshirani, 1996) that attempt to shrink certain regression coefficients in the solution to zero by using an L1 penalty norm (instead of an L2 penalty norm used by ridge regression). These methods are suitable for high-dimensional data sets, at the expense of requiring an open parameter (i.e., a fixed bound on the penalty norm) that needs to be set using cross-validation. Note that previous methods of sparse variational linear regression have been proposed by Bishop (2006) and Tipping (2001), however these are not computationally efficient and are unsuitable for large amounts of high-dimensional data.

We will use some of the previously described methods for comparison in the Evaluation section. In particular, we will compare our proposed algorithm to the following methods: (i) OLS regression, (ii) ridge regression with an empirically tuned ridge value, (iii) stepwise regression, (iv) PLS regression and (v) LASSO regression. In the next section, we will introduce

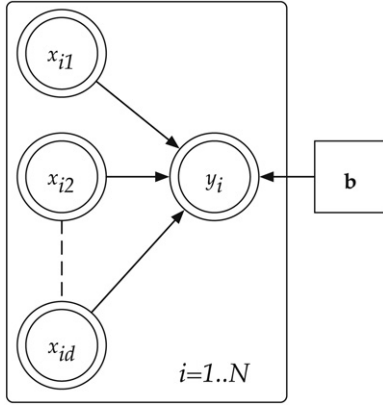


Fig. 1. Graphical model for linear regression. Random variables are in circular nodes, observed random variables are in double circles, and point estimated parameters are in square nodes.

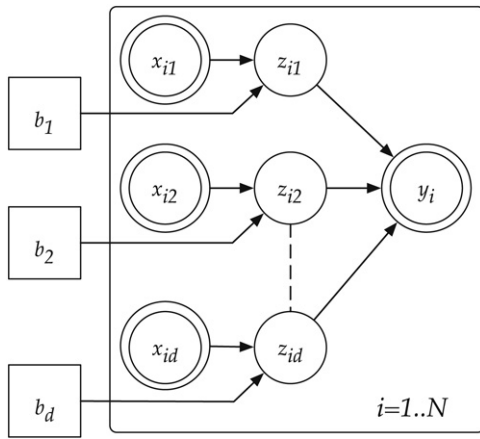


Fig. 2. Graphical model for Probabilistic Backfitting. Random variables are in circular nodes, observed random variables are in double circles, and point estimated parameters are in square nodes.

a linear regression algorithm in a Bayesian framework that *automatically* regularizes against problems of overfitting (in contrast, LASSO regression has an open parameter that requires cross-validation in order to find its optimal value). Additionally, the iterative nature of the algorithm – due to its formulation as an Expectation-Maximization problem (Dempster, Laird, & Rubin, 1977) – avoids the computational cost and numerical problems of matrix inversions that is faced in high-dimensional OLS regression and in Bishop (2006) and Tipping (2001). Thus, VBLS addresses the two major problems of high-dimensional OLS regression simultaneously. Note, however, that if accurate results are needed (and computational resources are unlimited) for data sets with fully relevant input dimensions, VBLS is not as efficient as the matrix inversion in OLS. The advantage of VBLS arises when dealing with high dimensional input spaces, serving as an efficient and robust “automatic” regression method. Conceptually, the algorithm can be interpreted as a Bayesian version of either backfitting or Partial Least Squares regression.

3. Variational Bayesian least squares

Figs. 1–3 illustrate the progression of graphical models that we need to develop a robust Bayesian version of linear regression. Fig. 1 depicts the standard linear regression model. Part of the inspiration for our algorithm comes from PLS regression, motivated by the question of how to find maximally predictive projections in input space, which is also part of various other

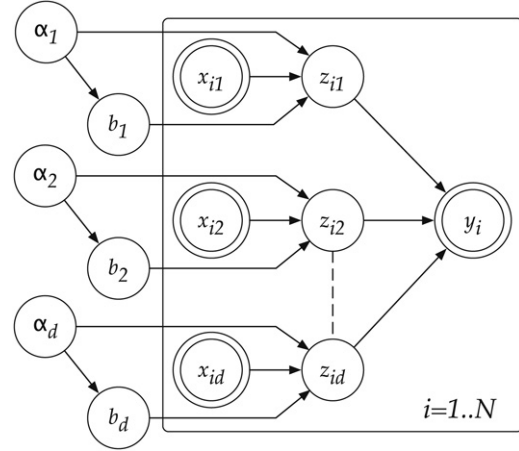


Fig. 3. Graphical model for Variational Bayesian Least Squares. Random variables are in circular nodes, observed random variables are in double circles, and point estimated parameters are in square nodes.

“subset” selection techniques in regression (Wessberg & Nicolelis, 2004). Indeed, if we knew the optimal projection direction of the input data, the entire regression problem could be solved by a univariate regression between the projected data and the outputs: this optimal projection direction is simply the true gradient between inputs and outputs. Since we do not know this projection direction, we now encode its coefficients as hidden variables, in the tradition of Expectation-Maximization (EM) algorithms (Dempster et al., 1977). Fig. 2 shows the corresponding graphical model. The unobservable variables z_{im} (where $i = 1, \dots, N$ denotes the index into the data set of N data points) are the result of the input variables being projected on the respective projection direction component (i.e., b_m). Then, the z_{im} ’s are summed up to form a predicted output y_i . More formally, we can modify the linear regression model in Eq. (1) to become:

$$y_i = \sum_{m=1}^d z_{im} + \epsilon_y \tag{2}$$

$$z_{im} = b_m x_{im} + \epsilon_{z_m}. \tag{3}$$

For a probabilistic treatment with EM, we make a standard normal assumption of all distributions in form of:

$$y_i | \mathbf{z}_i \sim \text{Normal}(\mathbf{1}^T \mathbf{z}_i, \psi_y) \tag{4}$$

$$z_{im} | x_{im} \sim \text{Normal}(b_m x_{im}, \psi_{z_m})$$

where $\mathbf{1} = [1, 1, \dots, 1]^T$. While this model is still identical to OLS, notice that in the graphical model of Fig. 2, the regression coefficients b_m are behind the fan-in to the outputs y_i . We call this model Probabilistic Backfitting, since the resulting derived update equation for the regression coefficient b_m can be viewed as a probabilistic version of backfitting. Given the data D , we can view this new regression model as an EM problem and maximize the incomplete log likelihood $\log p(\mathbf{y} | \mathbf{X})$ by maximizing the expected complete log likelihood ($\log p(\mathbf{y}, \mathbf{Z} | \mathbf{X})$), where:

$$\begin{aligned} \log p(\mathbf{y}, \mathbf{Z} | \mathbf{X}) = & -\frac{N}{2} \log \psi_y - \frac{1}{2\psi_y} \sum_{i=1}^N (y_i - \mathbf{1}^T \mathbf{z}_i)^2 \\ & - \frac{N}{2} \sum_{m=1}^d \log \psi_{z_m} - \sum_{m=1}^d \frac{1}{2\psi_{z_m}} (z_{im} - b_m x_{im})^2 + \text{const} \end{aligned} \tag{5}$$

where $\mathbf{Z} \in \mathfrak{R}^{N \times d}$ consists of z_{im} components. The resulting EM updates require standard manipulations of normal distributions

and are shown below:

E-step:

$$\mathbf{1}^T \Sigma_z \mathbf{1} = \left(\sum_{m=1}^d \psi_{zm} \right) \left[1 - \frac{1}{s} \left(\sum_{m=1}^d \psi_{zm} \right) \right] \quad (6)$$

$$\sigma_{zm}^2 = \psi_{zm} \left(1 - \frac{1}{s} \psi_{zm} \right) \quad (7)$$

$$\langle z_{im} \rangle = b_m \mathbf{x}_i + \frac{1}{s} \psi_{xm} (y_i - \mathbf{b}^T \mathbf{x}_i) \quad (8)$$

M-step:

$$b_m = \frac{\sum_{i=1}^N \langle z_{im} \rangle x_{im}}{\sum_{i=1}^N x_{im}^2} \quad (9)$$

$$\psi_y = \frac{1}{N} \sum_{i=1}^N (y_i - \mathbf{1}^T \langle \mathbf{z}_i \rangle)^2 + \mathbf{1}^T \Sigma_z \mathbf{1} \quad (10)$$

$$\psi_{zm} = \frac{1}{N} \sum_{i=1}^N (\langle z_{im} \rangle - b_m x_{im})^2 + \sigma_{zm}^2 \quad (11)$$

where we define $s = \psi_y + \sum_{m=1}^d \psi_{xm}$ and Σ_z is the covariance matrix of \mathbf{z} . It is very important to note that one EM update has a computational complexity of $O(d)$, where d is the number of input dimensions, instead of the $O(d^3)$ associated with OLS regression. This efficiency comes at the cost of an iterative solution, instead of a one-shot solution for \mathbf{b} as in OLS. It can be proved that this EM version of least squares regression is guaranteed to converge to the same solution as OLS (D'Souza et al., 2004).

This new EM algorithm appears to only replace the matrix inversion in OLS by an iterative method, as others have done with alternative algorithms (Hastie & Tibshirani, 1990; Strassen, 1969). However, the convergence guarantee of EM is an improvement over previous approaches. The true power of this probabilistic formulation becomes apparent when we add a Bayesian layer to achieve robustness in face of ill-conditioned data.

3.1. Automatic feature detection

From a Bayesian point of view, the parameters b_m should be treated probabilistically as well, such that we can integrate them out to safeguard against overfitting. For this purpose, as shown in Fig. 3, we introduce precision variables α_m over each regression parameter b_m , as previously done in Tipping (2001):

$$p(\mathbf{b}|\boldsymbol{\alpha}) = \prod_{m=1}^d \left(\frac{\alpha_m}{2\pi} \right)^{\frac{1}{2}} \exp \left(-\frac{\alpha_m}{2} b_m^2 \right) \quad (12)$$

$$p(\boldsymbol{\alpha}) = \prod_{m=1}^d \frac{b_{\alpha_m}^{\alpha_m}}{\text{Gamma}(a_{\alpha_m})} \alpha_m^{(a_{\alpha_m}-1)} \exp(-b_{\alpha_m} \alpha_m)$$

where $\boldsymbol{\alpha} \in \mathfrak{R}^{d \times 1}$ consists of α_m components. We now have a mechanism that infers the significance of each dimension's contribution to the observed output y . The key quantity that determines the relevance of a regression input is the parameter α_m . *A priori*, we assume that every b_m has a mean zero distribution with broad variance $1/\alpha_m$. If the posterior value of α_m turns out to be very large after all model parameters are estimated (equivalent to a very small variance of b_m), then the corresponding distribution of b_m must be sharply peaked at zero. Such a posterior gives strong evidence that b_m is very close to 0 and that the regression input \mathbf{x}_m has no contribution to the output. Thus, this Bayesian model

automatically detects irrelevant input dimensions and regularizes against ill-conditioned data sets.

Even though Eq. (12) looks very similar to that of Tipping (2001) and later work of Bishop (2006), our model has the key property that it is computationally efficient, requiring $O(d)$ per EM iteration. In contrast, the methods of Bishop (2006) and Tipping (2001) take $O(d^3)$ per EM iteration and $O(N^3)$, respectively, becoming prohibitively expensive for large data sets with a very large input dimensionality, d . It is the fast, efficient nature of our proposed algorithm, Variational Bayesian Least Squares, that makes it suitable for real-time analysis of very large amounts of very high-dimensional data, as required in brain-machine interfaces. We discuss this application in more detail in the Evaluation section. The final model for VBLS has the following distributions:

$$y_i | \mathbf{z}_i \sim \text{Normal}(\mathbf{1}^T \mathbf{z}_i, \psi_y)$$

$$z_{im} | b_m, \alpha_m, x_{im} \sim \text{Normal} \left(b_m x_{im}, \frac{\psi_{zm}}{\alpha_m} \right) \quad (13)$$

$$b_m | \alpha_m \sim \text{Normal} \left(0, \frac{1}{\alpha_m} \right)$$

$$\alpha_m \sim \text{Gamma}(a_{\alpha_m}, b_{\alpha_m}).$$

As a note, it should be observed that the Gaussian prior used above for b_m is a standard prior in Bayesian linear regression, e.g., Bishop (2006). However, the Laplace prior could be used as well, and the result, when used with MAP estimation, will be similar to LASSO. We choose to not pursue this direction, but note that the Laplace density can be re-written in a hierarchical manner as done above by modeling the variance of b_m as a Gamma distribution with one hyperparameter, i.e., an exponential, as done by Figueiredo (2003). Integrating out the hyperparameter gives the Laplace marginal prior.

An EM-like algorithm (Ghahramani & Beal, 2000) can be used to find the posterior updates of all distributions, where we maximize the incomplete log likelihood $\log p(\mathbf{y}|\mathbf{X})$ by maximizing the expected complete log likelihood $\langle \log p(\mathbf{y}, \mathbf{Z}, \mathbf{b}, \boldsymbol{\alpha}|\mathbf{X}) \rangle$:

$$\log p(\mathbf{y}, \mathbf{Z}, \mathbf{b}, \boldsymbol{\alpha}|\mathbf{X})$$

$$= \sum_{i=1}^N \log p(y_i | \mathbf{z}_i) + \sum_{i=1}^N \sum_{m=1}^d \log p(z_{im} | b_m, \alpha_m)$$

$$+ \sum_{m=1}^d \log p(b_m | \alpha_m) + \sum_{m=1}^d \log p(\alpha_m)$$

$$= -\frac{N}{2} \log \psi_y - \frac{1}{2\psi_y} \sum_{i=1}^N (y_i - \mathbf{1}^T \mathbf{z}_i)^2 - \frac{N}{2} \sum_{m=1}^d \log \frac{\psi_{zm}}{\alpha_m}$$

$$- \sum_{m=1}^d \frac{\alpha_m}{2\psi_{zm}} (z_{im} - b_m x_{im})^2$$

$$+ \sum_{m=1}^d \log \alpha_m - \frac{1}{2} \sum_{m=1}^d \alpha_m b_m^2 + \sum_{m=1}^d (a_{\alpha_m,0} - 1) \log \alpha_m$$

$$- \sum_{m=1}^d b_{\alpha_m,0} \alpha_m + \text{const} \quad (14)$$

where $a_{\alpha_m,0}$ and $b_{\alpha_m,0}$ are the initial parameter values that are set to reflect our confidence in the prior distribution of b_m . In order to obtain a tractable posterior distribution over all hidden variables \mathbf{b} , \mathbf{z}_i and $\boldsymbol{\alpha}$, we use a factorial variational approximation of the true posterior (Ghahramani & Beal, 2000): $Q(\boldsymbol{\alpha}, \mathbf{b}, \mathbf{Z}) = Q(\boldsymbol{\alpha}, \mathbf{b})Q(\mathbf{Z})$. Note that the connection from the α_m to the corresponding z_{im} in Fig. 3 is an intentional design. Under this graphical model, the marginal distribution of b_m becomes a Student t -distribution, allowing for traditional hypothesis testing (Gelman, Carlin, Stern,

& Rubin, 2000). The minimal factorization of the posterior into $Q(\boldsymbol{\alpha}, \mathbf{b})Q(\mathbf{Z})$ would not be possible without this special design.

The variational Bayesian approximation used here allows us to reach a tractable posterior distribution over all hidden variables, such that we can proceed to infer the posterior distributions. Variational Bayesian learning approximates the intractable joint distribution over hidden states and parameters with a simpler distribution, e.g., assuming independence between hidden states and parameters such that the posterior distributions are factorized. An exact Bayesian solution is not feasible since one would need to compute the marginals of the joint posterior distribution—and this is not analytically possible. For discussions on the quality of variational Bayesian approximations and how they compare to the true solution, please refer to Attias (2000), Ghahramani and Beal (2000), Jaakkola (2000) and Jordan, Ghahramani, Jaakkola, and Saul (1999). We will return to this point in the Discussion section.

After some algebraic manipulations, the final EM posterior update equations become:

E-step:

$$\begin{aligned} \boldsymbol{\Sigma}_z &= \left(\frac{1}{\psi_y} \mathbf{1}\mathbf{1}^T + \boldsymbol{\Psi}_z^{-1} \langle \mathbf{A} \rangle \right)^{-1} \\ &= \boldsymbol{\Psi}_z \langle \mathbf{A} \rangle^{-1} - \frac{\boldsymbol{\Psi}_z \langle \mathbf{A} \rangle^{-1} \mathbf{1}\mathbf{1}^T \boldsymbol{\Psi}_z \langle \mathbf{A} \rangle^{-1}}{\psi_y + \mathbf{1}^T \boldsymbol{\Psi}_z \langle \mathbf{A} \rangle^{-1} \mathbf{1}} \end{aligned} \quad (15)$$

$$\begin{aligned} \langle z_i \rangle &= \boldsymbol{\Sigma}_z \left(\frac{1}{\psi_y} \mathbf{1}y_i + \boldsymbol{\Psi}_z^{-1} \langle \mathbf{A} \rangle \langle \mathbf{B}|\mathbf{A} \rangle \mathbf{x}_i \right) \\ &= \left(\frac{\boldsymbol{\Psi}_z \langle \mathbf{A} \rangle^{-1} \mathbf{1}}{\psi_y + \mathbf{1}^T \boldsymbol{\Psi}_z \langle \mathbf{A} \rangle^{-1} \mathbf{1}} \right) y_i \\ &\quad + \left(\langle \mathbf{B}|\mathbf{A} \rangle - \frac{\boldsymbol{\Psi}_z \langle \mathbf{A} \rangle^{-1} \mathbf{1}\mathbf{1}^T \langle \mathbf{B}|\mathbf{A} \rangle}{\psi_y + \mathbf{1}^T \boldsymbol{\Psi}_z \langle \mathbf{A} \rangle^{-1} \mathbf{1}} \right) \mathbf{x}_i \end{aligned} \quad (16)$$

$$\sigma_{b_m|\alpha_m}^2 = \frac{\psi_{zm}}{\langle \alpha_m \rangle} \left(\sum_{i=1}^N x_{im}^2 + \psi_{zm} \right)^{-1} \quad (17)$$

$$\langle b_m|\alpha_m \rangle = \left(\sum_{i=1}^N x_{im}^2 + \psi_{zm} \right)^{-1} \left(\sum_{i=1}^N \langle z_{im} \rangle x_{im} \right) \quad (18)$$

$$\hat{a}_{\alpha_m} = a_{\alpha_m,0} + \frac{N}{2} \quad (19)$$

$$\begin{aligned} \hat{b}_{\alpha_m} &= b_{\alpha_m,0} + \frac{1}{2\psi_{zm}} \left\{ \sum_{i=1}^N \langle z_{im}^2 \rangle - \left(\sum_{i=1}^N x_{im}^2 + \psi_{zm} \right)^{-1} \right. \\ &\quad \left. \times \left(\sum_{i=1}^N \langle z_{im} \rangle x_{im} \right)^2 \right\} \end{aligned} \quad (20)$$

$$\langle \alpha_m \rangle = \frac{\hat{a}_{\alpha_m}}{\hat{b}_{\alpha_m}} \quad (21)$$

M-step:

$$\psi_y = \frac{1}{N} \sum_{i=1}^N (y_i - \mathbf{1}^T \langle z_i \rangle)^2 + \mathbf{1}^T \boldsymbol{\Sigma}_z \mathbf{1} \quad (22)$$

$$\begin{aligned} \psi_{zm} &= \frac{1}{N} \sum_{i=1}^N \langle \alpha_m \rangle (\langle z_{im} \rangle - \langle b_m|\alpha_m \rangle x_{im})^2 + \langle \alpha_m \rangle \sigma_{z_m}^2 \\ &\quad + \langle \alpha_m \rangle \sigma_{b_m|\alpha_m}^2 \left(\frac{1}{N} \sum_{i=1}^N x_{im}^2 \right) \end{aligned} \quad (23)$$

where $\langle \mathbf{A} \rangle$, $\langle \mathbf{B}|\mathbf{A} \rangle$, $\boldsymbol{\Psi}_z$ are diagonal matrices of $\langle \boldsymbol{\alpha} \rangle$, $\langle \mathbf{b}|\boldsymbol{\alpha} \rangle$, ψ_z , respectively. $\boldsymbol{\Sigma}_z$ is a diagonal covariance matrix with a diagonal

vector of σ_z^2 . Note that $\langle z_{im}^2 \rangle = \langle z_{im} \rangle^2 + \sigma_{z_m}^2$, where $\sigma_{z_m}^2$ is the m th term of the vector σ_z^2 .

The hyperparameters of α_m are learnt using EM, as shown by Eqs. (19) and (20). We set the initial values of the hyperparameters, $a_{\alpha,0}$ and $b_{\alpha,0}$, in an uninformative way and use values of $a_{\alpha_m,0} = 10^{-8}$ and $b_{\alpha_m,0} = 10^{-8}$ for all $m = 1, \dots, d$. This means that initial value of α_m is 1, with high uncertainty, i.e., α_m has a rather flat prior distribution.

Note that the update equation for $\langle b_m|\alpha_m \rangle$ can be rewritten as:

$$\begin{aligned} \langle b_m|\alpha_m \rangle^{(n+1)} &= \left(\frac{\sum_{i=1}^N x_{im}^2}{\sum_{i=1}^N x_{im}^2 + \psi_{zm}} \right) \langle b_m|\alpha_m \rangle^{(n)} \\ &\quad + \frac{\psi_{zm} \sum_{i=1}^N (y_i - \langle \mathbf{b}|\boldsymbol{\alpha} \rangle^{(n)T} \mathbf{x}_i) x_{im}}{\sigma_{\alpha_m} \sum_{i=1}^N x_{im}^2 + \psi_{zm}}. \end{aligned} \quad (24)$$

Eq. (24) demonstrates that in the absence of a correlation between the current input dimension and the residual error, the first term causes the current regression coefficient to decay. The resulting regression solution regularizes over the number of retained inputs in the final regression vector, performing a functionality similar to Automatic Relevance Determination (ARD) (Neal, 1994). The update equations of VBLS have an algorithmic complexity of $O(d)$ per EM iteration, making it suitable for real-time analysis of large amounts of high-dimensional data—unlike previously proposed computationally prohibitive sparse linear regression methods that require $O(d^3)$ per EM iteration (Bishop, 2006) or $O(N^3)$ (Tipping, 2001). One can further show that the marginal distribution of all b_m is a t -distribution with $t = \langle b_m|\alpha_m \rangle / \sigma_{b_m|\alpha_m}$ and $2\hat{a}_{\alpha_m}$ degrees of freedom, which allows a principled way of determining whether a regression coefficient was excluded by means of standard hypothesis testing. Thus, Variational Bayesian Least Squares (VBLS) regression is a computationally efficient, full Bayesian treatment of the linear regression problem and is suitable for large amounts of high-dimensional data.

3.2. Pseudocode of variational Bayesian least squares

The pseudocode for VBLS is listed in Algorithm 1. To know when to stop iterating through the EM-based algorithm, we should monitor the incomplete log likelihood and stop when the value appears to have converged. However, since the calculation of the true posterior distribution $Q(\boldsymbol{\alpha}, \mathbf{b}, \mathbf{Z})$ is intractable, we cannot determine the true incomplete log likelihood. Hence, for the purpose of monitoring the incomplete log likelihood in the EM algorithm, we monitor a lower bound of the incomplete log likelihood instead. In the derivation of VBLS, we approximated $Q(\boldsymbol{\theta})$, where $\boldsymbol{\theta} = \{\boldsymbol{\alpha}, \mathbf{b}, \mathbf{Z}\}$, as $Q(\boldsymbol{\alpha}, \mathbf{b})Q(\mathbf{Z})$. Using this variational approximation, we can derive the lower bound to the incomplete log likelihood (where $\phi = \{\psi_y, \psi_z\}$) to be:

$$\begin{aligned} \log p(\mathbf{y}|\mathbf{X}; \phi) &\geq \int Q(\boldsymbol{\theta}) \log \frac{p(\mathbf{y}, \boldsymbol{\theta}|\mathbf{X}; \phi)}{Q(\boldsymbol{\theta})} d\boldsymbol{\theta} = \int Q(\boldsymbol{\theta}) \log p(\mathbf{y}, \boldsymbol{\theta}|\mathbf{X}; \phi) d\boldsymbol{\theta} \\ &\quad - \int Q(\boldsymbol{\theta}) \log Q(\boldsymbol{\theta}) d\boldsymbol{\theta} \\ &\geq \langle \log p(\mathbf{y}, \boldsymbol{\theta}|\mathbf{X}; \phi) \rangle_{Q(\boldsymbol{\theta})} - \int Q(\boldsymbol{\theta}) \log Q(\boldsymbol{\theta}) d\boldsymbol{\theta} \end{aligned} \quad (25)$$

where Eq. (25) simplifies to:

$$\begin{aligned}
\log p(\mathbf{y}|\mathbf{X}; \phi) &\geq -\frac{N}{2} \log \psi_y \\
&- \frac{1}{2\psi_y} \sum_{i=1}^N (y_i^2 - 2y_i \mathbf{1}^T \langle \mathbf{z}_i \rangle + \mathbf{1}^T \langle \mathbf{z}_i \mathbf{z}_i^T \rangle \mathbf{1}) - \frac{N}{2} \sum_{m=1}^d \log \psi_{zm} \\
&- \sum_{m=1}^d \frac{\langle \alpha_m \rangle}{2\psi_{zm}} \sum_{i=1}^N (\langle z_{im}^2 \rangle - 2 \langle z_{im} \rangle \langle b_m | \alpha_m \rangle x_{im} + \langle (b_m | \alpha_m)^2 \rangle x_{im}^2) \\
&- \frac{1}{2} \sum_{m=1}^d \langle \alpha_m \rangle \langle (b_m | \alpha_m)^2 \rangle - \frac{N-1}{2} \sum_{m=1}^d \log \hat{b}_{\alpha_m} - \hat{a}_{\alpha_m} \\
&- \frac{1}{2} \log |\Sigma_z^{-1}| - \sum_{m=1}^d \log \hat{b}_{\alpha_m} \\
&+ \frac{1}{2} \sum_{m=1}^d \langle \alpha_m \rangle (\sigma_{b_m | \alpha_m}^2 + 1) + \text{const.} \tag{26}
\end{aligned}$$

We stop iterating when the lower bound to the incomplete log likelihood has converged (i.e., when a certain likelihood tolerance, t , has been reached). Additionally, note that the input and output data are assumed to be centered (i.e. have a mean of 0) before we analyze the data set with VBLS.

Algorithm 1 Pseudocode for VBLS

- 0: **Initialization:** $a_{\alpha,0} = 10^{-8} \mathbf{1}$, $b_{\alpha,0} = 10^{-8} \mathbf{1}$; threshold value for lower bound to the incomplete log likelihood, $t = 10^{-6}$
 - 1: **Start EM iterations:**
 - 2: **repeat**
 - 3: Perform the E-step: Calculate Eqs. (15)–(20)
 - 4: Perform the M-step: Calculate Eqs. (22) and (23)
 - 5: Monitor the lower bound to the incomplete log likelihood, Eq. (26), to see if the likelihood tolerance t has been reached
 - 6: **until** convergence of Eq. (26)
-

4. Evaluation

We now turn to the application and evaluation of VBLS in the context of predicting EMG data from neural data recorded in primary motor (M1) and premotor (PM) cortices of monkeys. The key questions addressed in this application were (i) whether EMG data can be reconstructed accurately with good generalization, (ii) how many neurons contribute to the reconstruction of each muscle, and (iii) how well the VBLS algorithm compares to other analysis techniques. The underlying assumption of this analysis was that the relationship between cortical neural firing and muscle activity is approximately linear.

Before applying VBLS to real data, however, we first run it on synthetic data sets where “ground truth” is known, in order to better evaluate its performance in a controlled setting.

4.1. Synthetic data

4.1.1. Data sets

We generated random input training data consisting of 100 dimensions, 10 of which were relevant dimensions. The other 90 were either irrelevant or redundant dimensions, as we explain below. Each of the first 10 input dimensions was drawn from a Gaussian distribution with some random covariance. The output data was then generated from the relevant input data using the vector $\mathbf{b} \in \mathcal{R}^{10 \times 1}$, where each coefficient of \mathbf{b} , b_m , was drawn from a Normal(0, 100) distribution, subject to the fact that it cannot be

zero (since this would indicate an irrelevant dimension). Additive mean-zero Gaussian noise of varying levels was added to the outputs.

Noise in the outputs was parameterized with the coefficient of determination, r^2 , of standard linear regression, defined as:

$$r^2 = \frac{(\sigma_y^2 - \sigma_{\text{res}}^2)}{\sigma_y^2}$$

where σ_y^2 is the variance of the outputs and σ_{res}^2 is the variance of the residual error. We added noise scaled to the variance of the noiseless outputs \bar{y} such that $\sigma_{\text{noise}}^2 = c\sigma_y^2$, where $c = \frac{1}{r^2} - 1$. Results are quantified as normalized mean squared errors (nMSE), that is, the mean squared error on the test set normalized by the variance of the outputs of the test set. Note that the best normalized mean squared training error that can be achieved by the learning system under this noise level is $1 - r^2$, unless the system overfits the data. We used a value of $r^2 = 0.8$ for high output noise and a value of $r^2 = 0.9$ for lower output noise.

A varying number of redundant data vectors was added to the input data, generated from random convex combinations of the 10 relevant vectors. Finally, we added irrelevant data columns, drawn from a Normal(0, 1) distribution, until a total of 100 input dimensions was reached, generating training input data that contained irrelevant and redundant dimensions.

We created the test data set in a similar manner except that the input data and output data were left noise-free. For our experiments, we considered a synthetic training data set with $N = 1000$ data samples and a synthetic test data set with 20 data samples. We examined the following four different combinations of redundant, v , and irrelevant, u , input dimensions in order to better analyze the performance of the algorithms on different data sets:

- (i) $v = 0$, $u = 90$ (all the 90 input dimensions are irrelevant)
- (ii) $v = 30$, $u = 60$
- (iii) $v = 60$, $u = 30$
- (iv) $v = 90$, $u = 0$ (all the 90 input dimensions are redundant)

4.1.2. Methods

We compared VBLS to four other methods that were previously described in Section 2: (i) ridge regression, (ii) stepwise regression, (iii) PLS regression and (iv) LASSO regression. For ridge regression, we introduced a small ridge parameter value of 10^{-10} to avoid ill-conditioned matrix inversions. We used Matlab’s “stepwisefit” function to run stepwise regression. The number of PLS projections for each data set fit was found by leave-one-out cross-validation. Finally, we chose the optimal tuning parameter in LASSO regression using k -fold cross-validation.

4.1.3. Results

For evaluation, we calculated the prediction error on noiseless test data, using the learned regression coefficients from each technique. Results are quantified as normalized mean squared errors (nMSE). Fig. 4 shows the average prediction error for noiseless test data, given training data where the output noise is either low ($r^2 = 0.9$) or high ($r^2 = 0.8$).

All the algorithms were executed on 10 randomly generated sets of data. The predictive nMSE results reported in Fig. 4 were averaged over the 10 trials. Note that the best training nMSE values possible under the two noise conditions are 0.1 for the low noise case and 0.2 for the high noise case. The training nMSE values were omitted for both graphs, since all algorithms attained training errors that were around the lowest possible values.

From Figs. 4(a) and (b), we see that regardless of output noise level, VBLS achieves either the lowest predictive nMSE

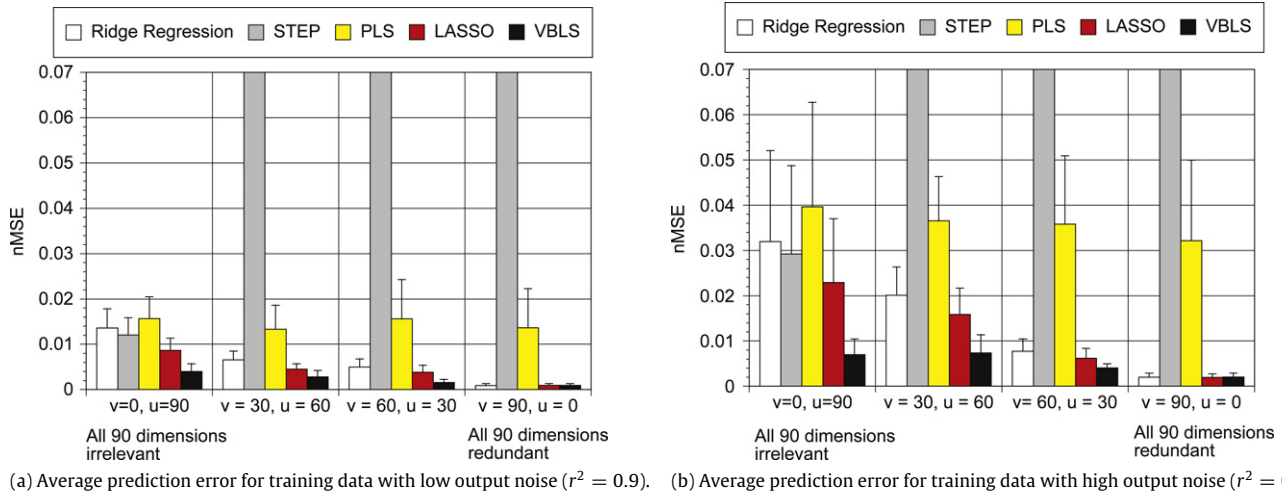


Fig. 4. Average normalized mean squared prediction error for synthetic 100 input-dimensional data, with a varying level of output noise in the training data, averaged over 10 trials. There are 10 relevant input dimensions and a total of 90 redundant and irrelevant input dimensions. The number of redundant dimensions is denoted by v , and the number of irrelevant dimensions is u .

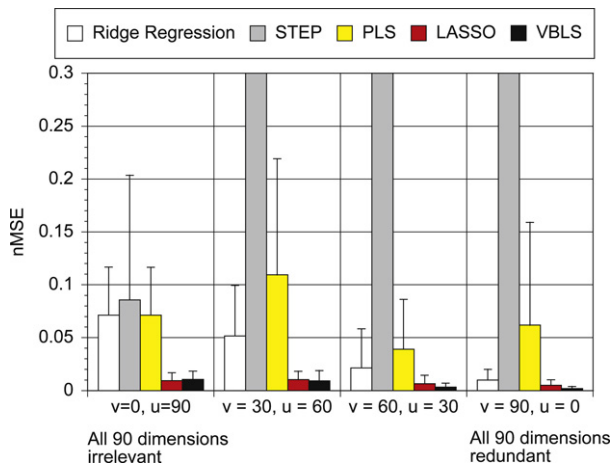


Fig. 5. Average normalized mean squared prediction error for synthetic non-Normal 100 input-dimensional data, with an output noise of $r^2 = 0.9999$ in the training data averaged over 10 trials. There are 10 relevant input dimensions and a total of 90 redundant and irrelevant input dimensions. The number of redundant dimensions is denoted by v , and the number of irrelevant dimensions is u . Each relevant dimension of the training data is drawn from a multi-modal distribution (a mixture of Gaussian distributions), and the output noise is drawn from a Student t -distribution.

value or a predictive nMSE value comparable with that of the other four algorithms. In general, as the number of redundant input dimensions increases and the number of irrelevant input dimensions decreases, the prediction error improves (i.e., it decreases). This may be attributed to the fact that redundancy in the input data provides more “information”, making the problem easier to solve.

The performance of stepwise regression degrades, as the number of redundant dimensions increases, as shown in Fig. 4(a), due to its inability to cope with collinear data. LASSO regression appears to perform quite well, compared with PLS regression and ridge regression. This is unsurprising, given it is known for its ability to produce sparse solutions.

In summary, we can confirm that VBLS performs very well—as well as or better than classical robust regression methods (such as LASSO) on synthetic tests. Interestingly, PLS regression and ridge regression are significantly inferior in problems that have a large number of irrelevant dimensions. Stepwise regression has deteriorated performance as soon as co-linear inputs are introduced.

4.1.4. Non-normal synthetic data

We can also examine synthetic data sets which do not correspond to the generative model (i.e., data and noise that are not generated from Normal distributions) in order to evaluate how dependent our model is on the Normal prior distributions that we assumed.

Synthetic data is generated in a similar fashion as in Section 4.1.1, with 100 dimensions—10 of which are relevant dimensions. The other 90 dimensions are chosen to be either irrelevant or redundant. The first 10 relevant input dimensions were generated from a multi-modal distribution, instead of a Normal distribution. Specifically, each of the relevant 10 input dimensions was drawn from a sum/mixture of 10 Gaussian distributions, with each Gaussian distribution having a different mean and variance, i.e., $x_m \sim \sum_{p=1}^N \text{Normal}(\mu_p, \sigma_p^2)$, for $m = 1, \dots, 10$ where σ_p is drawn randomly from a uniform distribution between 0 and 2 and μ_p is drawn similarly from a uniform distribution between 0 and 2. The second difference between this non-Normal synthetic data set and the data set used in Section 4.1.1 is the additive output noise. Instead of Gaussian distributed noise, noise drawn from a Student t -distribution was added to the outputs. We chose a noise level of $r^2 = 0.9999$ for the output noise, such that the noise was scaled to the variance of the noiseless outputs \bar{y} . Redundant and irrelevant data vectors were added to the input data in a similar way as described in Section 4.1.1. The test data was created in a similar manner, except the input and output data were left noise-free. As in Section 4.1.1, we considered synthetic training data with $N = 1000$ data samples and a synthetic test data set with 20 data samples.

Fig. 5 shows the prediction nMSE values, averaged over 10 trials. We can observe that both VBLS and LASSO outperform the other classical regression methods on non-Normal synthetic data sets. This figure demonstrates that even for data sets that do not follow the Normal prior distributions assumed in our generative model, VBLS continues to perform quite competitively.

4.2. EMG prediction from neural firing

4.2.1. Data sets

We investigated data from two different neurophysiological experiments. In the first experiment by Sergio and Kalaska (1998), a monkey moved a manipulandum in a center-out task in eight different directions, equally spaced in a horizontal planar circle of 8 cm radius. A variation of this experiment held the manipulandum

rigidly in place, while the monkey applied isometric forces in the same eight directions. In both conditions (whether the monkey was applying a movement or an isometric force), feedback was given through visual display on a monitor. Neural activity for 71 M1 neurons was recorded in all conditions, along with the EMG outputs of 11 muscles.¹ After preprocessing, we obtained a total of 2320 data samples for each neuron/muscle pair, collected over all eight directions and for both movement and isometric force conditions. Each data sample consisted of the average firing rates from a particular neuron (averaged over a window of 10 ms) and the corresponding EMG activation² from a particular muscle. A sampling interval of 10 ms was used. For each sample in this data set, a delay of 50 ms between M1 cortical neural firing and EMG muscle activation was empirically chosen, based on estimates from measurements.

The second experiment, conducted by Kakei et al. (1999, 2001), involved a monkey trained to perform eight different combinations of wrist flexion-extension, and radial-ulnar movements while in three different arm postures (pronated, supinated and midway between the two). These experiments resulted in two data sets. For the first, the EMG outputs of 7 contributing muscles³ were recorded, along with the neural data of 92 M1 neurons at all three wrist postures, resulting in 2616 data samples for each neuron/muscle pair. As for the Sergio & Kalaska data set, each data sample consisted of the average firing rates from a particular neuron (averaged over a window of 10 ms) and the corresponding EMG activation from a particular muscle. A sampling interval of 10 ms was used. For each sample in this data set, a delay of 20 ms⁴ between M1 cortical neural firing and EMG muscle activation was chosen empirically, based on estimates from measurements. The second data set also included EMG outputs of the same 7 muscles, but this time contained the recorded spiking data of 72 PM neurons at the three wrist postures. After preprocessing, this second data set had 2592 data samples for each neuron/muscle pair. For each sample, a delay of 30 ms⁵ between PM cortical neural firing and EMG muscle activation was assumed.

4.2.2. Methods

As a baseline comparison, EMG reconstruction was obtained through a combinatorial search over possible regression models. This approach served as our baseline study (referred to as ModelSearch in the figures). A particular model is characterized by a subset of neurons that is used to predict the EMG data. For the Sergio & Kalaska data, given 71 neurons, the number of possible models that exist for a particular muscle is:

$$\sum_{m=1}^{71} \binom{71}{m}.$$

¹ The 11 arm muscles analyzed included the (1) surraspinatus, (2) infraspinatus, (3) subscapularis, (4) rostral trapezius, (5) caudal trapezius, (6) posterior deltoid, (7) medial deltoid, (8) anterior deltoid, (9) triceps medial head, (10) brachialis and (11) pectoralis muscles.

² EMG was recorded from pairs of shoulder and elbow muscles, implanted percutaneously with Teflon-coated single-stranded stainless steel wires. EMG activity was amplified, rectified and integrated (over 10 ms bins) to generate summed histograms of activity. The EMG data had no physically meaningful units.

³ EMG was recorded using pairs of single-stranded stainless steel wires placed transcutaneously into each muscle. The 7 arm muscles considered were the (1) extensor carpi ulnaris (ECU), (2) extensor digitorum 2 and 3 (ED23), (3) extensor digitorum communis (EDC), (4) extensor carpi radialis brevis (ECRB), (5) extensor carpi radialis longus (ECRL), (6) abductor pollicis longus (APL), and (7) flexor carpi radialis (FCR) muscles.

⁴ The results of our analyses are insensitive to a delay in the range of 20–60 ms, since there was only a very small numerical difference between the quality of the fit of the data in this interval. Delays of 50 ms or higher are physiologically more plausible.

⁵ Within a delay range of 30–80 ms, there is no real difference in the quality of fit of our analyses.

Since the order of the contributing neurons is not important, the above expression lists the combinations instead of permutations of neurons. This value is too large for an exhaustive search. Therefore, we considered only possible combinations of up to 20 neurons, which required several weeks of computation on a 30-node cluster computer. The optimal predictive subset of neurons was determined from a series of 8-fold cross-validation sets.

For both data sets, the cross-validation procedure used in the baseline study was used in order to determine the optimal subset of neurons. Cross-validation was done in the context of the behavioral experiments and not in a statistically randomized way. For the Sergio & Kalaska experiment, the data was separated into different force categories (isometric force versus force generated during movement) and movement directions in space. Thus, cross-validation asked the meaningful question of whether isometric and movement conditions are predictive of each other and whether there is spatial generalization. Similarly, for the Kakei et al. experiment, data was separated into directional movements at the wrist (supinated, pronated and midway between the two wrist movements) and directional movements in space, which again allowed cross-validation to make meaningful statements about generalization over postures and space.

Fig. 6 shows how these 8 cross-validation sets are constructed from the Sergio & Kalaska data. This baseline study (i.e., ModelSearch) served as a comparison for ridge regression, stepwise regression, PLS regression, LASSO regression and VBLS. These five algorithms used the same validation sets employed in the baseline study. Again, as described in Section 4.1.2, ridge regression was implemented using a small ridge regression parameter of 10^{-10} , in order to avoid ill-conditioned matrices. We used Matlab's "stepwisefit" to run stepwise regression, and the number of PLS projections for each data fit was found by leave-one-out cross-validation. The average normalized mean squared error values depicted in Fig. 9(a) demonstrate how well each algorithm performs, averaging the generalization performances over all the cross-validation sets from Fig. 6.

The average number of relevant neurons⁶ (i.e., not including irrelevant neurons and neurons providing redundant information), shown in Fig. 11(a), was calculated by averaging over the number of relevant neurons in each of the 8 training sets in Fig. 6.

The final set of relevant neurons, used in Fig. 13(a) to calculate the percentage match of relevant neurons relative to those found by the baseline study (ModelSearch), was reached for each algorithm (except VBLS) by taking the common neurons found to be relevant over the 8 cross-validation sets. The relevant neurons found by VBLS and reported in Fig. 13(a) were obtained by using the entire data set, since no cross-validation procedure is required by VBLS (i.e., dividing the data into separate training and test sets is not necessary). As with all Bayesian methods, VBLS performs more accurately as the data size increases, without the danger of overfitting. Inference of relevant neurons in PLS was based on the subspace spanned by the PLS projections, while relevant neurons in VBLS were inferred from *t*-tests on the regression parameters, using a significance of $p < 0.05$. Stepwise regression determined the number of relevant neurons from the inputs that were included in the final model. Note that since ridge regression retained all input dimensions, this algorithm was omitted in relevant neuron comparisons.

Analogous to the first data set, a combinatorial analysis was performed on the Kakei et al. M1 neural and PM neural data sets

⁶ Relevant neurons are those that contribute to the regression result in a statistically sound way, according to a *t*-test with $p < 0.05$. It should be noted that in noisy data, two neurons that carry the same signal, but have independent noise will usually both remain significant in our algorithm, as the combined signal of both neurons helps to average out the noise in the spirit of population coding.

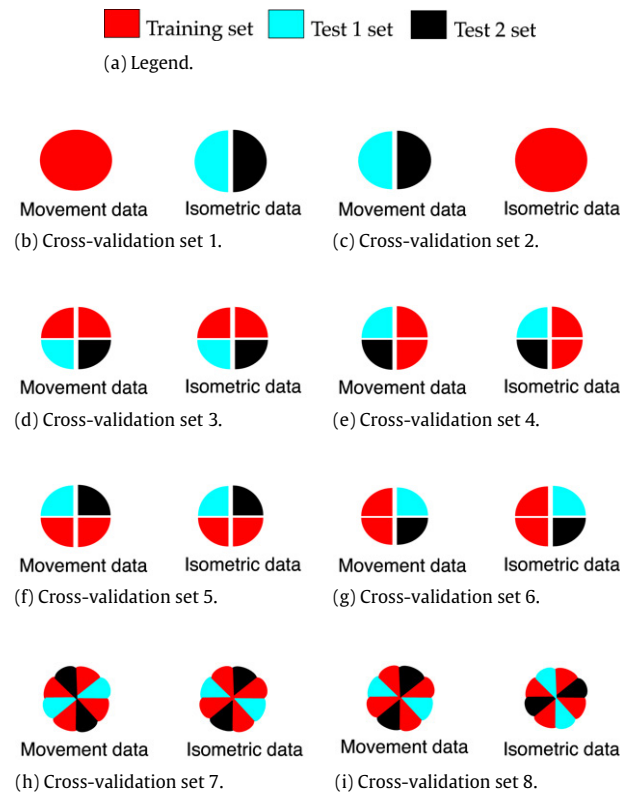


Fig. 6. Details of how the 8 cross-validation sets are created from the Sergio & Kalaska M1 neural data set. For each type of force applied by the monkey to the manipulandum, there are 8 possible directions that the manipulandum could have been moved. Each circle shown above is partitioned into 8 equal portions, corresponding to the 8 directional movements and numbered in increasing order (clockwise) starting from 1.

in order to determine the optimal set of M1 and PM neurons contributing to each muscle (i.e. producing the lowest possible prediction error) in a series of 6-fold cross-validation sets. Figs. 7 and 8 show the 6 cross-validation sets used for the M1 and PM neural data sets. PLS, stepwise regression, ridge regression and VBLS were applied using the same cross-validation sets, employing the same procedure described for the Sergio & Kalaska data set. The average normalized mean squared error values shown in Figs. 9(b) and 10 illustrate the generalization performance of each algorithm, averaged over all the cross-validation sets shown in Figs. 7 and 8.⁷ The average number of relevant neurons shown in Figs. 11(b) and 12 was calculated by averaging over the number of relevant neurons found in each of the 6 training sets from Figs. 7 and 8. As for the Sergio & Kalaska data set, the final set of relevant neurons, used in Figs. 11(b) and 12, was obtained for each algorithm (except VBLS) by taking the common neurons found to be relevant over the 6 cross-validation sets.

4.2.3. Results

Figs. 9 and 10 show that VBLS resulted in a generalization error, comparable to that produced by ModelSearch (i.e., the baseline study). In the Kakei et al. M1 and PM neural datasets, all algorithms performed similarly, as we see on the right hand side of Figs. 9(b) and 10. However, ridge regression, stepwise regression, PLS regression and LASSO regression performed far worse on the Sergio & Kalaska M1 neural dataset, with ridge regression attaining the worst error, as we see on the right

hand side of Fig. 9(a). Such performance is typical for traditional linear regression methods on ill-conditioned high-dimensional data, motivating the development of VBLS.

Interestingly, in Fig. 9(b), we observe that the prediction errors of ridge regression and of the baseline study (i.e. ridge regression using a selected subset of M1 neurons) are quite similar for the Kakei et al. M1 neural data set. This suggests that, for this particular data set, there is little advantage in performing a time-consuming manual search for the optimal subset of neurons. A similar observation can be made for the Kakei et al. PM neural data set when examining Fig. 10, although this effect is less pronounced in the PM neural data set. In contrast, Fig. 9(a) shows a sharp difference between the predictive error values of ridge regression and the baseline study's combinatorial-like model search. This may be attributed to the fact that the Sergio & Kalaska M1 neural data set is somehow much richer and hence, more challenging to analyze.

The average number of relevant M1 neurons found by VBLS was slightly higher than the baseline study, as seen in Fig. 11. This is unsurprising, since the baseline studies did not consider all possible combination of neurons. For example, the baseline study for the Sergio & Kalaska data set considered possible combinations of up to only 20 neurons, instead of the full set of 71 neurons. In particular, notice that in Figs. 11(b) and 12, small amounts of the total 92 M1 neurons and 72 PM neurons were found to be relevant by the baseline study for certain muscles (e.g., muscles 1, 6 and 7).

We compared the relevant neurons identified by each algorithm with those found by the baseline combinatorial-like model search, in an attempt to evaluate how well each algorithm performed in comparison with the model search approach. Table 1 shows the percentage of neuron matches found by each algorithm, averaged over all the muscles of the data set. The percentage of neuron

⁷ Note that the partitioning of the data into training and test cross-validation sets was essentially an intuitive process that tried to use insights from the different experimental conditions in which the data was collected.

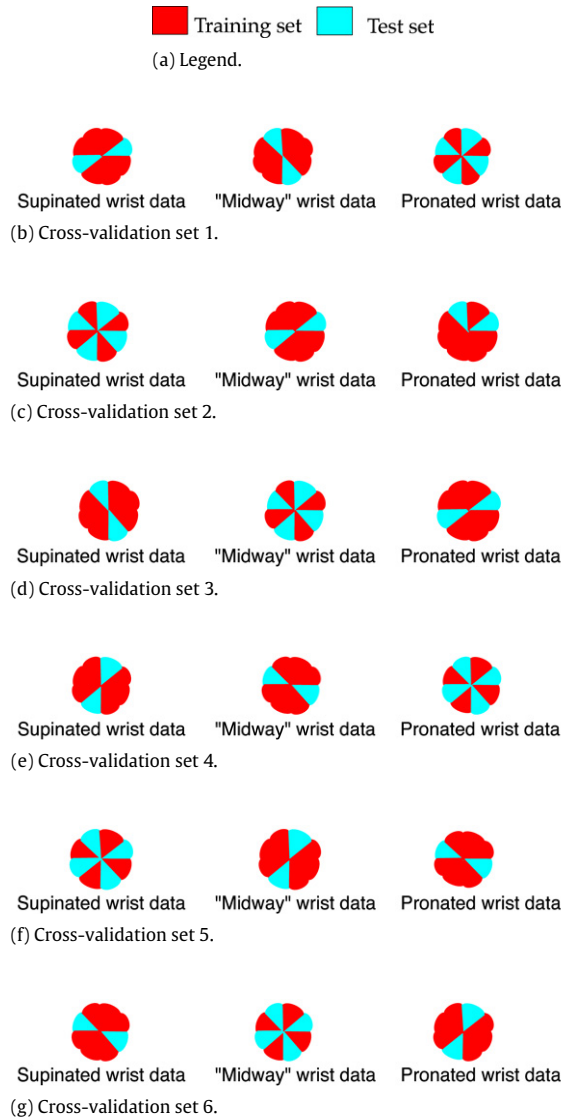


Fig. 7. Details of how the 6 cross-validation sets are created from the Kakei et al. M1 neural data set. For each of the three wrist positions, there are 8 possible directional movements. Each circle shown above is partitioned into 8 equal sections, corresponding to the 8 directional movements and numbered in increasing order (clockwise) starting from 1.

Table 1

Percentage of neuron matches found by each algorithm, as compared to those found by the baseline study (ModelSearch), averaged over all muscles in each data set

	STEP (%)	PLS (%)	LASSO (%)	VBLS (%)
Sergio and Kalaska (1998) M1 neural data set	7.2	7.4	6.4	94.2
Kakei et al. (1999) M1 neural data set	65.1	42.9	80.6	94.4
Kakei et al. (1999) PM neural data set	22.9	14.2	44.5	91.5

The percentage of relevant neuron matches for an algorithm is calculated by considering the list of relevant neurons found by the baseline study. The number of neurons in this list that the algorithm was successfully at identifying as relevant was counted, and the percentage of neuron matches calculated using this value.

matches was calculated by considering the list of relevant neurons found by the baseline study. The number of neurons in this list that the algorithm was successful at identifying as relevant was counted, and the percentage of relevant neuron matches was calculated using this value.

Table 1 shows that the relevant neurons identified by VBLS coincided at a very high percentage with those of the baseline model, while stepwise and PLS regression had inferior outcomes. This table illustrates that VBLS was able to reproduce comparable results to a combinatorial-like model search approach. However, the main advantage of VBLS arises in its speed: VBLS took 8 h

for all validation sets on a standard PC, while the model search took weeks on a cluster computer. LASSO regression matched a high percentage of the relevant M1 and PM neurons in the Kakei et al. data set, but fared far worse on the Sergio & Kalaska data set. These percentage values for the Kakei et al. data sets are perhaps inflated and should be given less consideration, since the numbers of relevant M1 and PM neurons found by the baseline study are relatively small for certain muscles. Figs. 13(a), (b) and 14 show the detailed breakdown of percentage M1 and PM neuron matches for each algorithm on each muscle. The consistent and good generalization properties of VBLS on all neural data sets, as

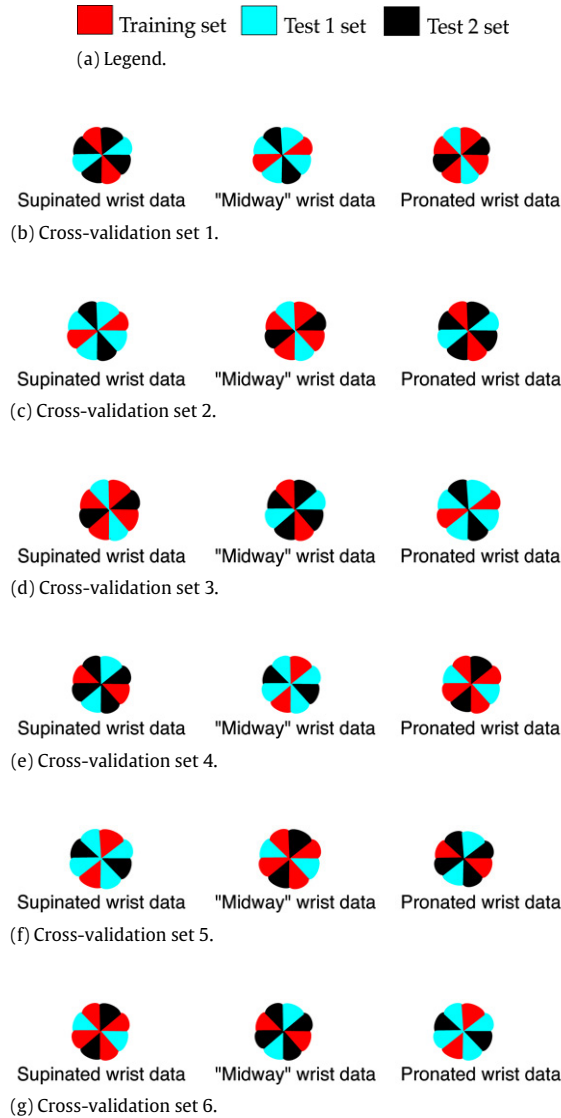
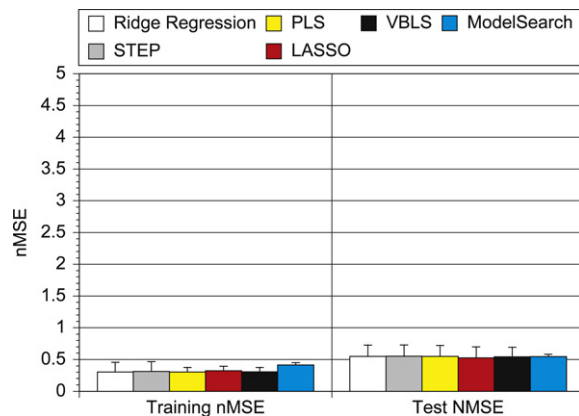


Fig. 8. Details of how the 6 cross-validation sets are created from the Kakei et al. PM neural data set. For each of the three wrist positions, there are 8 possible directional movements. Each circle shown above is partitioned into 8 equal sections, corresponding to the 8 directional movements and numbered in increasing order (clockwise) starting from 1.



(a) Average error on Sergio and Kalaska (1998) M1 neural data set.

(b) Average error on Kakei et al. (1999) M1 neural data set.

Fig. 9. Normalized mean squared error for M1 neurons, averaged over all cross-validation sets and over all muscles. Fig. 6 shows the 8 cross-validation sets used in the Sergio and Kalaska (1998) M1 neural data set, and Fig. 7 shows the 6 cross-validation sets used for the Kakei et al. (1999) M1 neural data set.

